2018 JANUARY 17

코드로 배우는 스프링 웹 프로젝트

PART 2 기본적인 기능의 게시물 관리

SEOKRAE KIM

내용

I. 코드로 배우는 스프링 웹 프로젝트

1. 기본적인 기능의 게시물 관리

- 프로젝트의 전체 구조가 갖추어 놓았고 본격적인 구현에 들어가도록 한다.
- 구현작업은 등록 기능을 먼저 구현하는데 그 중 영속 계층에 대한 처리부터 한다.
- 데이터베이스와 관련된 작업을 먼저 하는 이유는 개발 간에 변경되는 화면의 설계를 미룰 수 있기 때문이다.

1) 영속(persistence) 계층, 비즈니스 계층

개발 순서
● 테이블과 관련된 BoardVO 클래스를 설계하고 작성 ● MyBatis를 이용하는 BoardDAO, BoardDAOImpl을 작성 ● Resources 폴더 내에 XML Mapper를 작성 ● BoardMapperTests.java 작성 및 테스트

(1) BoardVO의 작성

- DB 테이블의 구조를 객체화하는 BoardVO 클래스를 작성하는 것
- DB 테이블의 컬럼 명과 멤버변수의 이름을 같게 해주도록 한다.

domain/BoardVO

```
public class BoardVO {

    private Integer bno;
    private String title;
    private String content;
    private String writer;
    private Date regdate;
    private int viewcnt;
```

(2) DAO의 생성과 XML Mapper 작업

① XML 네임스페이스의 추가

root-context.xml 파일을 선택하여 NameSpaces를 선택

② SessionFactory, SqlSessionTemplate 추가

- **MyBatis의 SqlSessionFactory, SqlSessionTemplate 등록**

root-context.xml 수정

```
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="configLocation" value="classpath:/mybatis-config.xml"></property>
        <property name="mapperLocations" value="classpath:mappers/**/*Mapper.xml"></property>
</bean>

<bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate" destroy-method="clearCache">
        <constructor-arg name="sqlSessionFactory" ref="sqlSessionFactory"></constructor-arg>
</bean>

<context:component-scan base-package="org.zerock.persistence"></context:component-scan>
```

③ BoardDAO 생성

persistence.BoardDAO

```
public interface BoardDAO {

        public void create(BoardVO vo) throws Exception;

        public BoardVO read(Integer bno) throws Exception;
```

```
        public void update(BoardVO vo) throws Exception;

        public void delete(Integer bno) throws Exception;

        public List<BoardVO> listAll() throws Exception;
}
```

④ XML Mapper SQL작성

```
<mapper namespace="org.zerock.mapper.BoardMapper">

        <insert id="create">
        insert into tbl_board (title, content, writer)
        values(#{title},#{content}, #{writer})
        </insert>

        <select id="read" resultType="org.zerock.domain.BoardVO">
        select
          bno, title, content, writer, regdate, viewcnt
        from
          tbl_board
        where bno = #{bno}
        </select>

        <update id="update">
        update tbl_board set title =#{title}, content =#{content}
        where bno = #{bno}
        </update>

        <delete id="delete">
        delete from tbl_board where bno = #{bno}
        </delete>

        <select id="listAll" resultType="org.zerock.domain.BoardVO">
        <![CDATA[
        select
```

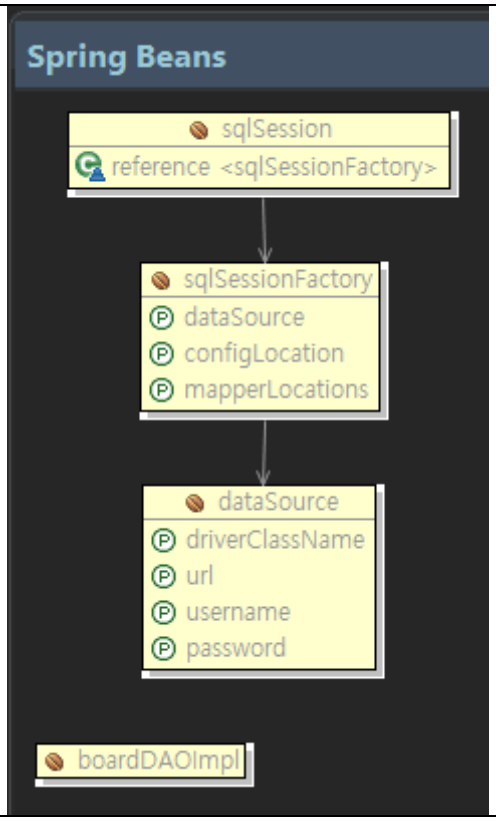
```
        bno, title, content, writer, regdate, viewcnt
    from
        tbl_board
    where bno > 0
    order by bno desc, regdate desc
    ]]>
    </select>
</mapper>
```

⑤ BoardDAO의 구현클래스 BoardDAOImpl

```
@Repository
public class BoardDAOImpl implements BoardDAO {

    @Inject
    private SqlSession session;

    private static String namespace = "org.zerock.mapper.BoardMapper";

    @Override
    public void create(BoardVO vo) throws Exception {
        session.insert(namespace + ".create", vo);
    }

    @Override
    public BoardVO read(Integer bno) throws Exception {
        return session.selectOne(namespace + ".read", bno);
    }

    @Override
    public void update(BoardVO vo) throws Exception {
        session.update(namespace + ".update", vo);
    }

    @Override
    public void delete(Integer bno) throws Exception {
```

```
                session.delete(namespace + ".delete", bno);
        }

        @Override
        public List<BoardVO> listAll() throws Exception {
                return session.selectList(namespace + ".listAll");
        }
}
```

⑥ 스프링 빈 등록 확인

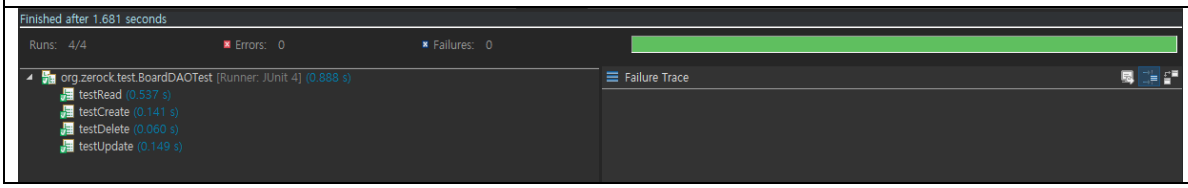


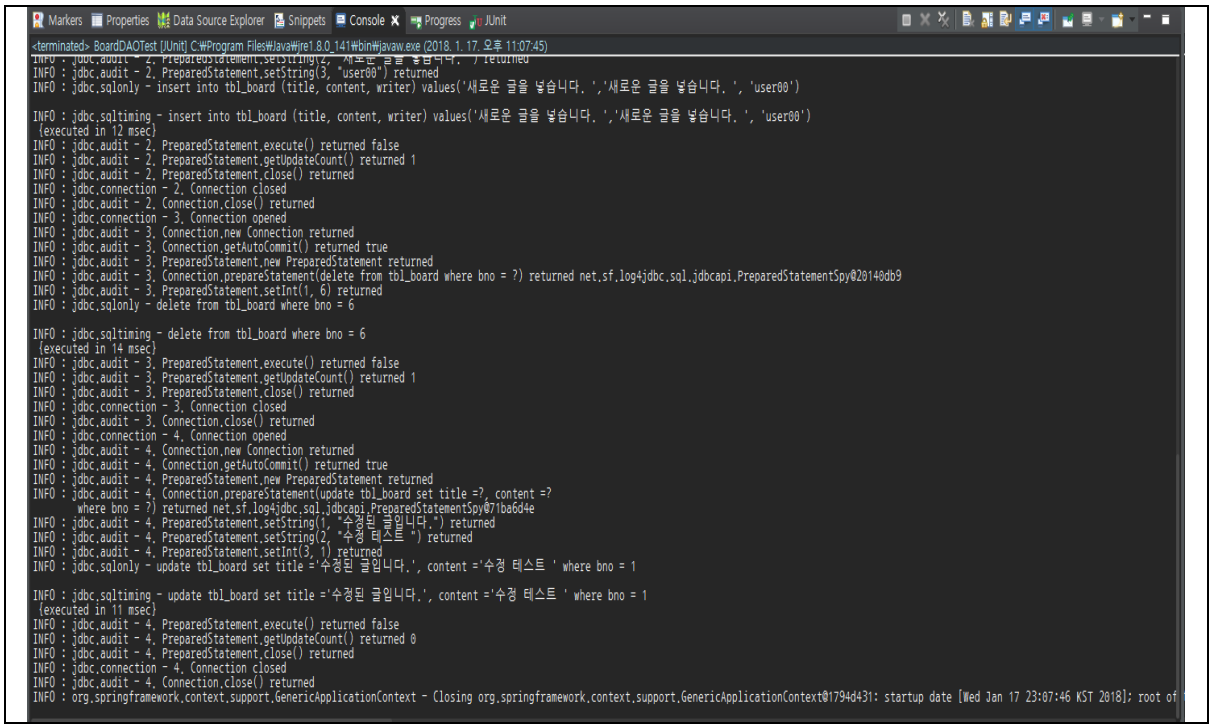
⑦ BoardDAO 테스트

jUnit을 통해 테스트 하기 위한 코드 작성

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = { "file:src/main/webapp/WEB-INF/spring/**/*.xml" })
public class BoardDAOTest {
```

```
    @Inject
    private BoardDAO dao;

    private static Logger logger = LoggerFactory.getLogger(BoardDAOTest.class);

    @Test
    public void testCreate() throws Exception {
        BoardVO board = new BoardVO();
        board.setTitle("새로운 글을 넣습니다. ");
        board.setContent("새로운 글을 넣습니다. ");
        board.setWriter("user00");
        dao.create(board);
    }

    @Test
    public void testRead() throws Exception {
        logger.info(dao.read(1).toString());
    }

    @Test
    public void testUpdate() throws Exception {
        BoardVO board = new BoardVO();
        board.setBno(1);
        board.setTitle("수정된 글입니다.");
        board.setContent("수정 테스트 ");
        dao.update(board);
    }

    @Test
    public void testDelete() throws Exception {
        dao.delete(1);
    }
}
```





⑧ <typeAliases> 적용

<typeAliases> 　　<package name="org.zerock.domain"/> </typeAliases>
<select id="read" resultType="BoardVO"> <select id="listAll" resultType="BoardVO">

(3) 계층별 구현 – 비즈니스 계층