비지니스 계층 (Service)

생성일	@2022년 8월 23일 오후 5:29
태그	

비지니스 계층

로직을 기준으로 설계하며, 프리젠테이션 계층과 영속 계층의 중간 다리역할을 한다.

1. Service 인터페이스 작성

각 계층 간의 연결은 인터페이스를 이용하여, 느슨한(loose) 연결을 하도록 한다.

```
import java.util.List;

import org.zerock.myapp.domain.BoardDTO;
import org.zerock.myapp.domain.BoardVO;
import org.zerock.myapp.exception.ServiceException;

public interface BoardService {

  // -- 1. 게시글 전체목록 획득
  public abstract List<BoardVO> getList() throws ServiceException;

  // -- 2. 새로운 게시글 등록
  public abstract boolean register(BoardDTO dto) throws ServiceException;

  // -- 3. 기존 게시글 수정
  public abstract boolean modify(BoardDTO dto) throws ServiceException;

  // -- 4. 기존 게시글 삭제
  public abstract boolean remove(BoardDTO dto) throws ServiceException;

  // -- 5. 기존 게시글 상세조회
  public abstract BoardVO get(BoardDTO dto) throws ServiceException;

} // interface
```

2. Service 구현클래스 작성

어노테이션 @Service를 이용하여, Bean으로 등록한다.

Service클래스에서는 영속성 계층을 통해, DB 접근이 필요하므로,

영속성 계층을 주입받는다.

```
@Log4j2
@NoArgsConstructor

@Service
public class BoardServiceImpl implements BoardService {

  // -- 비니지스 계층에서는 영속성 계층을 통해, DB에 접근한다.
  // -- 그러기 위해, 영속성 계층의 Bean을 주입받는다.
  @Setter(onMethod_= {@Autowired})
  private BoardMapper mapper;

  @Override
  public List<BoardVO> getList() throws ServiceException {
    log.trace("getList() invoked.");

    try {
      return this.mapper.selectAllList();
    } catch (Exception e) {
      throw new ServiceException(e);
    }

  } // getList

  @Override
  public boolean register(BoardDTO dto) throws ServiceException {

    try {
      return this.mapper.insert(dto) == 1;
    } catch (Exception e) {
      throw new ServiceException(e);
    }

  } // register

  @Override
  public boolean modify(BoardDTO dto) throws ServiceException {

    try {
      return this.mapper.update(dto) == 1;
    } catch (Exception e) {
      throw new ServiceException(e);
    }
```

```
  } // modify

  @Override
  public boolean remove(BoardDTO dto) throws ServiceException {

    try {
      return this.mapper.delete(dto.getBno()) == 1;
    } catch (Exception e) {
      throw new ServiceException(e);
    }

  } // remove

  @Override
  public BoardVO get(BoardDTO dto) throws ServiceException {

    try {
      return this.mapper.select(dto);
    } catch (Exception e) {
      throw new ServiceException(e);
    }

  } // get

} // end class
```

3. JUnit5 TEST

비지니스 계층인 Service가 제대로 작동하는지 JUnit기반 TEST를 실행한다.

Setter 메소드 의존성 주입이 아닌, 생성자 주입으로 TEST 하였다.

-. 생성자 주입으로 할 경우, BoardService객체에 final 키워드를 입력하여 immutable 하게 만들 수 있다는 장점이 있다.

```
@Log4j2

@ExtendWith(SpringExtension.class)
@ContextConfiguration(locations = {
    "file:src/main/webapp/WEB-INF/**/*-context.xml",
})

@TestInstance(Lifecycle.PER_CLASS)
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
public class BoardServiceTests {

//  @Setter(onMethod_= {@Autowired})
```

```
  private final BoardService service;

  @Autowired
  public BoardServiceTests(BoardService service) {
    this.service = service;
  }

  @BeforeAll
  void beforeAll() {
    log.trace("beforeAll() invoked.");

    assertNotNull(this.service);

    log.info("\t+ this.service : {}", this.service);
  } // beforeAll


//  @Disabled
  @Test
  @Order(1)
  @DisplayName("1. BoardService.getList()")
  void testGetList() throws ServiceException {
    log.trace("testGetList() invoked.");

    List<BoardVO> list = new ArrayList<>();

    list = service.getList();

    list.forEach(log::info);

  } // testGetList


//  @Disabled
  @Test
  @Order(2)
  @DisplayName("2. BoardService.register()")
  @Timeout(value=3, unit=TimeUnit.SECONDS)
  void testRegister() throws ServiceException {
    log.trace("testRegister() invoked.");

    BoardDTO dto = new BoardDTO();

    dto.setTitle("TITLE_NEW");
    dto.setContent("CONTENT_NEW");
    dto.setWriter("WRITER_NEW");

    log.info("\t+ result : {}", this.service.register(dto));

  } // testRegister

//  @Disabled
  @Test
  @Order(3)
  @DisplayName("3. BoardService.modify()")
  @Timeout(value=3, unit=TimeUnit.SECONDS)
  void testModify() throws ServiceException {
    log.trace("testModify() invoked.");
```

```
    BoardDTO dto = new BoardDTO();

    dto.setBno(30);
    dto.setTitle("UPDATE_NEW");
    dto.setContent("UPDATE_NEW");
    dto.setWriter("UPDATE_NEW");

    log.info("\t+ result : {}", this.service.modify(dto));

  } // testModify

//  @Disabled
  @Test
  @Order(4)
  @DisplayName("4. BoardService.remove()")
  void testRemove() throws ServiceException {
    log.trace("testRemove() invoked.");

    BoardDTO dto = new BoardDTO();
    dto.setBno(132);

    log.info("\t+ result : {}", this.service.remove(dto));

  } // testRemove


//  @Disabled
  @Test
  @Order(5)
  @DisplayName("5. BoardService.get()")
  @Timeout(value=3, unit=TimeUnit.SECONDS)
  void testGet() throws ServiceException {
    log.trace("testGet() invoked.");

    BoardDTO dto = new BoardDTO();

    dto.setBno(50);

    log.info("\t+ result : {}", this.service.get(dto));

  } // testGet

} // end class
```