

비즈니스 계층

비즈니스 계층 : 고객의 요구사항을 반영하는 계층으로
프레젠테이션 계층과 영속 계층의 중간 다리 역할을 담당.

영속 계층 : 데이터베이스를 기준으로 설계를 나누어 구현.

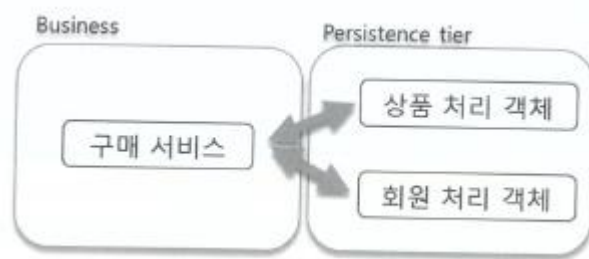
비즈니스 계층 : 로직을 기준으로 해서 처리.

예) '쇼핑몰에서 상품을 구매한다'고 가정

해당 쇼핑몰의 로직 예 : 물건을 구매한 회원에게는 포인트를 올려준다

영속 계층의 설계 : '상품'과 '회원'으로 나누어서 설계.

비즈니스 계층 설계 : 상품 영역과 회원 영역을 동시에 사용하여 하나의 로직을 처리.

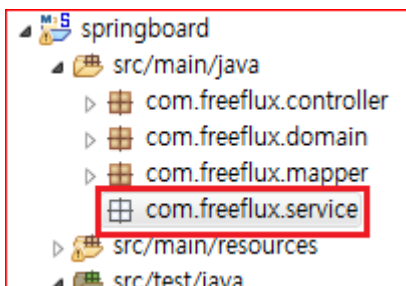
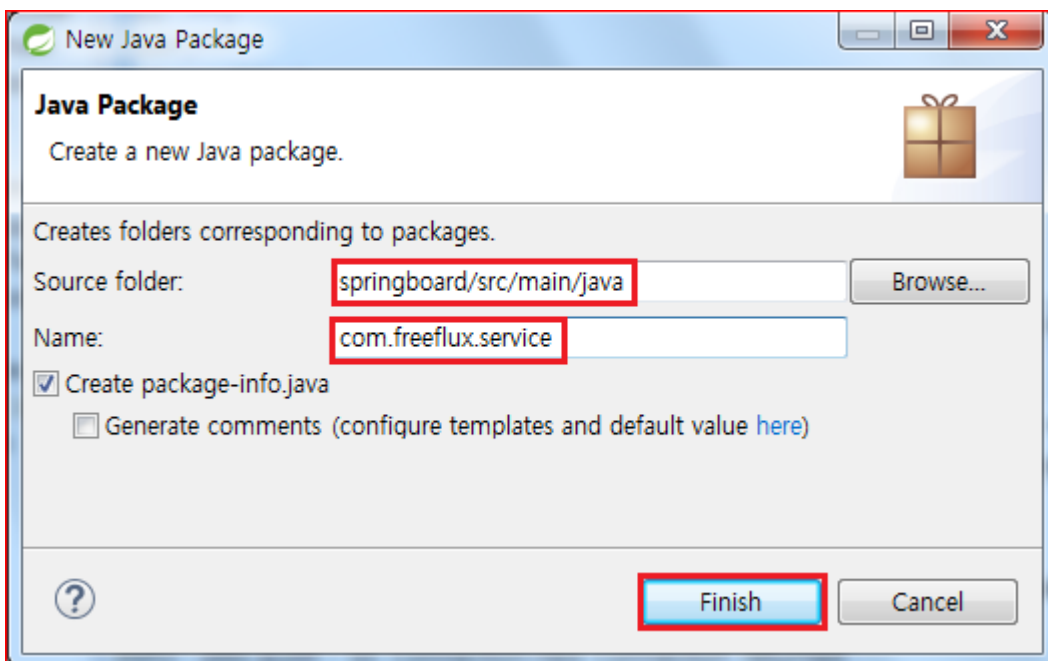
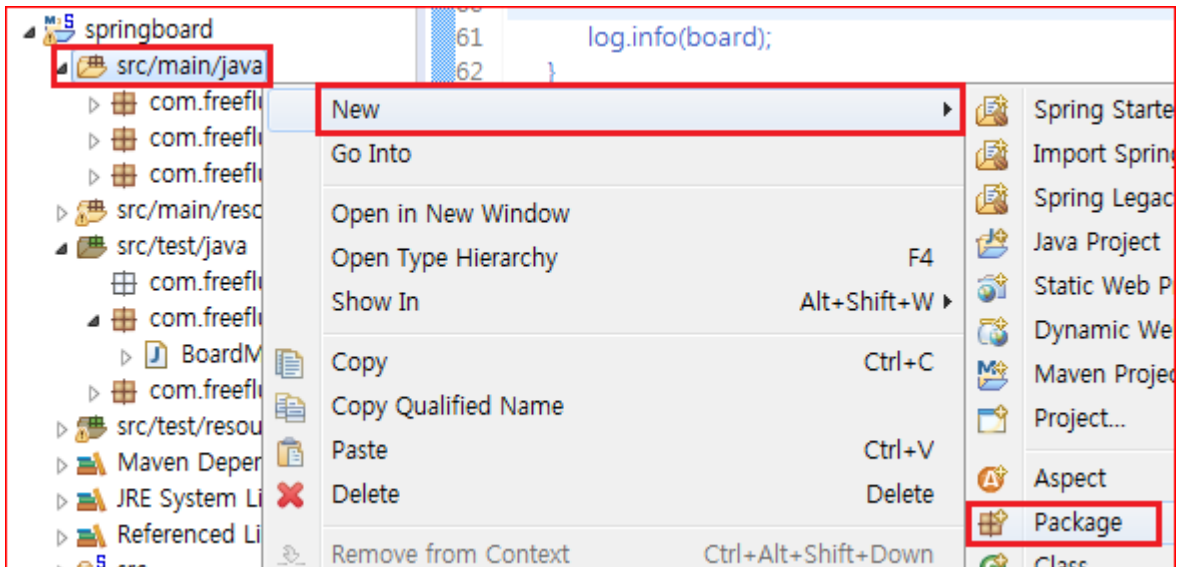


현재 단일 테이블을 사용하고 있기 때문에 위와 같은 구조는 아니지만,
설계를 할 때는 원칙적으로 영역을 구분하여 작성.

일반적으로 비즈니스 영역에 있는 객체들을 '서비스(Service)'라는 용어를 사용.

1. 비즈니스 계층의 설정

1. 비즈니스 계층을 위한 패키지 작성



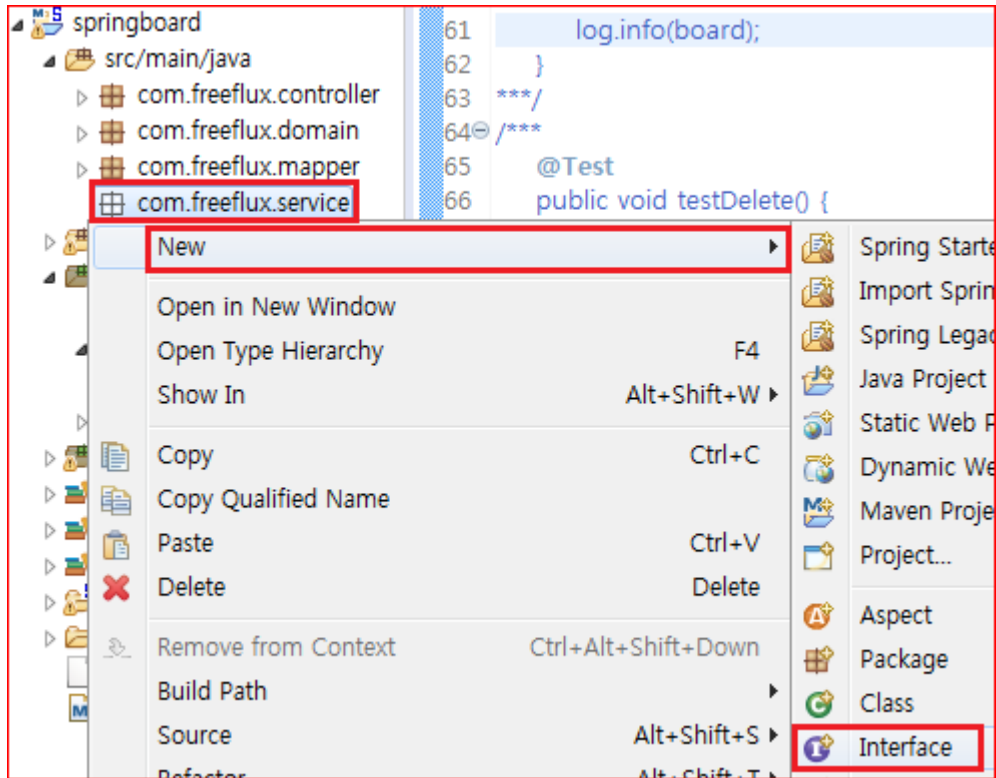
2. 비즈니스 계층 설계

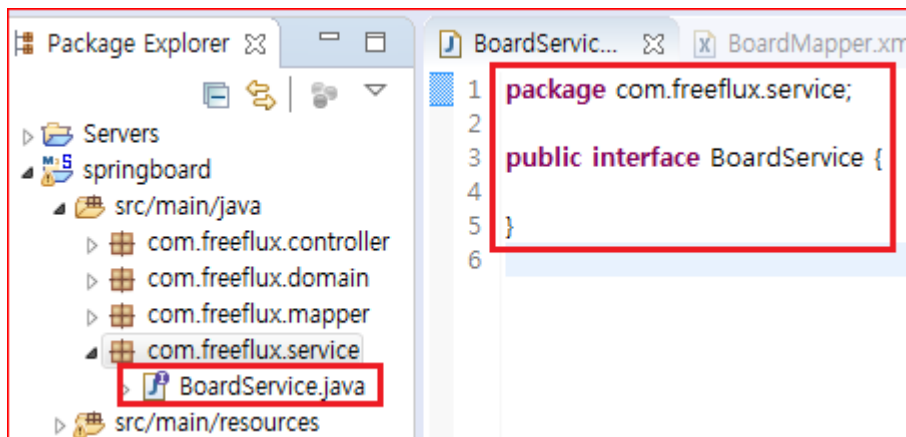
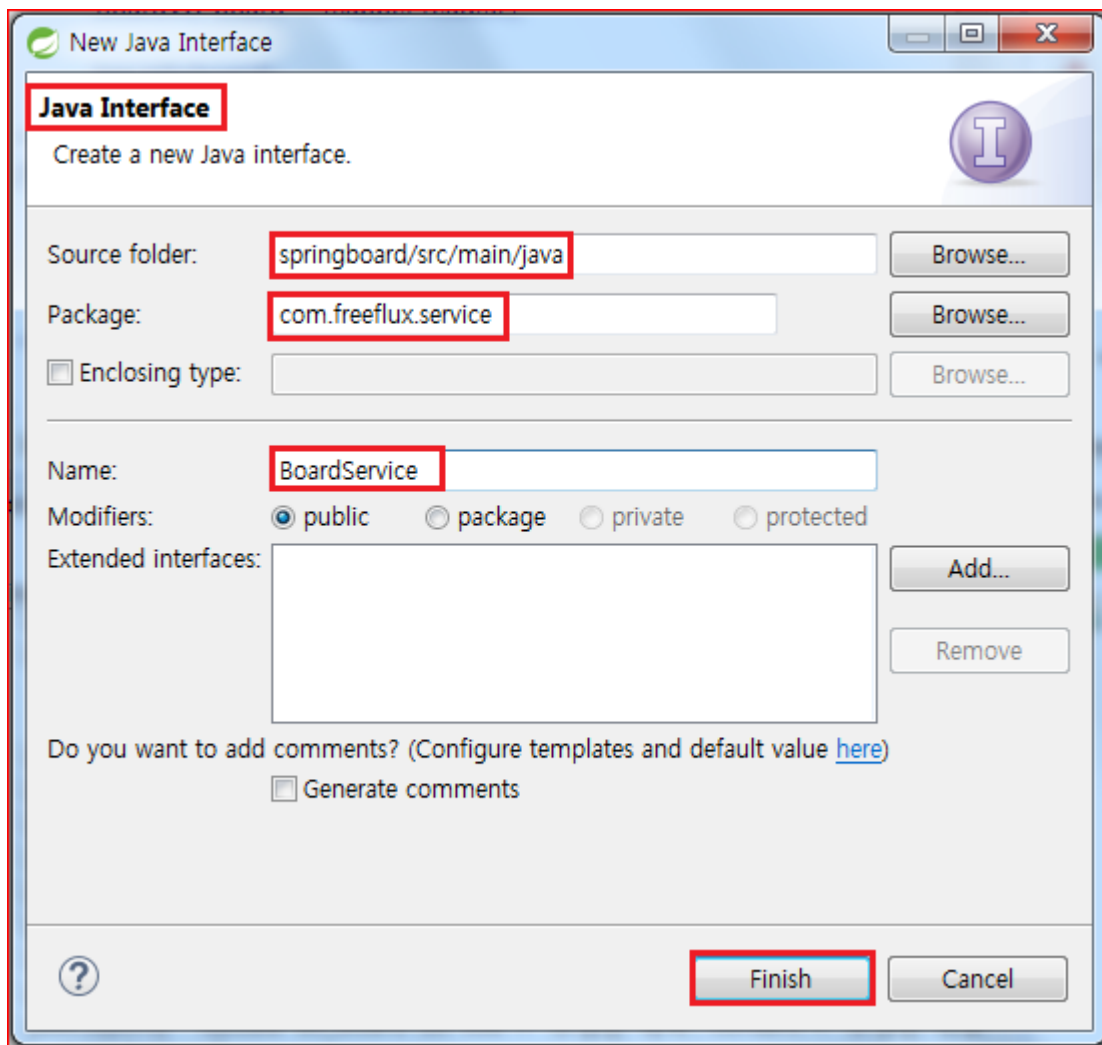
각 계층 간의 연결은 인터페이스를 이용하여 느슨한(loose) 연결(결합)을 한다.

게시물은

BoardService 인터페이스와

인터페이스를 구현 받는 BoardServiceImpl 클래스를 선언





New Java Class

Java Class

Create a new Java class.

Source folder: Browse...

Package: Browse...

☐ Enclosing type: Browse...

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)
☒ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

Finish Cancel

Implemented Interfaces Selection

Choose interfaces:

Matching items:

- BoardMapper - com.freeflux.mapper - springboard/si
- BoardService - com.freeflux.service - springboard/src**
- Body - org.springframework.http.StreamingHttpOutp
- ServletBuilder - org.springframework.http.RequestEntity

com.freeflux.mapper - springboard/src/main/java

Add OK Cancel

New Java Class

Java Class
Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)
☒ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

Package Explorer

- Servers
- springboard
 - src/main/java
 - com.freeflux.controller
 - com.freeflux.domain
 - com.freeflux.mapper
 - com.freeflux.service
 - BoardService.java
 - BoardServiceImpl.java**
 - src/main/resources

BoardService... BoardService... BoardMapper...

```
1 package com.freeflux.service;  
2  
3 public class BoardServiceImpl implements BoardService {  
4  
5     public BoardServiceImpl() {  
6         // TODO Auto-generated constructor stub  
7     }  
8  
9 }  
10
```

3. BoardService 인터페이스에 메서드 선언 추가

BoardService 메서드 설계 시,

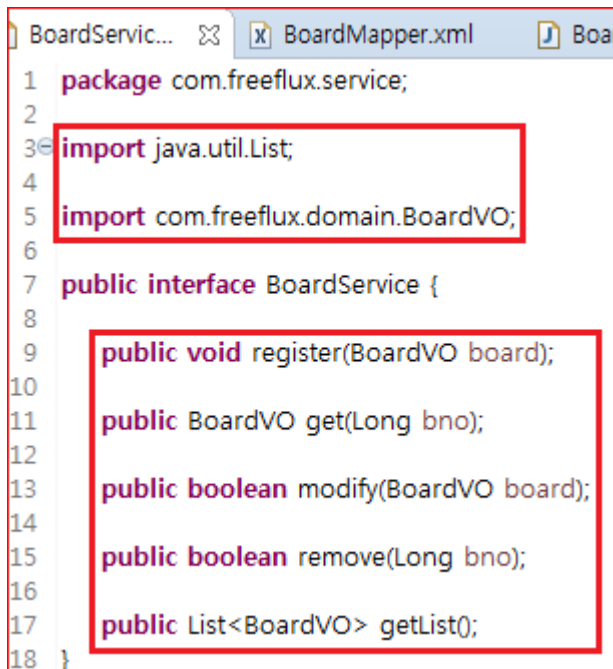
메서드 이름은 현실적인 로직의 이름을 사용.

명백하게 반환해야 하는 데이터가 있는(select) 메서드 : 리턴 타입을 명시.

get(): 게시물은 특정한 게시물을 가져오는 메서드.

getList() : 전체 리스트를 구하는 메서드.

이 두 가지 메서드는 처음부터 리턴 타입을 결정하여 진행.



```
1 package com.freeflux.service;
2
3 import java.util.List;
4
5 import com.freeflux.domain.BoardVO;
6
7 public interface BoardService {
8
9     public void register(BoardVO board);
10
11     public BoardVO get(Long bno);
12
13     public boolean modify(BoardVO board);
14
15     public boolean remove(Long bno);
16
17     public List<BoardVO> getList();
18 }
```

4. BoardServiceImpl 클래스 내부 코드 작성

상세 내용은 추후에, 테스트 할 수 있는 로그만 작성.

@Service :

계층 구조상 주로 비즈니스 영역을 담당하는 객체임을 표시하기 위해 사용.

작성된 어노테이션은 패키지를 읽어 들이는 동안 처리.

BoardServiceImpl 이 정상적으로 작동하기 위해서는 BoardMapper 객체가 필요.

@Autowired 또는 Setter 를 이용하여 처리할 수도 있다.

Lombok 을 이용하여 처리.

```

1 package com.freeflux.service;
2
3 import java.util.List;
4
5 import com.freeflux.domain.BoardVO;
6
7 public class BoardServiceImpl implements BoardService {
8
9     public BoardServiceImpl() {
10         // TODO Auto-generated constructor stub
11     }
12
13     @Override
14     public void register(BoardVO board) {
15         // TODO Auto-generated method stub
16     }
17
18     @Override
19     public BoardVO get(Long bno) {
20         // TODO Auto-generated method stub
21         return null;
22     }
23
24     @Override
25     public boolean modify(BoardVO board) {
26         // TODO Auto-generated method stub
27         return false;
28     }
29
30     @Override
31     public boolean remove(Long bno) {
32         // TODO Auto-generated method stub
33         return false;
34     }
35
36     @Override
37     public List<BoardVO> getList() {
38         // TODO Auto-generated method stub
39         return null;
40     }
41
42 }

```

필요한 클래스 import

```

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.freeflux.domain.BoardVO;
import com.freeflux.mapper.BoardMapper;

import lombok.AllArgsConstructor;
import lombok.Setter;
import lombok.extern.log4j.Log4j;

```


클래스 선언부에 어노테이션 추가

```
@Log4j
@Service
@AllArgsConstructor
public class BoardServiceImpl implements BoardService {
```

만약 @Log4j 부분에 에러 발생 할 경우,
pom.xml 에서 log4j 부분의 버전을 1.2.17 로 변경.
그 외에 불필요한 부분은 제거

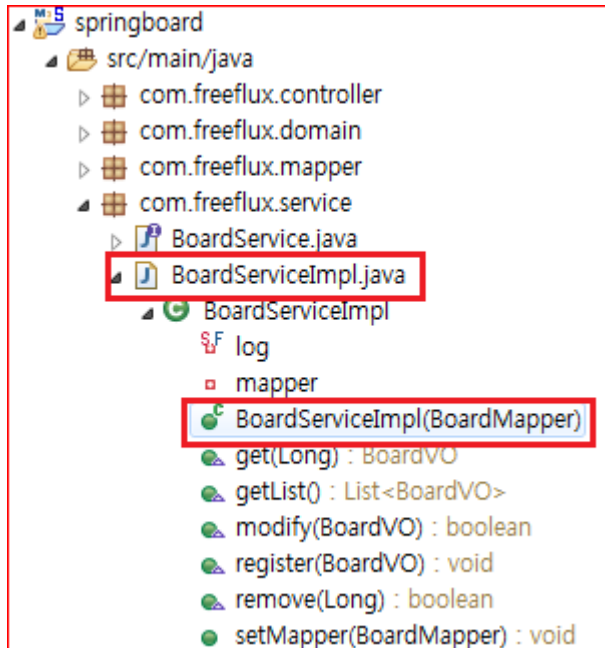
```
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-log4j12</artifactId>
  <version>${org.slf4j-version}</version>
  <scope>runtime</scope>
</dependency>
<dependency>
  <groupId>log4j</groupId>
  <artifactId>log4j</artifactId>
  <version>1.2.17</version>
  <!--
  <exclusions>
    <exclusion>
      <groupId>javax.mail</groupId>
      <artifactId>mail</artifactId>
    </exclusion>
    <exclusion>
      <groupId>javax.jms</groupId>
      <artifactId>jms</artifactId>
    </exclusion>
    <exclusion>
      <groupId>com.sun.jdmk</groupId>
      <artifactId>jmxtools</artifactId>
    </exclusion>
    <exclusion>
      <groupId>com.sun.jmx</groupId>
      <artifactId>jmxri</artifactId>
    </exclusion>
  </exclusions>
  <scope>runtime</scope>
  -->
</dependency>

<!-- @Inject -->
<dependency>
  <groupId>javax.inject</groupId>
  <artifactId>javax.inject</artifactId>
  <version>1</version>
</dependency>
```

스프링 4.3 부터는 단일 파라미터를 받는 생성자의 경우, 필요한 파라미터를 자동으로 주입 가능.

@AllArgsConstructor : 모든 파라미터를 이용하는 생성자를 생성.

프로젝트 구조에서 확인해보면 스프링 4.3 의 자동주입 기능에 의해서 아래와 같은 형태가 된다.



BoardMapper 객체를 위한 멤버(인스턴스) 변수 선언

```
@Setter(onMethod_ = @Autowired)
private BoardMapper mapper;
```

그 외의 **@Override** 메서드 내부 수정

```
@Override
public void register(BoardVO board) {
    log.info("register....." + board);
    mapper.insertSelectKey(board);
}

@Override
public BoardVO get(Long bno) {
    log.info("get....." + bno);
    return mapper.read(bno);
}

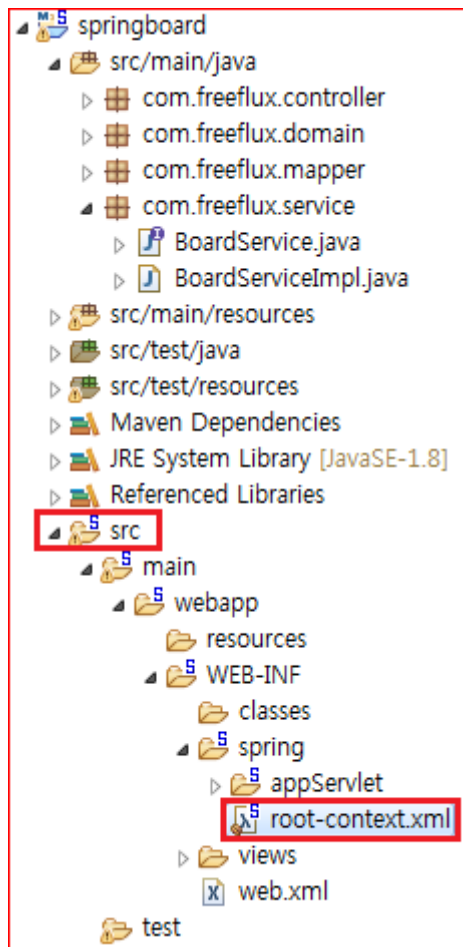
@Override
public boolean modify(BoardVO board) {
    log.info("modify....." + board);
    return mapper.update(board) == 1;
}
```

```
@Override
public boolean remove(Long bno) {
    log.info("remove...." + bno);
    return mapper.delete(bno) == 1;
}

@Override
public List<BoardVO> getList() {
    log.info("getList.....");
    return mapper.getList();
}
```

5. 스프링의 서비스 객체 설정 (root-context.xml)

스프링의 빈으로 주입하기 위하여 root-context.xml 에
@Service 어노테이션이 있는 com.freeflux.service 패키지를 스캔(조사)하도록 추가.



```
<mybatis-spring:scan base-package="com.freeflux.mapper" />  
  
<context:component-scan base-package="com.freeflux.service"></context:component-scan>  
</beans>
```

6. 비즈니스 계층의 구현과 테스트 (BoardServiceTests.java)

New Java Class

Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

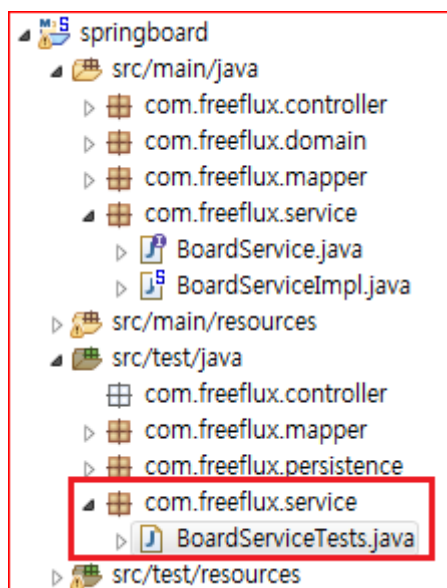
☐ public static void main(String[] args)

☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments



필요한 클래스 import

```
import static org.junit.Assert.assertNotNull;  
  
import org.junit.Test;  
import org.junit.runner.RunWith;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.test.context.ContextConfiguration;  
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;  
  
import lombok.Setter;  
import lombok.extern.log4j.Log4j;
```

테스트를 위한 클래스 선언부에 어노테이션 추가

```
@RunWith(SpringJUnit4ClassRunner.class)  
@ContextConfiguration("file:src/main/webapp/WEB-INF/spring/root-context.xml")  
@Log4j  
public class BoardServiceTests {
```

BoardService 객체가 제대로 주입이 가능한지 확인하기 위한 멤버(인스턴스)변수 및 메서드 추가

```
@RunWith(SpringJUnit4ClassRunner.class)  
@ContextConfiguration("file:src/main/webapp/WEB-INF/spring/root-context.xml")  
@Log4j  
public class BoardServiceTests {  
  
    @Setter(onMethod_ = { @Autowired })  
    private BoardService service;  
  
    @Test  
    public void testExist() {  
  
        log.info(service);  
        assertNotNull(service);  
    }  
}
```

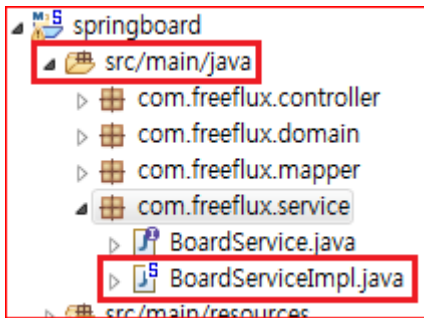
테스트

정상적으로 BoardService 객체가 생성되고, BoardMapper 가 주입되었다면
아래와 같이 BoardService 객체와 데이터베이스 관련 로그가 출력된다.

```
INFO : com.freeflux.service.BoardServiceTests - com.freeflux.service.BoardServiceImpl@55dfebeb  
INFO : org.springframework.context.support.GenericApplicationContext - Closing org.springframework...  
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
```

7. 등록 작업의 구현과 테스트

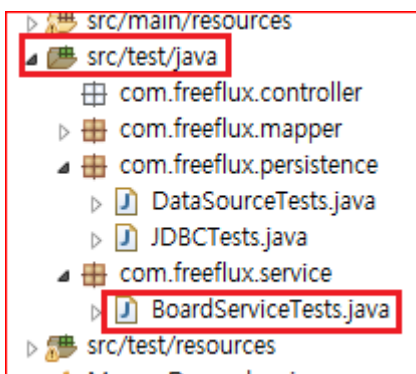
등록작업은 BoardServiceImpl 에서
파라미터로 전달되는 BoardVO 타입의 객체를 BoardMapper 를 통해서 처리.



```
@Override
public void register(BoardVO board) {
    log.info("register....." + board);
    mapper.insertSelectKey(board);
}
```

mapper.insertSelectKey()의 반환 값인 int 를 사용하고 있지는 않지만,
필요에 의해서 예외처리나 void 대신 int 타입을 이용할 수 있다.

BoardServiceTests 에 추가



```
@Test
public void testRegister() {

    BoardVO board = new BoardVO();
    board.setTitle("새로 작성하는 글 테스트");
    board.setContent("새로 작성하는 내용 테스트");
    board.setWriter("newbie test");

    service.register(board);

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

```

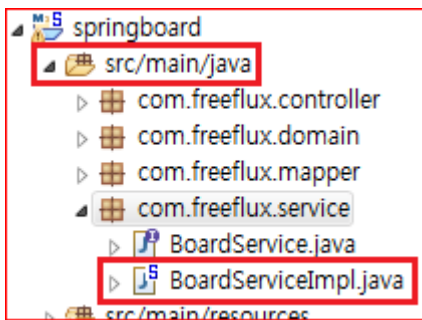
INFO : jdbc.audit - 1. PreparedStatement.new PreparedStatement returned
INFO : jdbc.audit - 1. Connection.prepareStatement(insert into tbl_board (bno,title,content,
        values (?, ?, ?, ?)) returned net.sf.log4jdbc.sqljdbcapi.PreparedStatementSpy@
INFO : jdbc.audit - 1. PreparedStatement.setLong(1, 42) returned
INFO : jdbc.audit - 1. PreparedStatement.setString(2, "새로 작성하는 글 테스트") returned
INFO : jdbc.audit - 1. PreparedStatement.setString(3, "새로 작성하는 내용 테스트") returned
INFO : jdbc.audit - 1. PreparedStatement.setString(4, "newbie test") returned
INFO : jdbc.sqlonly - insert into tbl_board (bno,title,content, writer) values (42, '새로 작성하
테스트', 'newbie test')

INFO : jdbc.sqltiming - insert into tbl_board (bno,title,content, writer) values (42, '새로 작성
테스트', 'newbie test')
{executed in 0 msec}
INFO : jdbc.audit - 1. PreparedStatement.execute() returned false
INFO : jdbc.audit - 1. PreparedStatement.getUpdateCount() returned 1
INFO : jdbc.audit - 1. PreparedStatement.isClosed() returned false
INFO : jdbc.audit - 1. PreparedStatement.close() returned
INFO : jdbc.audit - 1. Connection.clearWarnings() returned
INFO : com.freeflux.service.BoardServiceTests - 생성된 게시물의 번호: 42
INFO : org.springframework.context.support.GenericApplicationContext - Closing org.spring
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...

```

8. 리스트(목록) 작업의 구현과 테스트

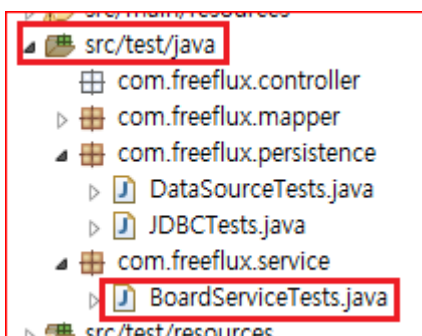
BoardServiceImpl 클래스에서
현재 테이블에 저장된 모든 데이터를 가져오는 getList() 에서 처리.



```

@Override
public List<BoardVO> getList() {
    log.info("getList.....");
    return mapper.getList();
}

```



@Test

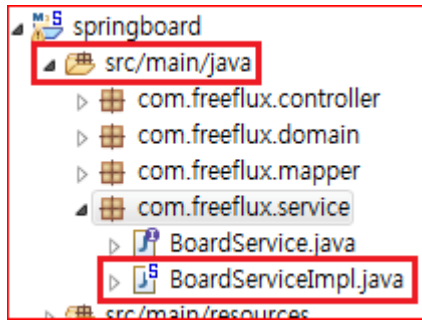
```
public void testGetList() {  
    service.getList().forEach(board -> log.info(board));  
}
```

```
INFO : jdbc.resultset - 1. ResultSet.getLong(BNO) returned 42  
INFO : jdbc.resultset - 1. ResultSet.isNull() returned false  
INFO : jdbc.resultset - 1. ResultSet.getString(TITLE) returned 새로 작성하는 글 테스트  
INFO : jdbc.resultset - 1. ResultSet.isNull() returned false  
INFO : jdbc.resultset - 1. ResultSet.getString(CONTENT) returned 새로 작성하는 내용  
INFO : jdbc.resultset - 1. ResultSet.isNull() returned false  
INFO : jdbc.resultset - 1. ResultSet.getString(WRITER) returned newbie test  
INFO : jdbc.resultset - 1. ResultSet.isNull() returned false  
INFO : jdbc.resultset - 1. ResultSet.getTimestamp(REGDATE) returned 2019-09-11 13:07:00  
INFO : jdbc.resultset - 1. ResultSet.isNull() returned false  
INFO : jdbc.resultset - 1. ResultSet.getTimestamp(UPDATEDATE) returned 2019-09-11 13:07:00  
INFO : jdbc.resultset - 1. ResultSet.isNull() returned false  
INFO : jdbc.resultsettable -  
|-----|-----|-----|-----|-----|  
| bno | title | content | writer | regdate | updated |  
|-----|-----|-----|-----|-----|  
| 1 | 제목 수정합니다. | 테스트 내용 | user00 | 2019-09-09 18:07:00 | 2019-09-09 18:07:00 |  
| 4 | 테스트제목3 | 테스트 내용3 | user03 | 2019-09-09 18:07:00 | 2019-09-09 18:07:00 |  
| 5 | 수정된 제목 | 수정된 내용 | user00 | 2019-09-09 18:07:00 | 2019-09-09 18:07:00 |  
| 21 | 새로 작성하는 글 select key | 새로 작성하는 내용 select key | newbie | 2019-09-11 13:07:00 | 2019-09-11 13:07:00 |  
| 22 | 새로 작성하는 글 | 새로 작성하는 내용 | newbie | 2019-09-11 13:07:00 | 2019-09-11 13:07:00 |  
| 23 | 새로 작성하는 글 | 새로 작성하는 내용 | newbie | 2019-09-11 13:07:00 | 2019-09-11 13:07:00 |  
| 24 | 새로 작성하는 글 select key | 새로 작성하는 내용 select key | newbie | 2019-09-11 13:07:00 | 2019-09-11 13:07:00 |  
| 25 | 새로 작성하는 글 | 새로 작성하는 내용 | newbie | 2019-09-11 13:07:00 | 2019-09-11 13:07:00 |  
| 26 | 새로 작성하는 글 select key | 새로 작성하는 내용 select key | newbie | 2019-09-11 13:07:00 | 2019-09-11 13:07:00 |  
| 41 | 새로 작성하는 글 | 새로 작성하는 내용 | newbie | 2019-09-11 13:07:00 | 2019-09-11 13:07:00 |  
| 42 | 새로 작성하는 글 테스트 | 새로 작성하는 내용 테스트 | newbie test | 2019-09-11 13:07:00 | 2019-09-11 13:07:00 |  
|-----|-----|-----|-----|-----|
```

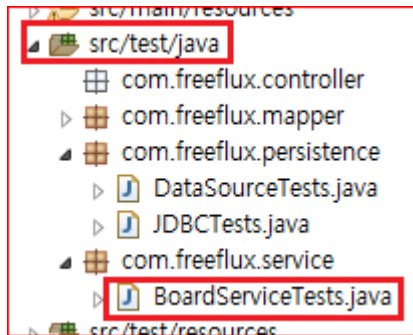
```
INFO : jdbc.resultset - 1. ResultSet.next() returned false  
INFO : jdbc.resultset - 1. ResultSet.close() returned void  
INFO : jdbc.audit - 1. Connection.getMetaData() returned oracle.jdbc.driver.OracleDriverMetaData  
INFO : jdbc.audit - 1. PreparedStatement.isClosed() returned false  
INFO : jdbc.audit - 1. PreparedStatement.close() returned  
INFO : jdbc.audit - 1. Connection.clearWarnings() returned  
INFO : com.freeflux.service.BoardServiceTests - BoardVO(bno=1, title=제목 수정합니다, content=테스트 내용, writer=user00, regdate=2019-09-09 18:07:00, updated=2019-09-09 18:07:00)  
INFO : com.freeflux.service.BoardServiceTests - BoardVO(bno=4, title=테스트제목3, content=테스트 내용3, writer=user03, regdate=2019-09-09 18:07:00, updated=2019-09-09 18:07:00)  
INFO : com.freeflux.service.BoardServiceTests - BoardVO(bno=5, title=수정된 제목, content=수정된 내용, writer=user00, regdate=2019-09-09 18:07:00, updated=2019-09-09 18:07:00)  
INFO : com.freeflux.service.BoardServiceTests - BoardVO(bno=21, title=새로 작성하는 글 select key, content=새로 작성하는 내용 select key, writer=newbie, regdate=2019-09-11 13:07:00, updated=2019-09-11 13:07:00)  
INFO : com.freeflux.service.BoardServiceTests - BoardVO(bno=22, title=새로 작성하는 글, content=새로 작성하는 내용, writer=newbie, regdate=2019-09-11 13:07:00, updated=2019-09-11 13:07:00)  
INFO : com.freeflux.service.BoardServiceTests - BoardVO(bno=23, title=새로 작성하는 글, content=새로 작성하는 내용, writer=newbie, regdate=2019-09-11 13:07:00, updated=2019-09-11 13:07:00)  
INFO : com.freeflux.service.BoardServiceTests - BoardVO(bno=24, title=새로 작성하는 글 select key, content=새로 작성하는 내용 select key, writer=newbie, regdate=2019-09-11 13:07:00, updated=2019-09-11 13:07:00)  
INFO : com.freeflux.service.BoardServiceTests - BoardVO(bno=25, title=새로 작성하는 글, content=새로 작성하는 내용, writer=newbie, regdate=2019-09-11 13:07:00, updated=2019-09-11 13:07:00)  
INFO : com.freeflux.service.BoardServiceTests - BoardVO(bno=26, title=새로 작성하는 글 select key, content=새로 작성하는 내용 select key, writer=newbie, regdate=2019-09-11 13:07:00, updated=2019-09-11 13:07:00)  
INFO : com.freeflux.service.BoardServiceTests - BoardVO(bno=41, title=새로 작성하는 글, content=새로 작성하는 내용, writer=newbie, regdate=2019-09-11 13:07:00, updated=2019-09-11 13:07:00)  
INFO : com.freeflux.service.BoardServiceTests - BoardVO(bno=42, title=새로 작성하는 글 테스트, content=새로 작성하는 내용 테스트, writer=newbie test, regdate=2019-09-11 13:07:00, updated=2019-09-11 13:07:00)  
INFO : org.springframework.context.support.GenericApplicationContext - Closing org.springframework.context.support.GenericApplicationContext@7b0b1c1b: shutdown complete  
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
```


9. 조회 작업의 구현과 테스트

조회는 게시물 번호가 파라미터이고, BoardVO 의 인스턴스가 반환된다.



```
@Override
public BoardVO get(Long bno) {
    log.info("get....." + bno);
    return mapper.read(bno);
}
```



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

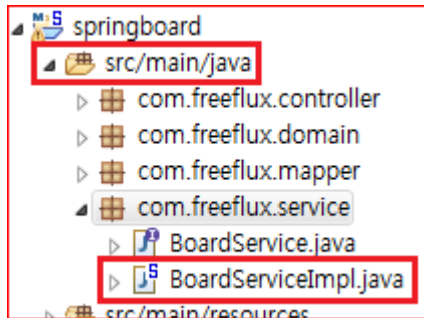
```
INFO : jdbc.resultsettable -
|---|-----|-----|-----|-----|-----|
|bno|title   |content|writer|regdate          |updatedate          |
|---|-----|-----|-----|-----|-----|
|1  |제목 수정합니다. |테스트 내용 |user00 |2019-09-09 18:05:44.0 |2019-09-11 13:00:00.0 |
|---|-----|-----|-----|-----|-----|

INFO : jdbc.resultset - 1. ResultSet.next() returned false
INFO : jdbc.resultset - 1. ResultSet.close() returned void
INFO : jdbc.audit - 1. Connection.getMetaData() returned oracle.jdbc.driver.Oracle
INFO : jdbc.audit - 1. PreparedStatement.isClosed() returned false
INFO : jdbc.audit - 1. PreparedStatement.close() returned
INFO : jdbc.audit - 1. Connection.clearWarnings() returned
INFO : com.freeflux.service.BoardServiceTests - BoardVO(bno=1, title=제목 수정합
INFO : org.springframework.context.support.GenericApplicationContext - Closing
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
```

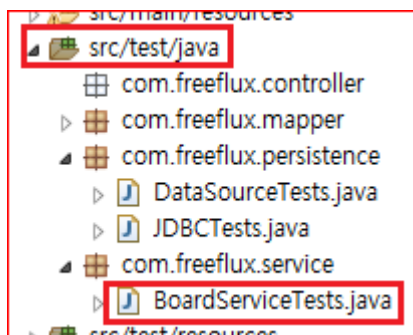
10. 삭제 구현과 테스트

삭제 메서드의 리턴 타입을 void로 설계할 수도 있지만, 엄격하게 처리하기 위해서 Boolean 타입으로 처리.

정상적으로 삭제가 이루어지면 '1' 값이 반환되기 때문에 '==' 연산자를 이용하여 true/false를 처리할 수 있다.



```
@Override
public boolean remove(Long bno) {
    log.info("remove...." + bno);
    return mapper.delete(bno) == 1;
}
```



```
@Test
public void testDelete() {

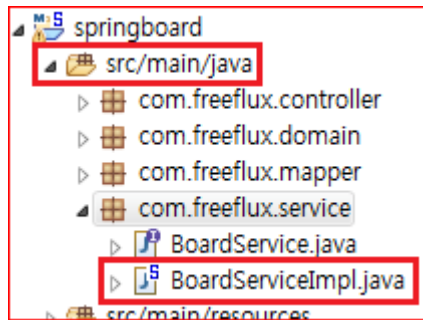
    // 게시물 번호의 존재 여부를 확인하고 테스트할 것
    log.info("REMOVE RESULT: " + service.remove(7L));
}
```

```
INFO : jdbc.sqltiming - delete from tbl_board where bno = 7
      {executed in 3 msec}
INFO : jdbc.audit - 1. PreparedStatement.execute() returned false
INFO : jdbc.audit - 1. PreparedStatement.getUpdateCount() returned 0
INFO : jdbc.audit - 1. PreparedStatement.isClosed() returned false
INFO : jdbc.audit - 1. PreparedStatement.close() returned
INFO : jdbc.audit - 1. Connection.clearWarnings() returned
INFO : com.freeflux.service.BoardServiceTests - REMOVE RESULT: false
INFO : org.springframework.context.support.GenericApplicationContext - Closin
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
```

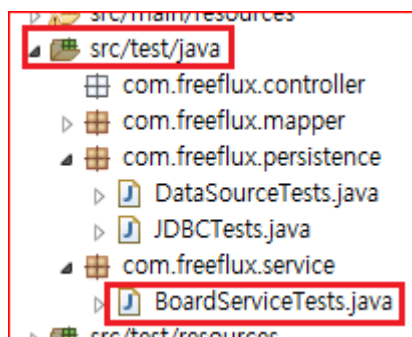
11. 수정 구현과 테스트

수정 메서드 역시 리턴 타입을 void 로 설계할 수도 있지만, 엄격하게 처리하기 위해서 Boolean 타입으로 처리.

정상적으로 수정이 이루어지면 '1' 값이 반환되기 때문에 '==' 연산자를 이용하여 true/false 를 처리할 수 있다.



```
@Override
public boolean modify(BoardVO board) {
    log.info("modify....." + board);
    return mapper.update(board) == 1;
}
```



```
@Test
public void testUpdate() {

    BoardVO board = service.get(1L);

    if (board == null) {
        return;
    }

    board.setTitle("제목을 수정 테스트합니다.");
    log.info("MODIFY RESULT: " + service.modify(board));
}
```

```
INFO : jdbc.audit - 1. PreparedStatement.new PreparedStatement returned
INFO : jdbc.audit - 1. Connection.prepareStatement(update tbl_board
      set title = ?, content = ?, writer = ?, updateDate = sysdate
      where bno = ?) returned net.sf.log4jdbc.sql.jdbcapi.PreparedStatement
INFO : jdbc.audit - 1. PreparedStatement.setString(1, "제목을 수정 테스트합니다") returned
INFO : jdbc.audit - 1. PreparedStatement.setString(2, "테스트 내용") returned
INFO : jdbc.audit - 1. PreparedStatement.setString(3, "user00") returned
INFO : jdbc.audit - 1. PreparedStatement.setLong(4, 1) returned
INFO : jdbc.sqlonly - update tbl_board set title = '제목을 수정 테스트합니다.', c
= sysdate where bno = 1

INFO : jdbc.sqltiming - update tbl_board set title = '제목을 수정 테스트합니다.',
= sysdate where bno = 1
{executed in 1 msec}
INFO : jdbc.audit - 1. PreparedStatement.execute() returned false
INFO : jdbc.audit - 1. PreparedStatement.getUpdateCount() returned 1
INFO : jdbc.audit - 1. PreparedStatement.isClosed() returned false
INFO : jdbc.audit - 1. PreparedStatement.close() returned
INFO : jdbc.audit - 1. Connection.clearWarnings() returned
INFO : com.freeflux.service.BoardServiceTests - MODIFY RESULT: true
INFO : org.springframework.context.support.GenericApplicationContext - Closin
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
```