KB금융그룹 | 국민의 평생 금융파트너

DAO객체 만들어졌으니 갖고 왔으니
BL해보자

2025년 상반기 K-디지털 트레이닝

비즈니스 계층

서비스 레이어 BL

[KB] IT's Your Life

KB 국민은행

DTO, Data Transfer Object

- VO, Value Object
  - 데이터베이스 테이블의 모양을 그대로 반영한 객체
  - Dao에서 사용

- DTO
  - Dao를 제외한 나머지 계층(Service, Controller, View 등)에서 사용
  - 테이블에 없지만 비즈니스 로직에 필요한 추가 정보도 포함 가능

- VO와 DTO간 상호 변환 기능 필요
  - VO → DTO
  - DTO → VO

JPA에서
VO와 DTO를 분리하는 이유

vo의 엔티티 라이프사이클이
hibernate구현체의 의해서
자동으로 닫기 때문에
나중에 DTO가 아닌 VO를 사용한
뷰에서 못 사용할 수 있기에

분리한다

mybatis는 VO생명주기를
개발자가 하기때문에 둘의 분리가
필수가 아니다

굳이 엄격하게 구분해야하냐 그냥 통일하지.

프렘웤마다 다름. jpa hibernate는 엄격하게 구분해야함.

mybatis는 둘의 구분을 허물어도된다. 문법적인 문제는 없음

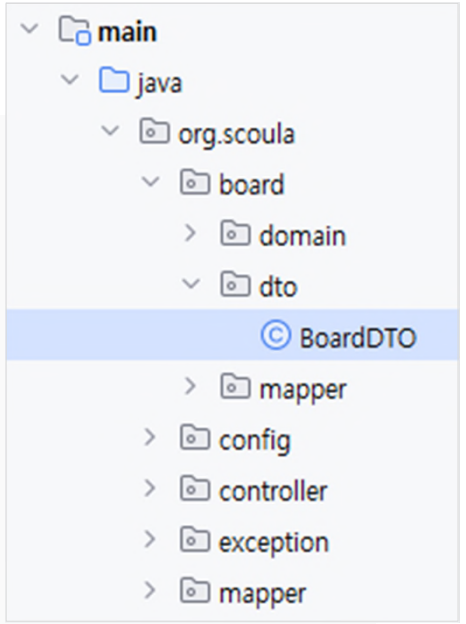
그래도 구분해서 사용해보자

BoardDTO.java

BL에 필요한 객체

```
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;
```

현재는 VO와 같지만 추가될 예정



BoardDTO.java

DTO객체가 없을때니까 static

```
// 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();
}
}
```

널이면 그대로 반환

아니면 빌더를 써서 똑같이 필드를 채움

변환된거 반환

이때는 인스턴스가 있을때니까 일반 메소드

빌더 패턴을 사용하여 똑같이 ㅂ변환 후 반환

이거 만드는 것도 귀찮다.
이를 만들어주는 라이브러리도 존재한다.

다음에 소개 ㄱㄱ

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"})
@MapperScan(basePackages  = {"org.scoula.board.mapper"})
@ComponentScan(basePackages={ "org.scoula.board.service" })
public class RootConfig {

        ...
}
```

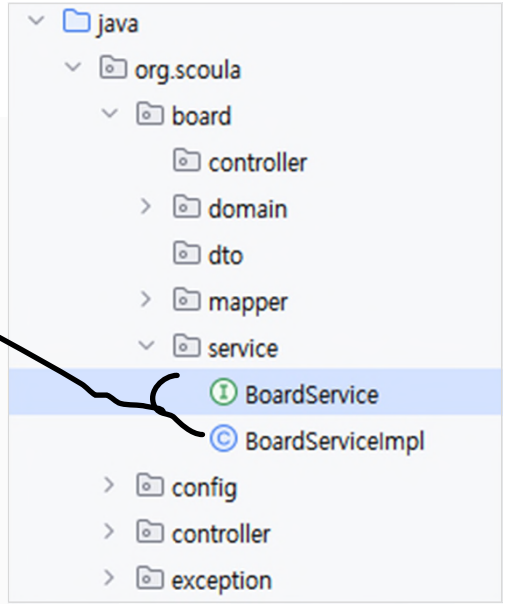
✓

서비스 만들어보자

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);

}
```

Board VO에 대한
CRUD
메서드 인터페이스로
선언하고 구현까지



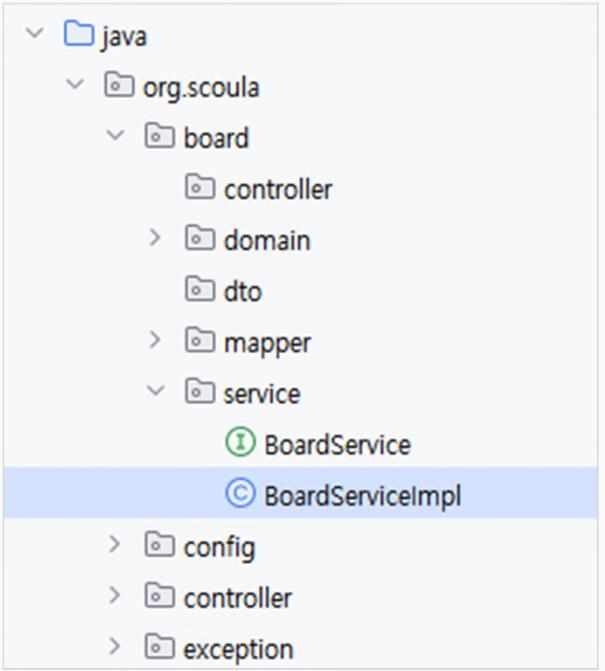
BoardServiceImpl.java

```
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

@Component 자식임



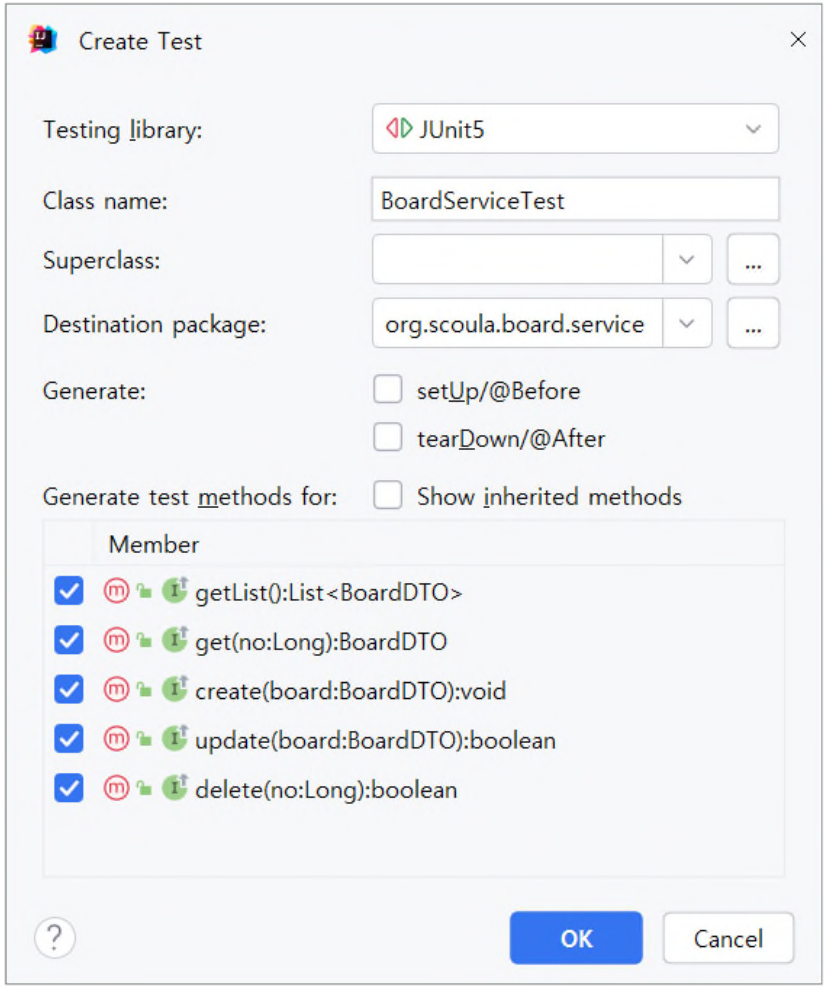
@Autowired는 일반객체에서 테스트할때만 권장된다.
비궈장!

BoardServiceImpl.java

```
    @Override
    public List<BoardDTO> getList() {
        return List.of();
    }


    @Override
    public BoardDTO get(Long no) {
        return null;
    }

    @Override
    public void create(BoardDTO board) {

    }

    @Override
    public boolean update(BoardDTO board) {
        return false;
    }

    @Override
    public boolean delete(Long no) {
        return false;
    }
}
```

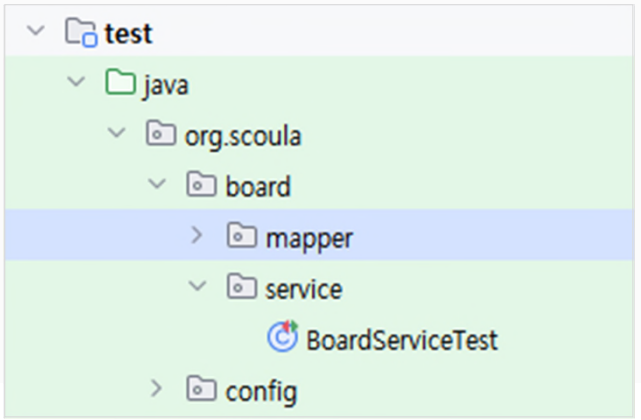
비즈니스 계층의 구현과 테스트

test :: BoardServiceTest.java



test :: BoardServiceTest.java

```
...

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

        @Autowired
        private BoardService service;

            ...

}
```



JDBC는 예외처리가 필수

마이바티스는 jdbc위에서 돌아간다.

지금까지는 예외처리 안했는데 잘되네?
마이바티스가 해주는 또다른 일
SQLException=>runtimeException으로 변환해줌
runtimeException은 처리가 강제사항이 아님!
왜 이렇게 했냐 매번 trycatch묶으면 힘들잖아
웹앱 특성상 예외는 중앙처리하기로 했잖아.
예외를 받아서 중앙처리. 매번 catch하지 말고
한곳에서 처리하자!
== 예외처리가 필수가 아니게 만들어줌.
advice가 처리하도록 해놨죠?

BoardServiceImpl.java

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

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

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

스트림을 활용하면 쉽게 바꿀 수 있다.

getList=> List<BoardVO>
stream=> Stream<BoardVO>
map//element타입 바꾼다.(BoardDTO::of)=>stream<BoardDTO>
toList=>List<BoardDTO> 변환됨.

test :: BoardServiceTest.java

```
@Test
public void getList() {

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

}
```

```
INFO  org.scoula.board.service.BoardServiceImpl(getList:20) - getList.....
INFO  jdbc.sqlonly(sqlOccurred:228) - select * from tbl_board order by no desc

INFO  org.scoula.board.service.BoardServiceTest(getList:26) - BoardDTO(no=7, title=새로 작성하는 글, content=새로 작성하는 내용, writer=user0, regDate=Wed Jan 15 12:56:07 KST 2025, updateDate=Wed Jan 15 12:56:07 KST 2025)
INFO  org.scoula.board.service.BoardServiceTest(getList:26) - BoardDTO(no=6, title=새로 작성하는 글, content=새로 작성하는 내용, writer=user0, regDate=Wed Jan 15 12:52:46 KST 2025, updateDate=Wed Jan 15 12:52:46 KST 2025)
INFO  org.scoula.board.service.BoardServiceTest(getList:26) - BoardDTO(no=5, title=수정된 제목, content=수정된 내용, writer=user00, regDate=Wed Jan 15 11:53:10 KST 2025, updateDate=Wed Jan 15 12:57:42 KST 2025)
INFO  org.scoula.board.service.BoardServiceTest(getList:26) - BoardDTO(no=4, title=테스트 제목4, content=테스트 내용4, writer=user00, regDate=Wed Jan 15 11:53:10 KST 2025, updateDate=Wed Jan 15 11:53:10 KST 2025)
INFO  org.scoula.board.service.BoardServiceTest(getList:26) - BoardDTO(no=2, title=테스트 제목2, content=테스트 내용2, writer=user00, regDate=Wed Jan 15 11:53:10 KST 2025, updateDate=Wed Jan 15 11:53:10 KST 2025)
INFO  org.scoula.board.service.BoardServiceTest(getList:26) - BoardDTO(no=1, title=테스트 제목1, content=테스트 내용1, writer=user00, regDate=Wed Jan 15 11:53:10 KST 2025, updateDate=Wed Jan 15 11:53:10 KST 2025)
```

BoardServiceImpl.java

하나 변환하는 메서드 구현

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

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

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

of 확신
empty 확실히null
ofNullable null일수도?

리스트와 다르게 비면 null이 반환됨.
null안정성 체크 오바임.

서비스쪽에서 추가적인 코드가 발생하므로

optional객체에 래핑하여 반환하자!

만약 예외가 발생하더라도 advice로 이동하게끔
전시간에 해놨죠.

orElseThrow 데이터가 있으면 리턴
없으면 예외 일부러 발생.
매개변수로는 어떤 예외인지알려주기위한 정보
( ) =>new NoSuchElementException( )
런타임 예외이다.advice로 전달됨

컨트롤러에서는 널 검사와 예외처리할 필요 없어짐!
호출하는 쪽에서 개이득

근데 지금 들어가있는 인자가 디폴트긴해;;

test :: BoardServiceTest.java

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

```
INFO  org.scoula.board.service.BoardServiceImpl(get:31) - get.....1
INFO  jdbc.sqlonly(sqlOccurred:228) - select * from tbl_board where no = 1

INFO  org.scoula.board.service.BoardServiceTest(get:32) - BoardDTO(no=1, title=테스트 제목1, content=테스트 내용1, writer=user0
0, regDate=Wed Jan 15 11:53:10 KST 2025, updateDate=Wed Jan 15 11:53:10 KST 2025)
```

BoardServiceImpl.java

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

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

  BoardVO vo = board.toVo();
  mapper.create(vo);
  board.setNo(vo.getNo());
}
```

DTO->VO

set

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());
}
```

DTO로 생성

INFO org.scoula.board.service.BoardServiceImpl(create:43) - create.....BoardDTO(no=null, title=새로 작성하는 글, content=새로 작성하는 내용, writer=user1, regDate=null, updateDate=null)
INFO jdbc.sqlonly(sqlOccurred:228) - **insert into tbl_board (title, content, writer) values ('새로 작성하는 글', '새로 작성하는 내용', 'user1')**

INFO jdbc.sqlonly(sqlOccurred:228) - **SELECT LAST_INSERT_ID()**

INFO org.scoula.board.service.BoardServiceTest(create:45) - 생성된 게시물의 번호: 8

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(get:32) - get.....1
INFO  jdbc.sqlonly(sqlOccurred:228) - select * from tbl_board where no = 1

INFO  org.scoula.board.service.BoardServiceImpl(update:52) - update.....BoardDTO(no=1, title=제목 수정합니다., content=테스트 내용1, writer=user00, regDate=Wed Jan 15 11:53:10 KST 2025, updateDate=Wed Jan 15 11:53:10 KST 2025)
INFO  jdbc.sqlonly(sqlOccurred:228) - update tbl_board set title = '제목 수정합니다.', content = '테스트 내용1', writer = 'user00', update_date = now() where no = 1

INFO  org.scoula.board.service.BoardServiceTest(update:54) - update RESULT: true
```

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:60) - delete....2
INFO  jdbc.sqlonly(sqlOccurred:228) - delete from tbl_board where no = 2

INFO  org.scoula.board.service.BoardServiceTest(delete:61) - delete RESULT: true
```