Part 2

Chapter 02 영속(persistence), 비즈니스 계층

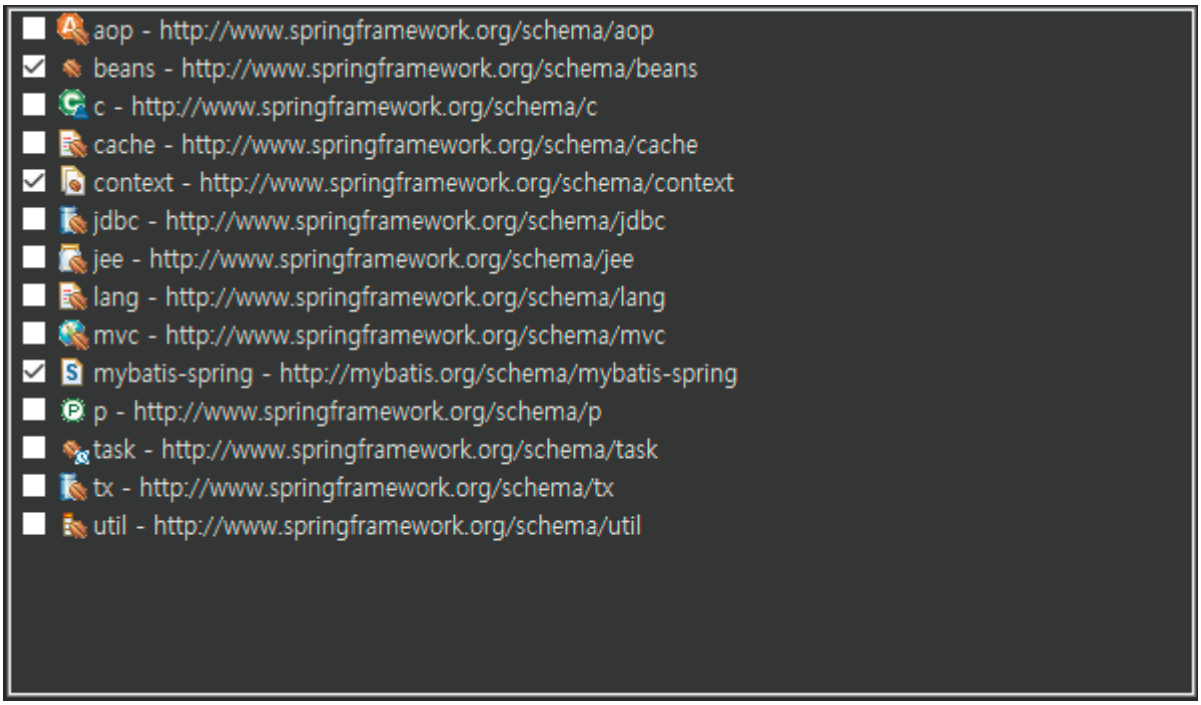
2.1. BoardVO 작성

```
public class BoardVO {

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

    public Integer getBno() {
        return bno;
    }
    public void setBno(Integer bno) {
        this.bno = bno;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }
    public String getWriter() {
        return writer;
    }
    public void setWriter(String writer) {
        this.writer = writer;
    }
    public Date getRegdate() {
        return regdate;
    }
    public void setRegdate(Date regdate) {
        this.regdate = regdate;
    }
    public int getViewcnt() {
        return viewcnt;
    }
    }
```

```
    public void setViewcnt(int viewcnt) {
        this.viewcnt = viewcnt;
    }
    @Override
    public String toString() {
        return "BoardVO [bno=" + bno + ", title=" + title + ", content="
                + content + ", writer=" + writer + ", regdate=" + regdate
                + ", viewcnt=" + viewcnt + "]";
    }
}
```

2.2. DAO의 생성과 XML Mapper 작업

2.2.1. XML 네임스페이스의 추가



2.2.2. SessionFactory, SqlSessionTemplate의 추가

```
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value ="net.sf.log4jdbc.sql.jdbcapi.DriverSpy"></property>
    <property name="url" value="jdbc:log4jdbc:mysql://{ip}:3306/{dbName}?useSSL=false">
</property>
    <property name="username" value="{id}"></property>
    <property name="password" value="{pwd}"></property>
</bean>
<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>
```

2.2.3. BoardDAO의 생성

```
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;
}
```

2.2.4. XML MApper에서의 SQL 처리

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<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="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="BoardVO">
    <![CDATA[
    select
      bno, title, content, writer, regdate, viewcnt
    from
      tbl_board
    where bno > 0
    order by bno desc, regdate desc
    ]]>
    </select>
</mapper>
```

2.2.5. BoardDAO의 구현 클래스 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");
    }
}
```

2.2.6. BoardDAO의 테스트

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

- MySQL Query Auto_increment 수정

```
delete from tbl_board;
alter table tbl_board auto_increment = 0;
```

2.2.7. 의 적용

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <typeAliases>
        <package name="org.zerock.domain"/>
    </typeAliases>
</configuration>
```

2.3. 계층별 구현 - 비즈니스 계층

- 요구사항을 메소드로 정리하여 인터페이스를 정의
- 인터페이스 구현 객체 생성

2.3.1. 비즈니스 계층의 구현

2.3.1.1. root-context.xml의 설정

- 패키지 자동 인식

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

2.3.1.2. BoardService/BoardServiceImpl의 작성

```
@Service
public class BoardServiceImpl implements BoardService {

    @Inject
    private BoardDAO dao;

    @Override
    public void regist(BoardVO board) throws Exception {
        dao.create(board);
    }

    @Override
    public BoardVO read(Integer bno) throws Exception {
        return dao.read(bno);
    }

    @Override
    public void modify(BoardVO board) throws Exception {
        dao.update(board);
    }

    @Override
    public void remove(Integer bno) throws Exception {
        dao.delete(bno);

    }
```

```
    @Override
    public List<BoardVO> listAll() throws Exception {
        return dao.listAll();
    }
}
```

2.3.2. 비즈니스 계층의 테스트

- 현재는 중요한 작업이 없기 때문에 테스트 코드를 작성하지 않으나 트랜잭션이나 AOP 기능 테스트를 위해서는 미리 테스트 코드를 가지고 있는 것이 좋다.