1. 개발 패키지 구성

org.zerock.controller : 스프링 컨트롤러 클래스 패키지 경로
org.zerock.service : 트랜잭션 적용등을 위한 서비스 패키지 경로
org.zerock.persistence : MyBatis DAO 패키지
org.zerock.domain : 데이터 저장빈 클래스 패키지
org.zerock.mappers.* : MyBatis xml 매퍼 패키지

tbl_board 테이블 생성

```
create table tbl_board(
   bno number(38) primary key
   ,writer varchar2(100) not null --작성자
   ,title varchar2(200) not null --제목
   ,content varchar2(4000) --내용
   ,viewcnt number(38) default 0 --조회수 default 0제약조
   --건을 주면 해당컬럼에 레코드를 insert하지 않으면 기본값
   --0이 저장됨.
   ,regdate date --등록날짜
);
select * from tbl_board order by bno desc;
```

bno_seq 시퀀스 생성

```
create sequence bno_seq
increment by 1 --1씩 증가 옵션
start with 1 -- 1부터 시작
nocache;

--시퀀스 번호값 발생
select bno_seq.nextval from dual;
```

method=post로 전달되는 한글을 서버에서 받을 때 안깨지게 하는 web.xml 설정작업

```
<!-- method=post로 전달되는 한글데이터를 안깨지게 함-->
      <filter>
       <filter-name>encoding</filter-name>
       <filter-class>
   org.springframework.web.filter.CharacterEncodingFilter
       </filter-class>
       <init-param>
         <param-name>encoding</param-name>
         <param-value>UTF-8</param-value>
       </init-param>
      </filter>

      <filter-mapping>
         <filter-name>encoding</filter-name>
```

```
        <url-pattern>/*</url-pattern>
    </filter-mapping>
```

org.zerock.domain BoardVO.java 데이터 저장 빈 클래스

```
package org.zerock.domain;

public class BoardVO {
/* 테이블 컬럼명과 일치하는 빈클래스 변수명을 정의한다.
 */
        private int bno;
        private String writer;
        private String title;
        private String content;
        private int viewcnt;//조회수
        private String regdate;//등록날짜

        //페이징
        private int startrow;//시작행번호
        private int entrow;//끝행번호

        public int getBno() {
                return bno;
        }
        public void setBno(int bno) {
                this.bno = bno;
        }
        public String getWriter() {
                return writer;
        }
        public void setWriter(String writer) {
                this.writer = writer;
        }
        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 int getViewcnt() {
		return viewcnt;
	}
	public void setViewcnt(int viewcnt) {
		this.viewcnt = viewcnt;
	}
	public String getRegdate() {
		return regdate;
	}
	public void setRegdate(String regdate) {
		this.regdate = regdate;
	}
	public int getStartrow() {
		return startrow;
	}
	public void setStartrow(int startrow) {
		this.startrow = startrow;
	}
	public int getEntrow() {
		return entrow;
	}
	public void setEntrow(int entrow) {
		this.entrow = entrow;
	}
}
```

org.zerock.persistence BoardDAO.java 인터페이스

```
package org.zerock.persistence;

import java.util.List;

import org.zerock.domain.BoardVO;

public interface BoardDAO {

	void insertBoard(BoardVO b);
	int getRowCount();
	List<BoardVO> getBoardList(BoardVO b);
```

```
        BoardVO getCont(int bno);
        void editBoard(BoardVO eb);
        void delBoard(int bno);
        void updateHit(int bno);
}
```

org.zerock.persistence BoardDAOImpl.java

```
package org.zerock.persistence;

import java.util.List;

import org.apache.ibatis.session.SqlSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
import org.zerock.domain.BoardVO;

@Repository
public class BoardDAOImpl implements BoardDAO {

        @Autowired //자동의존성 주입
        private SqlSession sqlSession; //mybatis 쿼리문 실행
        //객체 의존성(DI) 주입

        @Override
        public void insertBoard(BoardVO b) {
                this.sqlSession.insert("b_in",b);
                //b_in은 insert 아이디명.
        }//게시물저장

        @Override
        public int getRowCount() {
                return this.sqlSession.selectOne("b_count");
//selectOne()은 단 한개의 레코드만 반환.b_count는 select
//아이디명
        }//총레코드 개수

        @Override
        public List<BoardVO> getBoardList(BoardVO b) {
        return this.sqlSession.selectList("b_list",b);
//mybatis에서 selectList()는 하나이상의 레코드를 검색해서
//컬렉션 List로 반환. b_list는 select아이디명
```

```
	}//목록

	@Override
	public BoardVO getCont(int bno) {
		return this.sqlSession.selectOne("b_cont",bno);
	}//내용보기

	@Override
	public void editBoard(BoardVO eb) {
	sqlSession.update("b_edit",eb);
	}//수정

	@Override
	public void delBoard(int bno) {
		this.sqlSession.delete("b_del",bno);
	}//삭제

	@Override
	public void updateHit(int bno) {
		this.sqlSession.update("b_hit",bno);
	}
}
```

org.zerock.mappers.board board.xml 매퍼 태그

```
<?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="Board">

 <!-- 게시물 저장 -->
  <insert id="b_in">
  <!-- parameterType전달인자 타입을 생략가능 -->
   insert into tbl_board (bno,writer,title,content,
   regdate) values(bno_seq.nextval,#{writer},#{title},
   #{content},sysdate)
  </insert>

  <!-- 총레코드 개수 -->
```

```
  <select id="b_count" resultType="int">
  <!-- resultType속성은 반환타입 -->
   select count(*) from tbl_board
   <!-- count는 레코드 개수를 구하는 오라클 함수 -->
  </select>

  <!-- 목록 -->
  <select id="b_list" resultType="b" parameterType="b">
  <!-- parameterType속성은 전달인자 타입 -->
  select * from
  (select rowNum rNum,bno,writer,title,viewcnt,regdate
  from ( select * from tbl_board order by bno desc))
  where rNum &gt;= #{startrow} and rNum &lt;= #{entrow}
  <!-- &gt;=은 >=, &lt;=은 <= 를 의미 -->
  </select>

  <!-- 내용보기 -->
  <select id="b_cont" parameterType="int"
  resultType="b">
   select * from tbl_board where bno=#{bno}
  </select>

  <!-- 수정 -->
  <update id="b_edit">
   update tbl_board set writer=#{writer},title=#{title},
   content=#{content} where bno=#{bno}
  </update>

   <!-- 삭제 -->
   <delete id="b_del" parameterType="int">
    delete from tbl_board where bno=#{bno}
   </delete>

   <!-- 조회수증가 -->
   <update id="b_hit">
    update tbl_board set viewcnt=viewcnt+1
    where bno=#{bno}
   </update>
</mapper>
```

src/main/resources mybatis-config.xml

```
<?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>
     <typeAlias alias="m"
     type="org.zerock.domain.MemberVO"/>
     <!-- MemberVO 빈클래스 객체명 m을 생성 -->

     <typeAlias alias="b"
     type="org.zerock.domain.BoardVO"/>

</configuration>
```

test 밑의 org.zerock.controller 의 BoardDAOTest.java

```
package org.zerock.controller;

import javax.inject.Inject;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.zerock.domain.BoardVO;
import org.zerock.persistence.BoardDAO;

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

          @Inject //자동의존성 주입
          private BoardDAO boardDAO;

          @Test
          public void boardInsert() throws Exception{
        BoardVO b=new BoardVO();
```

```
        b.setWriter("홍길동"); b.setTitle("제목입니다.");
        b.setContent("내용입니다.");

        this.boardDAO.insertBoard(b);//게시물 저장
          }
}
```

계층별 구현 – 비즈니스 계층

1. 비즈니스 계층은 고객의 요구사항이 반영되는 영역이다.
2. 비즈니스 영역에 만들어 지는 클래스나 인터페이스는 반드시 고객 요구사항과 일치하도록 설계돼야 한다.
3. 스프링에서 비즈니스 영역은 일반적으로 서비스(이하 Service)라는 이름을 칭한다. 일반적으로 개발순서는 xxxService 인터페이스를 정의하고, xxxServiceImpl이라는 구현 클래스를 만들어 주는 순서로 진행한다.
4. 비즈니스 계층의 구현 목적
 가. 쉽게 말래서 컨트롤러와 DAO사이의 접착제 역할을 한다.
 나. 고객마다 다른 부분을 처리할 수 있는 완충장치 역할을 한다.
 다. 데이터베이스를 이용하게 되면 트랜잭션 처리로 인한 일관성을 유지하게 해준다.

root-context.xml Service 설정

```
<!-- 비지니스 오토스캔 추가. 이 패키지의 역할은 첫번째
    로 컨트롤과 모델 dao를 연결하는 중간 매개체 역할을 한
    다. 둘째 고객의 추가요구사항을 반영할 때 개발로직코드를
    유연하게 작성가능하게 한다. 셋째 스프링의 AOP를 통한
    트랜잭션 적용 핵심이다. -->
    <context:component-scan  base-package="org.zerock.service" />
```

BoardService.java 자바 코드

```
package org.zerock.service;

import java.util.List;

import org.zerock.domain.BoardVO;

public interface BoardService {

          void insertBoard(BoardVO b);
          int getListCount();
          List<BoardVO> getBoardList(BoardVO b);
          BoardVO getCont(int bno);
          void editBoard(BoardVO eb);
          void delBoard(int bno);
          void updateHit(int bno);
```

```
}
```

BoardServiceImpl.java 코드

```
package org.zerock.service;

import java.util.List;

import javax.inject.Inject;

import org.springframework.stereotype.Service;
import org.zerock.domain.BoardVO;
import org.zerock.persistence.BoardDAO;

@Service
public class BoardServiceImpl implements BoardService {

	@Inject
	private BoardDAO boardDAO;

	@Override
	public void insertBoard(BoardVO b) {
	this.boardDAO.insertBoard(b);
	}
	@Override
	public int getListCount() {
		return this.boardDAO.getRowCount();
	}
	@Override
	public List<BoardVO> getBoardList(BoardVO b) {
		return this.boardDAO.getBoardList(b);
	}
	@Override
	public BoardVO getCont(int bno) {
		return this.boardDAO.getCont(bno);
	}
	@Override
	public void editBoard(BoardVO eb) {
		this.boardDAO.editBoard(eb);
	}
	@Override
	public void delBoard(int bno) {
```

```
                this.boardDAO.delBoard(bno);
        }
        @Override
        public void updateHit(int bno) {
        this.boardDAO.updateHit(bno);
        }
}
```

컨트롤러 선언

1. 스프링 MVC를 이용하는 경우 컨트롤러 역시 @Controller 라는 애노테이션을 추가하는 것만으로 설정이 완료된다. org.zerock.controller.BoardController.java

```
package org.zerock.controller;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;
import org.zerock.domain.BoardVO;
import org.zerock.service.BoardService;

@Controller
@RequestMapping("/board/*") //컨트롤 자체에 매핑주소
//board를 등록
public class BoardController {

        @Autowired //자동의존성 주입
        private BoardService boardService;

        @RequestMapping(value="/board_write",
                        method=RequestMethod.GET)
        //get방식으로 접근하는 매핑주소 board_write를 처리
        public void board_write() throws Exception{
```

```
                //리턴타입이 없는 void형이면 매핑주소가 뷰페이지
                //파일명 즉 board_write.jsp가된다.
        }//board_write()=>게시판 글쓰기 폼

        //게시물 저장
        @RequestMapping(value="/board_write",
                        method=RequestMethod.POST)
        //POST로 접근하는 매핑주소를 처리한다.폼태그에서 액션
        //서버 매핑주소를 등록하지 않으면 이전 매핑주소값이
        //액션 매핑주소가 된다. 같은 매핑주소값 구분은 GET OR
        //POST로 한다.
        public String board_write_ok(BoardVO b,
                        RedirectAttributes rttr) throws Exception{
    //BoardVO b는 board_write.jsp의 네임피라미터 이름과
//BoardVO빈클래스 변수명이 같으면 BoardVO b라고 전달인자만
//사용해도 b객체에 값이 저장되어 쉽게 전달된다.
                this.boardService.insertBoard(b);//게시물저장
                rttr.addFlashAttribute("msg","SUCCESS");
                //msg키이름에 SUCCESS문자열을 담아서 다른 매핑주소
                //로 전달한다.이 방법은 웹주소창에 값 노출이 안된
                //다.보안상 좋다.
                return "redirect:/board/board_list";//get방식으로
                // /board/board_list로 매핑주소가 이동된다.
        }//board_write_ok()

        //목록보기
        @RequestMapping(value="/board_list",
                        method=RequestMethod.GET)
        public void board_list(Model m,
                        HttpServletRequest request,
                        @ModelAttribute BoardVO b) throws Exception{
//@ModelAttribute BoardVO b는 b객체 생성 효과

                int page=1;//현재 쪽번호
                int limit=10;//한페이지 보여지는 목록개수
                if(request.getParameter("page") != null) {
//get으로 전달된 쪽번호가 있는 경우 실행
page=Integer.parseInt(request.getParameter("page"));
//쪽번호를 정수 숫자로 바꿔서 저장
                }
```

```
                    b.setStartrow((page-1)*10+1);//시작행번호
                    b.setEntrow(b.getStartrow()+limit-1);//끝행번호

                    int listCount=this.boardService.getListCount();
                    //총게시물수
List<BoardVO> blist=this.boardService.getBoardList(b);
//게시물 목록

          //총페이지수
          int maxpage=(int)((double)listCount/limit+0.95);
          //현재페이지에 보여질 시작페이지
int startpage=(((int)((double)page/10+0.9))-1)*10+1;
          //현재 페이지에 보여질 마지막 페이지
          int endpage=maxpage;

       if(endpage>startpage+10-1) endpage=startpage+10-1;

          m.addAttribute("list",blist);//list 키이름에
          //blist 를 저장
          m.addAttribute("totalCount",listCount);
          //totalCount 키이름에 listCount저장
          m.addAttribute("startpage",startpage);
          m.addAttribute("endpage",endpage);
          m.addAttribute("maxpage",maxpage);
          m.addAttribute("page",page);//page키이름에 쪽번호
          //를 저장
          }//board_list()->반환타입이 void라서 매핑주소가
          //뷰페이지 파일명이 된다.

          //게시물내용보기
          @RequestMapping("/board_cont")//get or post방식일때
          //모두 실행된다.
          public ModelAndView board_cont(
                              @RequestParam("bno") int bno,
                              @RequestParam("page") int page)
          throws Exception{
//@RequestParam("bno")는 request.getParameter("bno")와
//같다. 즉 bno 피라미터 이름에 전달된 번호값을 가져옴
                    this.boardService.updateHit(bno);
                    //조회수 증가
```

```
                BoardVO b=this.boardService.getCont(bno);
                //디비로 부터 번호에 해당하는 레코드값을 가져옴

                ModelAndView cm=new ModelAndView();
                cm.addObject("b",b);//b키이름에 b객체값을 저장
                cm.addObject("page",page);
                cm.setViewName("board/board_cont");//뷰페이지 경
                //로와 파일명
                return cm;
        }

        //get방식 수정폼 매핑작업
        @RequestMapping("/board_edit")
        public ModelAndView board_edit(int bno)
        throws Exception{
                //int bno는 히든으로 전달된 번호값을 저장
                BoardVO eb=this.boardService.getCont(bno);//번호
                //에 해당하는 레코드값을 가져옴
                ModelAndView em=
                                new ModelAndView("board/board_edit");
//생성자 인자값으로 뷰페이지 경로와 파일명 설정
                em.addObject("b",eb);//b키이름에 eb객체를 저장
                return em;
        }

        //post로 접근하는 수정완료
        @RequestMapping("/board_edit_ok")
        public String board_edit_ok(
                        @ModelAttribute BoardVO eb,
                        RedirectAttributes rttr) throws Exception{
//@ModelAttribute BoardVO eb는 board_edit.jsp의 네임피라
//미터이름과 빈클래스 변수명이 일치하면 eb객체에 입력값이
//저장되어 있다.
                this.boardService.editBoard(eb);//게시물 수정
                rttr.addFlashAttribute("msg","SUCCESS");
                return "redirect:/board/board_list";
        }

        //게시물 삭제
        @RequestMapping("/board_del")
```

```
        public ModelAndView board_del(
                        @RequestParam("bno") int bno,
                        RedirectAttributes rttr) throws Exception{
                this.boardService.delBoard(bno);
                rttr.addFlashAttribute("msg","SUCCESS");
return new ModelAndView("redirect:/board/board_list");
        }
}
```