

웹 게시판(CRUD)



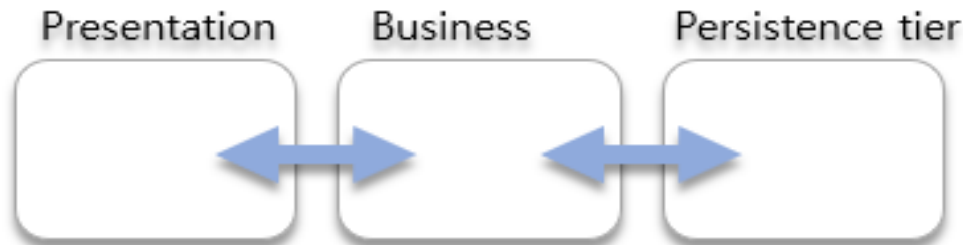
Objectives

- 스프링 MVC를 이용하는 웹 프로젝트 전체 구조에 대한 이해
- 개발의 각 단계에 필요한 설정 및 테스트 환경
- 기본적인 등록, 수정, 삭제, 조회, 리스트 구현
- 목록(리스트) 화면의 페이징(paging) 처리
- 검색 처리와 페이지 이동

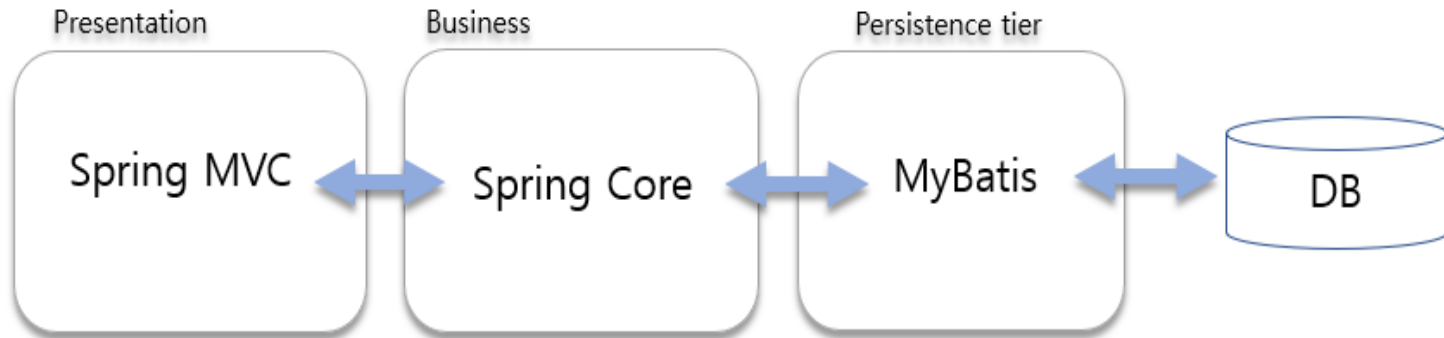
1. 프로젝트의 구성

프로젝트의 구성

- 일반적인 웹 프로젝트의 구조는 3-Tier의 구조를 활용



- 스프링 MVC를 이용하는 예제의 구성



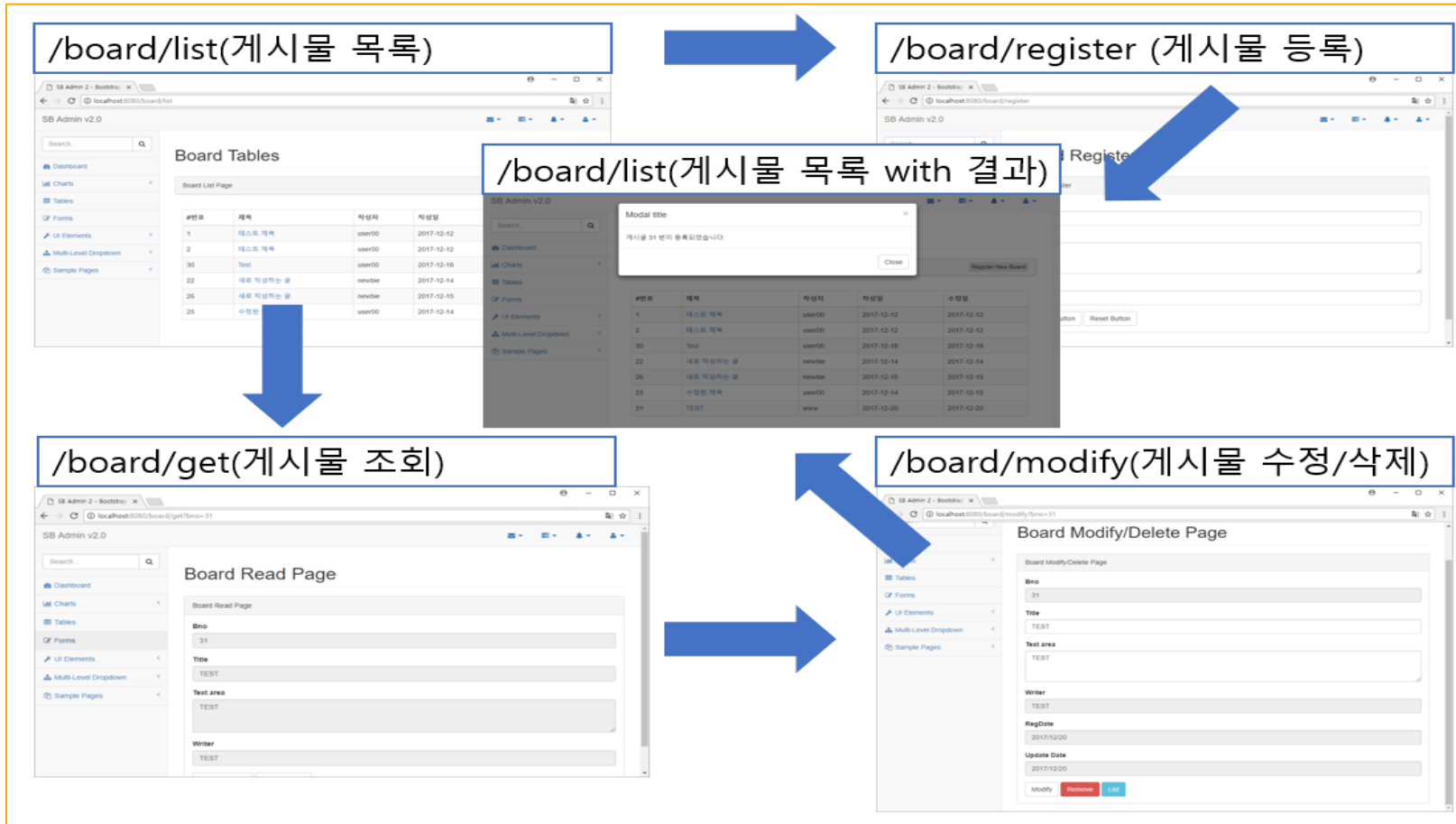
각 영역의 네이밍 규칙

- xxxController: 스프링 MVC에서 동작하는 Controller 클래스
- xxxService, xxxServiceImpl: 비즈니스 영역을 담당하는 인터페이스는 'xxxService'라는 방식을 사용하고, 인터페이스를 구현한 클래스는 'xxxServiceImpl'이라는 이름을 사용
- xxxDAO, xxxRepository: DAO(Data-Access-Object)나 Repository(저장소)라는 이름으로 영역을 따로 구성하는 것이 보편적. 예제에서는 별도의 DAO를 구성하는 대신에 MyBatis의 Mapper 인터페이스를 활용.
- VO, DTO: VO의 경우는 주로 Read Only의 목적이 강하고, 데이터 자체도 Immutable(불변)하게 설계. DTO는 주로 데이터 수집의 용도

프로젝트 패키지의 구성

org.zerock	org.zerock.config	프로젝트와 관련된 설정 클래스들의 보관 패키지
	org.zerock.controller	스프링 MVC의 Controller들의 보관 패키지
	org.zerock.service	스프링의 Service 인터페이스와 구현 클래스 패키지
	org.zerock.domain	VO, DTO 클래스들의 패키지
	org.zerock.persistence	MyBatis Mapper 인터페이스 패키지
	org.zerock.exception	웹 관련 예외처리 패키지
	org.zerock.aop	스프링의 AOP 관련 패키지
	org.zerock.security	스프링 Security 관련 패키지
	org.zerock.util	각종 유틸리티 클래스 관련 패키지

기본적인 게시물의 CRUD 흐름



프로젝트의 생성및 준비

- Spring Legacy Project의 생성
- pom.xml에서 스프링 버전 변경
- spring-test, spring-jdbc, spring-tx 추가
- junit버전 변경
- Servlet 버전 변경
- HikariCP, MyBatis, mybatis-spring, Log4jdbc 추가
- JDBC드라이버 프로젝트내 추가
- 기타 Lombok의 설정 등

데이터베이스내 테이블 생성

```
create sequence seq_board;
```

일련 번호를 위한 **sequence** 생성

```
create table tbl_board (  
  bno number(10,0),  
  title varchar2(200) not null,  
  content varchar2(2000) not null,  
  writer varchar2(50) not null,  
  regdate date default sysdate,  
  update date default sysdate  
);
```

게시물 저장을 위한 테이블 생성

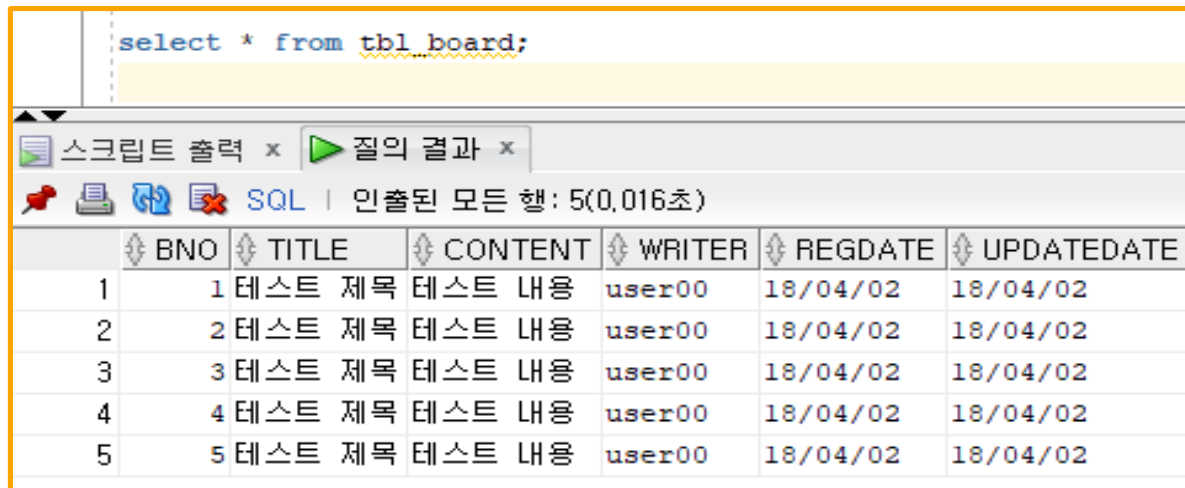
```
alter table tbl_board add constraint  
pk_board  
primary key (bno);
```

게시물의 **PK** 지정

Dummy(더미) 데이터의 추가

```
insert into tbl_board (bno, title, content, writer)  
values (seq_board.nextval, '제목 테스트', '내용 테스트', user00);
```

반복적인 실행으로 여러 개의 데이터 생성 및 확인 및 commit



select * from tbl_board;

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 5(0.016초)

	BNO	TITLE	CONTENT	WRITER	REGDATE	UPDATEDATE
1	1	테스트 제목	테스트 내용	user00	18/04/02	18/04/02
2	2	테스트 제목	테스트 내용	user00	18/04/02	18/04/02
3	3	테스트 제목	테스트 내용	user00	18/04/02	18/04/02
4	4	테스트 제목	테스트 내용	user00	18/04/02	18/04/02
5	5	테스트 제목	테스트 내용	user00	18/04/02	18/04/02

데이터베이스 설정 및 테스트

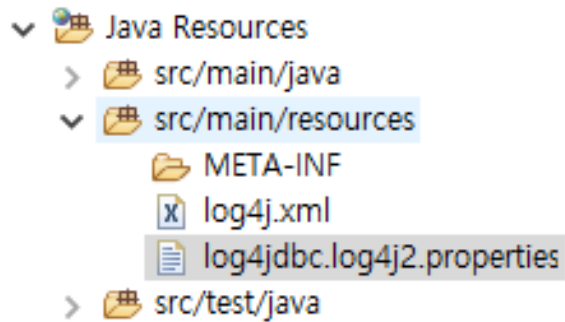
■ root-context.xml

- DataSource의 설정
- SqlSessionFactory 설정

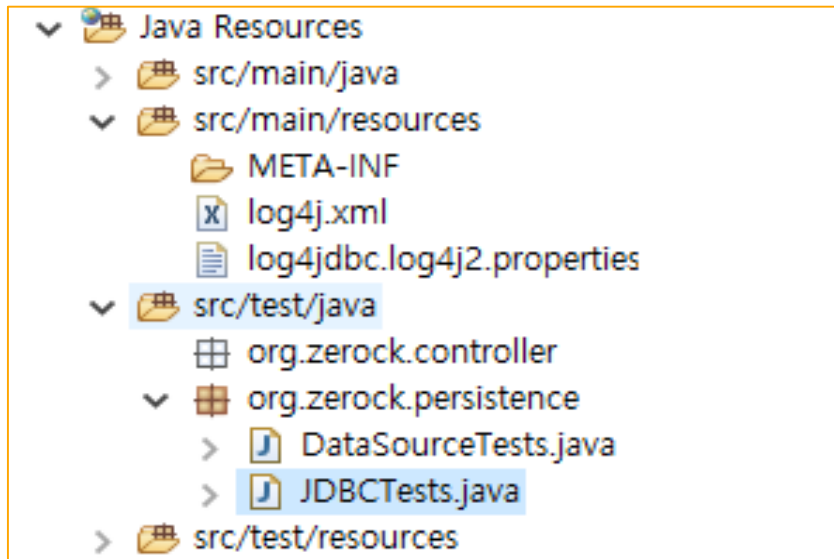
```
<bean id="hikariConfig" class="com.zaxxer.hikari.HikariConfig">
  <property name="driverClassName" value="net.sf.log4jdbc.sql.jdbcapi.DriverSpy"></property>
  <property name="jdbcUrl" value="jdbc:log4jdbc:oracle:thin:@localhost:1521:XE"></property>
  <property name="username" value="book_ex"></property>
  <property name="password" value="book_ex"></property>
</bean>

<!-- HikariCP configuration -->
<bean id="dataSource" class="com.zaxxer.hikari.HikariDataSource"
destroy-method="close">
  <constructor-arg ref="hikariConfig" />
</bean>

<bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource"></property>
</bean>
```



JDBC연결과 DataSource에 대한 테스트 실행



2. 영속 계층 구현

영속 계층의 처리

- 테이블을 반영하는 VO(Value Object) 클래스의 생성
- MyBatis의 Mapper 인터페이스의 작성/XML 처리
- 작성한 Mapper 인터페이스의 테스트

BoardVO 클래스

	❖ COLUMN_NAME	❖ DATA_TYPE	❖ NULLABLE	DATA_DEFAULT	❖ COLUMN_ID
1	BNO	NUMBER(10,0)	No	(null)	1
2	TITLE	VARCHAR2(200 BYTE)	No	(null)	2
3	CONTENT	VARCHAR2(2000 BYTE)	No	(null)	3
4	WRITER	VARCHAR2(50 BYTE)	No	(null)	4
5	REGDATE	DATE	Yes	sysdate	5
6	UPDATEDATE	DATE	Yes	sysdate	6

```
package org.zerock.domain;
```

```
import java.util.Date;  
import lombok.Data;
```

```
@Data
```

```
public class BoardVO {  
    private Long bno;  
    private String title;  
    private String content;  
    private String writer;  
    private Date regdate;  
    private Date updateDate;  
}
```

Mapper인터페이스

```
package org.zerock.mapper;

import java.util.List;

import org.apache.ibatis.annotations.Select;
import org.zerock.domain.BoardVO;

public interface BoardMapper {

    @Select("select * from tbl_board where bno > 0")
    public List<BoardVO> getList();

}
```


BoardMapper의 테스트

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("file:src/main/webapp/WEB-INF/spring/root-context.xml")
@Log4j
public class BoardMapperTests {
    @Setter(onMethod = @__(@Autowired))
    private BoardMapper mapper;

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

```
INFO : jdbc.resultset - 1. ResultSet.isNull() returned false
```

```
INFO : jdbc.resultsettable -
```

-----	-----	-----	-----	-----	-----
bno	title	content	writer	regdate	updatedate
-----	-----	-----	-----	-----	-----
1	테스트 제목	테스트 내용	user00	2018-06-22 11:30:51.0	2018-06-22 11:30:51.0
2	테스트 제목	테스트 내용	user00	2018-06-22 11:31:21.0	2018-06-22 11:31:21.0
3	테스트 제목	테스트 내용	user00	2018-06-22 11:31:21.0	2018-06-22 11:31:21.0
4	테스트 제목	테스트 내용	user00	2018-06-22 11:31:21.0	2018-06-22 11:31:21.0
5	테스트 제목	테스트 내용	user00	2018-06-22 11:31:21.0	2018-06-22 11:31:21.0
-----	-----	-----	-----	-----	-----

```
INFO : jdbc.resultset - 1. ResultSet.next() returned false
```

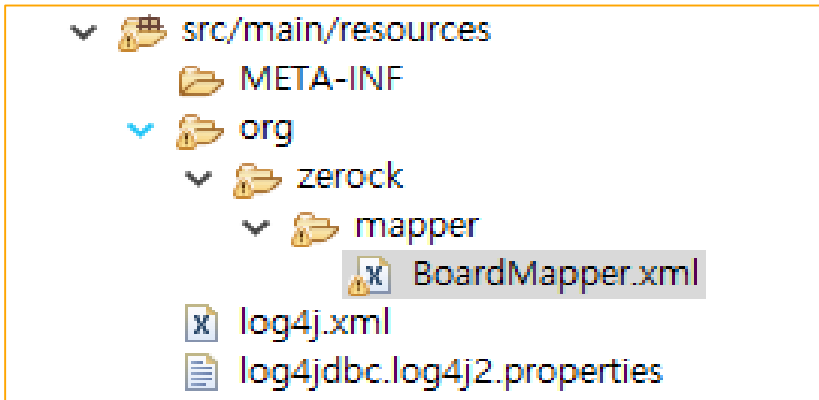
```
INFO : jdbc.resultset - 1. ResultSet.close() returned void
```

```
INFO : jdbc.audit - 1. Connection.getMetaData() returned oracle.jdbc.driver.OracleDatabaseMetaData@4044fb95
```

```
INFO : jdbc.audit - 1. PreparedStatement.isClosed() returned false
```

```
INFO : jdbc.audit - 1. PreparedStatement.close() returned
```

Mapper XML파일



BoardMapper.xml

```
<?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">

  <select id="getList" resultType="org.zerock.domain.BoardVO">
    select * from tbl_board where bno > 0
  </select>

</mapper>
```

게시물 등록(Create)

- 생성된 게시물의 번호를 사용하는지에 따른 구분
 - insert만 처리되고 생성된 PK 값을 알 필요가 없는 경우
 - insert문이 실행되고, 생성된 PK 값을 알아야 하는 경우
<selectkey> 사용

```
public interface BoardMapper {  
  
    // @Select("select * from tbl_board where bno > 0")  
    public List<BoardVO> getList();  
    public void insert(BoardVO board);  
    public void insertSelectKey(BoardVO board);  
  
}
```

BoardMapper.xml

...

```
<insert id="insert">
  insert into tbl_board (bno,title,content,writer)
  values (seq_board.nextval, #{title}, #{content}, #{writer})
</insert>
```

```
<insert id="insertSelectKey">
```

```
  <selectKey keyProperty="bno" order="BEFORE"
    resultType="long">
    select seq_board.nextval from dual
  </selectKey>
```

```
    insert into tbl_board (bno,title,content, writer)
    values (#{bno}, #{title}, #{content}, #{writer})
  </insert>
```

<selectkey>

- SQL이 실행되기 전에 별도의 PK값등을 얻기 위해서 사용
- order='before' 를 이용해서 insert구문이 실행되기 전에 호출
- keyProperty를 통해 BoardVO의 bno값으로 세팅

BoardMapper의 insert테스트

- spring-test를 이용한 테스트 환경을 이용해서 동작 확인
- <selectkey>를 이용하는 경우와 그렇지 않은 경우의 비교

```
@Test
public void testInsert() {

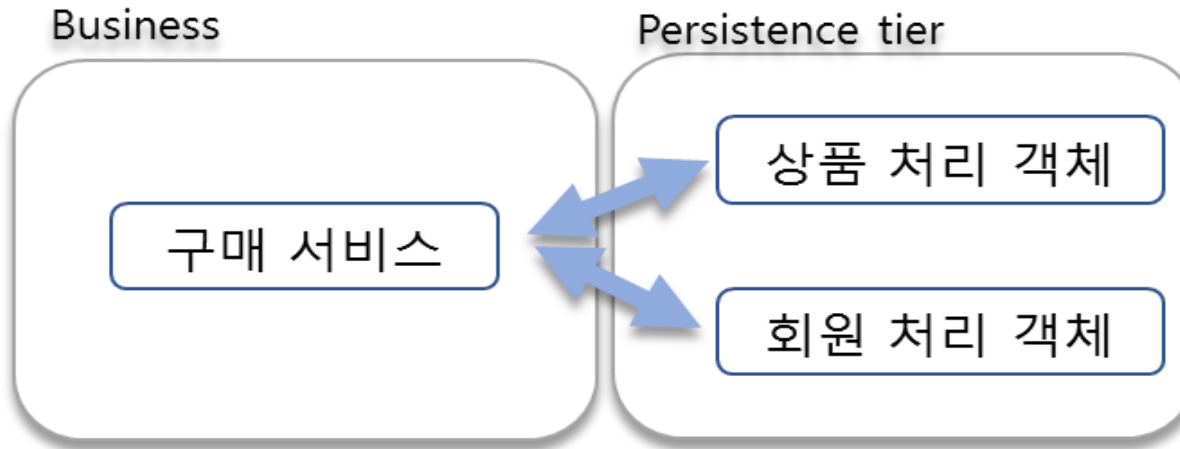
    BoardVO board = new BoardVO();
    board.setTitle("새로 작성하는 글");
    board.setContent("새로 작성하는 내용");
    board.setWriter("newbie");

    mapper.insert(board);

    log.info(board);
}
```

게시물의 조회(read)

- BoardMapper에 read관련 메서드의 추가
- BoardMapper.xml의 SQL 추가
- 테스트를 통한 확인



```
public interface BoardMapper {  
    ...  
    public BoardVO read(Long bno);  
}
```

```
<select id="read" resultType="org.zerock.domain.BoardVO">  
    select * from tbl_board where bno = #{bno}  
</select>
```

```
@Test  
public void testRead() {  
  
    // 존재하는 게시물 번호로 테스트  
    BoardVO board = mapper.read(5L);  
  
    log.info(board);  
}
```


게시물의 삭제

- BoardMapper인터페이스에 메서드 추가 – 파라미터는 PK
- BoardMapper.xml의 수정

```
public interface BoardMapper {  
    ...  
    public int delete(Long bno);  
}
```

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

게시물의 수정

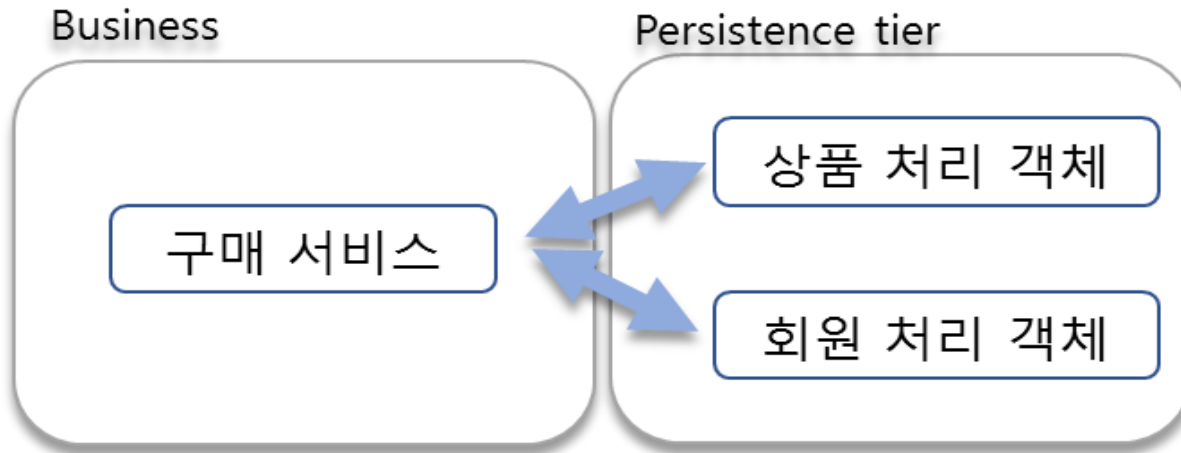
```
public interface BoardMapper {  
    ...  
    public int update(BoardVO board);  
}
```

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

3. 비즈니스 계층 구현

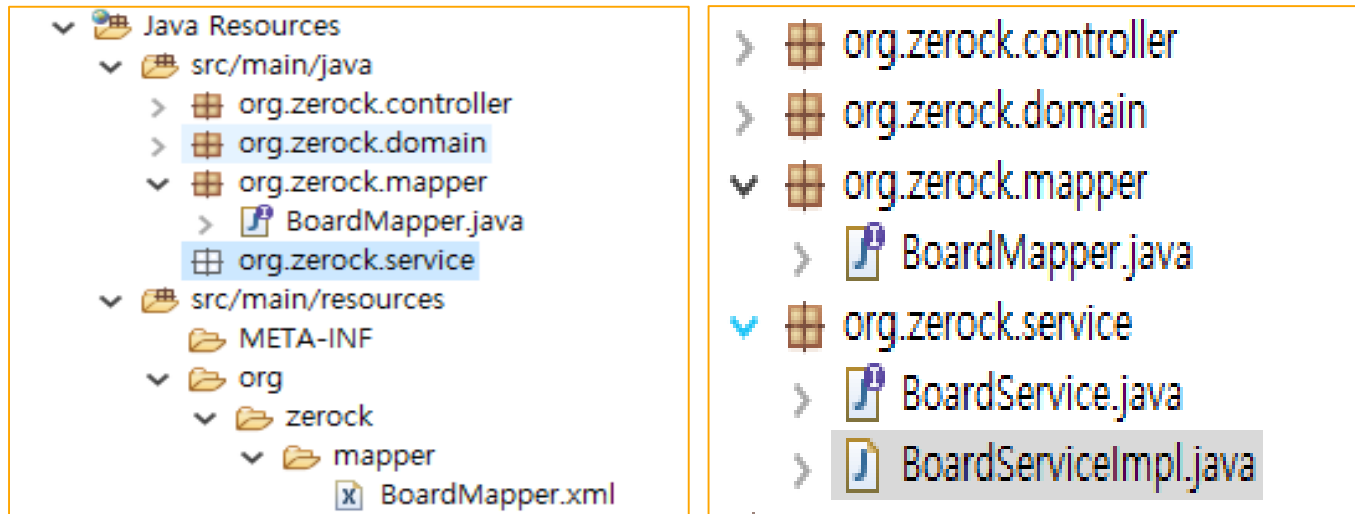
비즈니스 계층(서비스 계층)

- 고객의 요구사항을 반영하는 계층
- 업무의 단위로 설계
 - 트랜잭션의 단위
- 여러 개의 Mapper나 DAO를 사용하는 경우가 존재함
- xxxService의 형태로 작성



서비스 패키지 설정

- 인터페이스와 클래스를 설정하고, root-context.xml에 등록



root-context.xml의 일부

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

```
public interface BoardService {
    public void register(BoardVO board);
    public BoardVO get(Long bno);
    public boolean modify(BoardVO board);
    public boolean remove(Long bno);
    public List<BoardVO> getList();
}
```

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

    //spring 4.3 이상에서 자동 처리
    private BoardMapper mapper;

    @Override
    public void register(BoardVO board) { }

    ..이하 생략..
```

@Service 어노테이션

- @Service는 스프링에 빈으로 등록되는 서비스객체의 어노테이션
- XML의 경우에는 <component-scan>에서 조사하는 패키지의 클래스들 중에 @Service가 있는 클래스의 인스턴스를 스프링의 빈으로 설정

서비스 계층의 구현과 테스트 진행

- 원칙적으로는 서비스 계층 역시 Mapper나 DAO와 같이 별도로 테스트를 진행하는 것이 바람직
- 하나의 Mapper나 DAO를 이용하는 경우에는 테스트를 생략하는 경우도 많은 편

4. 프레젠테이션(웹) 계층의 **CRUD** 구현

웹 계층의 구현

- 웹 계층에서 가장 먼저 설계하는 것은 URI의 설계

Task	URL	Method	Parameter	From	URL 이동
전체 목록	/board/list	GET			
등록 처리	/board/register	POST	모든 항목	입력화면 필요	이동
조회	/board/read	GET	bno=123		
삭제 처리	/board/modify	POST	bno	입력화면 필요	이동
수정 처리	/board/remove	POST	모든 항목	입력화면 필요	이동

진행 작업의 순서

- 목록 페이지 - 모든 진입 경로인 동시에 입력을 가는 링크
- 등록 입력/처리 - 게시물 등록 및 처리, 처리후 이동
- 조회 - 목록 페이지에서 특정 게시물로 이동
- 수정/삭제 - 조회 페이지에서 수정/삭제 선택해 처리

BoardController 목록의 처리

- 게시물(BoardVO)의 목록을 Model에 담아서 전달

```
@Controller
@Log4j
@RequestMapping("/board/*")
@AllArgsConstructor
public class BoardController {
    private BoardService service;
    @GetMapping("/list")
    public void list(Model model) {

        Log.info("list");
        model.addAttribute("list", service.getList());
    }
}
```

BoardController의 테스트

```
@RunWith(SpringJUnit4ClassRunner.class)
//Test for Controller
@WebAppConfiguration

@ContextConfiguration({
    "file:src/main/webapp/WEB-INF/spring/root-context.xml",
    "file:src/main/webapp/WEB-INF/spring/appServlet/servlet-context.xml"})
// Java Config
// @ContextConfiguration(classes = {
//     org.zerock.config.RootConfig.class,
//     org.zerock.config.ServletConfig.class} )
@Log4j
public class BoardControllerTests {

    @Setter(onMethod_ = {@Autowired} )
    private WebApplicationContext ctx;

    private MockMvc mockMvc;
```

@Before

```
public void setup() {  
    this.mockMvc = MockMvcBuilders.webAppContextSetup(ctx).build();  
}
```

@Test

```
public void testList() throws Exception {  
  
    Log.info(  
        mockMvc.perform(  
            MockMvcRequestBuilders.get("/board/list"))  
            .andReturn()  
            .getModelAndView()  
            .getModelMap());  
}  
  
}
```

BoardController의 등록 처리

```
@PostMapping("/register")
public String register(BoardVO board, RedirectAttributes rttr) {
    Log.info("register: " + board);
    service.register(board);
    rttr.addFlashAttribute("result", board.getBno());
    return "redirect:/board/list";
}
```

- POST방식으로 처리되는 데이터를 BoardVO 타입의 인스턴스로 바인딩해서 메서드에서 활용
- BoardService를 이용해서 등록 처리
- 'redirect:'를 이용해서 다시 목록으로 이동

등록처리의 테스트

```
@Test
public void testRegister()throws Exception{

    String resultPage = mockMvc.perform(MockMvcRequestBuilders.post("/board/register")
        .param("title", "테스트 새글 제목")
        .param("content", "테스트 새글 내용")
        .param("writer", "user00")
    ).andReturn().getModelAndView().getViewName();

    Log.info(resultPage);

}
```


BoardController의 조회/테스트

```
@GetMapping("/get")
public void get(@RequestParam("bno") Long bno, Model model) {

    Log.info("/get");
    model.addAttribute("board", service.get(bno));
}
```

```
@Test
public void tetGet() throws Exception {

    Log.info(mockMvc.perform(MockMvcRequestBuilders
        .get("/board/get")
        .param("bno", "2"))
        .andReturn()
        .getModelAndView().getModelMap());
}
```

BoardController의 수정/테스트

```
@PostMapping("/modify")
public String modify(BoardVO board, RedirectAttributes rttr) {
    Log.info("modify:" + board);
    if (service.modify(board)) {
        rttr.addFlashAttribute("result", "success");
    }
    return "redirect:/board/list";
}
```

```
@Test
public void testModify() throws Exception {
    String resultPage = mockMvc
        .perform(MockMvcRequestBuilders.post("/board/modify")
            .param("bno", "1")
            .param("title", "수정된 테스트 새글 제목")
            .param("content", "수정된 테스트 새글 내용")
            .param("writer", "user00"))
        .andReturn().getModelAndView().getViewName();
    Log.info(resultPage);
}
```

BoardController의 삭제/테스트

```
@PostMapping("/remove")
public String remove(@RequestParam("bno") Long bno, RedirectAttributes rttr) {
    Log.info("remove..." + bno);
    if (service.remove(bno)) {
        rttr.addFlashAttribute("result", "success");
    }
    return "redirect:/board/list";
}
```

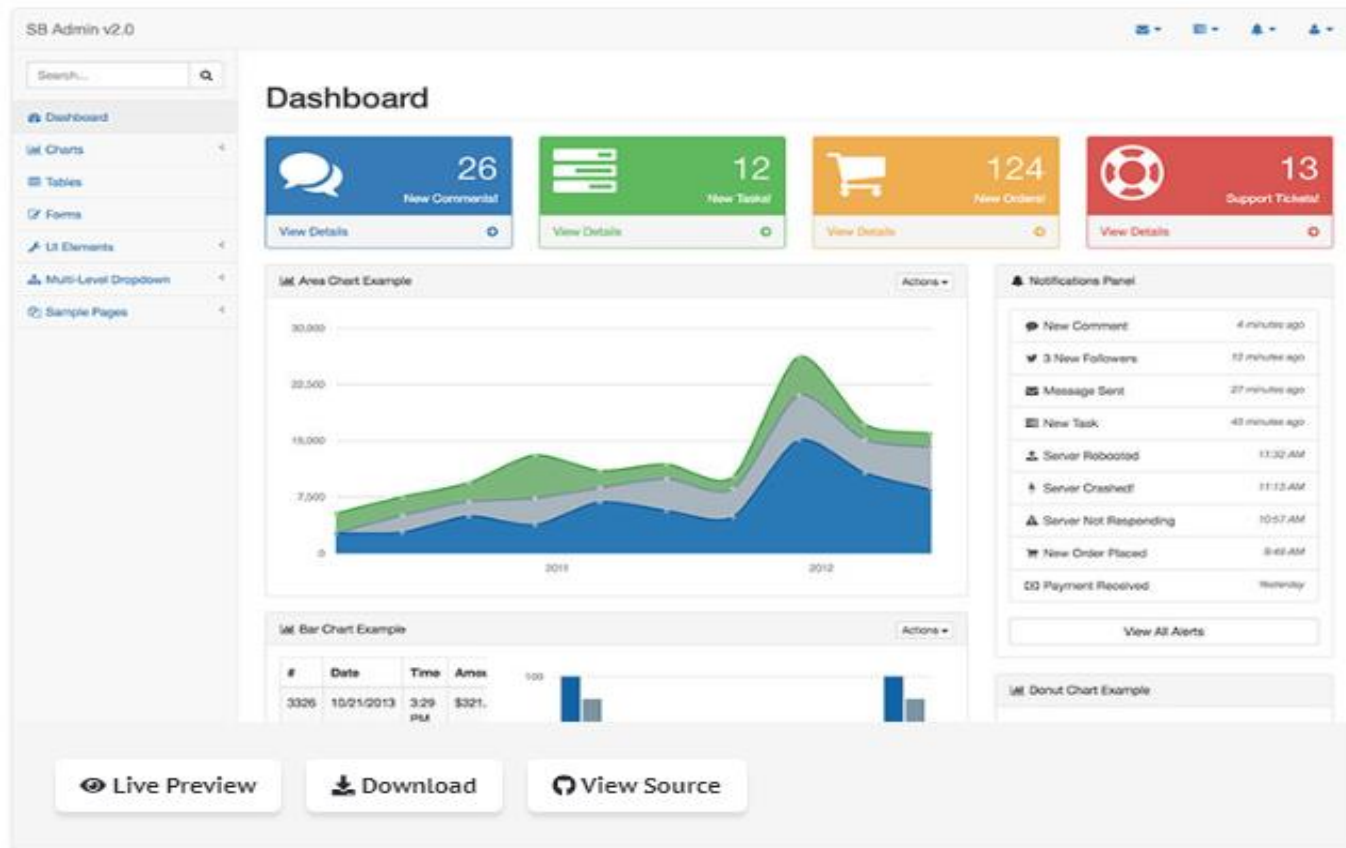
```
@Test
public void testRemove() throws Exception{
    //삭제전 데이터베이스에 게시물 번호 확인할 것
    String resultPage = mockMvc.perform(MockMvcRequestBuilders.post("/board/remove")
        .param("bno", "25")
        ).andReturn().getModelAndView().getViewName();

    Log.info(resultPage);
}
```

5. 화면 처리

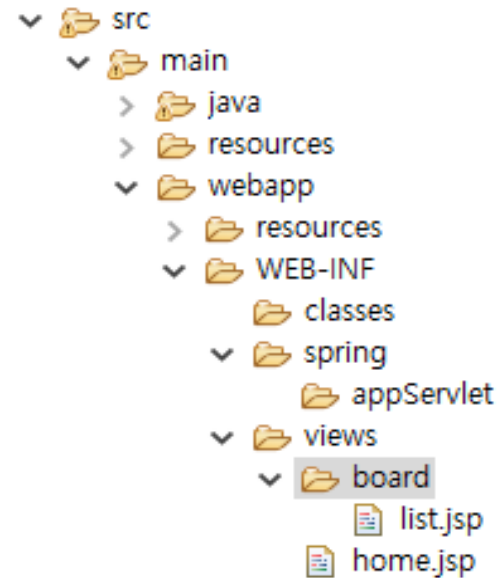
화면의 처리

- SB-Admin 2(MIT 라이선스)를 이용해서 페이지 디자인
- <https://startbootstrap.com/template-overviews/sb-admin-2/>



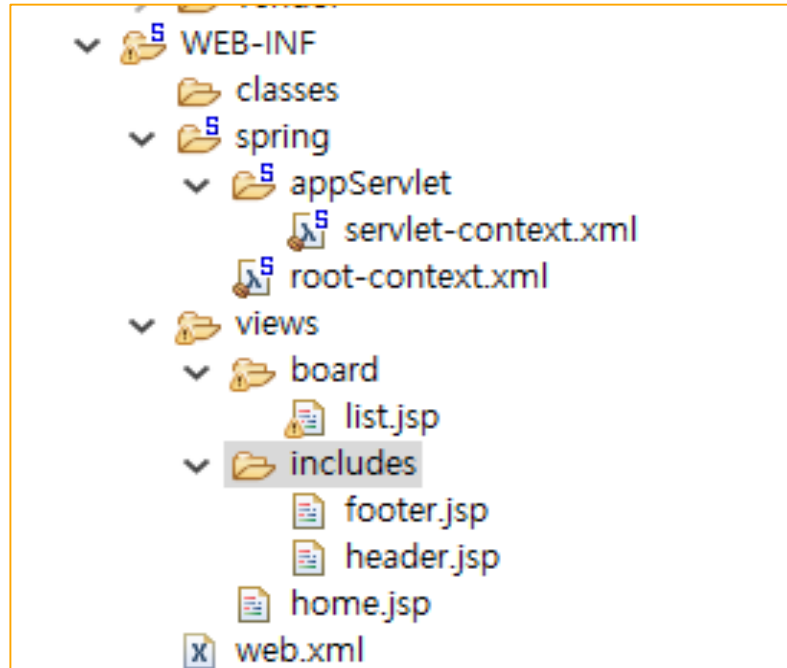
목록 페이지 작성

- 프로젝트의 경로는 '/'를 이용하도록 수정
- Tomcat등을 이용해서 실제로 JSP 처리에 문제 없는지 확인후 진행



includes 적용

- JSP페이지의 공통으로 사용되는 페이지의 일부를 header.jsp와 footer.jsp로 분리해서 각 페이지에서 include 하는 방식으로 사용



list.jsp의 적용

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
```

```
<%@include file="../includes/header.jsp" %>
```

```
    <div class="row">
```

```
        <div class="col-lg-12">
```

```
            <h1 class="page-header">Tables</h1>
```

...생략...

jQuery 버전 변경과 문제

jQuery 교체 후 모바일 크기에서 새로 고침 시에 메뉴가 펼쳐지는 문제

The image displays two browser windows side-by-side, both showing the 'SB Admin v2.0' application. The left window is in desktop view, showing a sidebar menu with options like 'Charts', 'Tables', 'Forms', 'UI Elements', 'Multi-Level Dropdown', and 'Sample Pages'. The right window is in mobile view, where the sidebar menu is expanded, covering the content area. A text box in the center explains the issue: 'jQuery 교체 후 모바일 크기에서 새로 고침 시에 메뉴가 펼쳐지는 문제' (Problem of menu expanding when refreshing in mobile size after jQuery replacement).

Desktop View (Left): The sidebar menu is collapsed. The main content area shows 'Tables' and 'DataTables Advanced Tables'.

Rendering engine	Browser	Platform(s)	Engine version
Gecko	Firefox 1.0	Win 98+ / OSX.2+	1.7
Gecko	Firefox 1.5	Win 98+ / OSX.2+	1.8
Gecko	Firefox 2.0	Win 98+ / OSX.2+	1.8
Gecko	Firefox 3.0	Win 2k+ / OSX.3+	1.9
Gecko	Camino 1.0	OSX.2+	1.8

Mobile View (Right): The sidebar menu is expanded, covering the content area. The main content area is partially obscured by the expanded menu.

```
<script>
$(document).ready(function() {
  $('#dataTables-example').DataTable({
    responsive: true
  });
  $(".sidebar-nav")
    .attr("class", "sidebar-nav navbar-collapse collapse")
    .attr("aria-expanded", 'false')
    .attr("style", "height:1px");
});
</script>
```

목록 화면의 처리

```
<thead>
```

```
<tr>
```

```
<th>#번호</th>
```

```
<th>제목</th>
```

```
<th>작성자</th>
```

```
<th>작성일</th>
```

```
<th>수정일</th>
```

```
</tr>
```

```
</thead>
```

```
<c:forEach items="${list}" var="board">
```

```
<tr>
```

```
<td><c:out value="${board.bno}" /></td>
```

```
<td><c:out value="${board.title}" /></td>
```

```
<td><c:out value="${board.writer}" /></td>
```

```
<td><fmt:formatDate pattern="yyyy-MM-dd"
    value="${board.regdate}" /></td>
```

```
<td><fmt:formatDate pattern="yyyy-MM-dd"
    value="${board.updateDate}" /></td>
```

```
</tr>
```

```
</c:forEach>
```

```
</table>
```

SB Admin 2 - Bootstrap x

localhost:8080/board/list

Tables

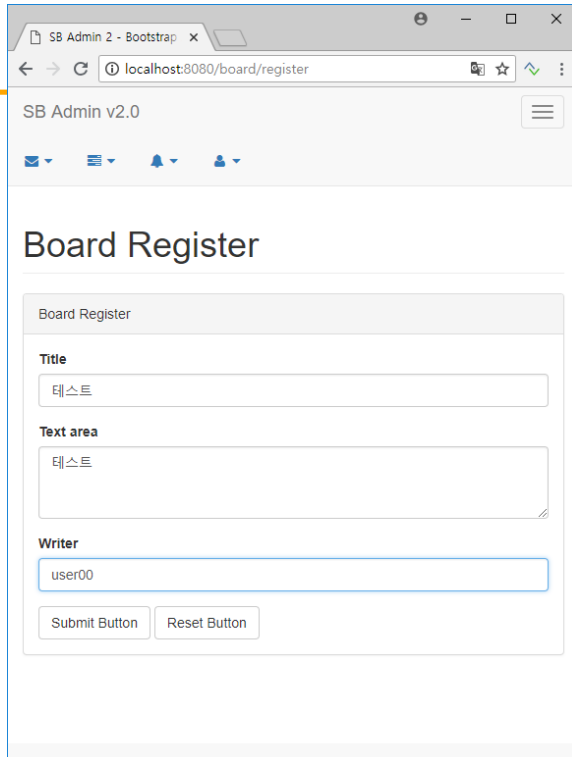
Board List Page

#번호	제목	작성자	작성일	수정일
1	테스트 제목	user00	2018-06-22	2018-06-22
2	테스트 제목	user00	2018-06-22	2018-06-22
3	테스트 제목	user00	2018-06-22	2018-06-22
4	테스트 제목	user00	2018-06-22	2018-06-22
5	테스트 제목	user00	2018-06-22	2018-06-22
21	새로 작성하는 글	newbie	2018-06-23	2018-06-23
22	새로 작성하는 글 select key	newbie	2018-06-23	2018-06-23
23	새로 작성하는 글 select key	newbie	2018-06-23	2018-06-23
24	새로 작성하는 글	newbie	2018-06-25	2018-06-25
25	테스트 새글 제목	user00	2018-06-25	2018-06-25

등록 입력 페이지와 등록

- GET방식으로 게시물 등록 화면 제공
- POST방식으로 실제 게시물 등록 처리
- 이후 목록 페이지로 이동

```
@GetMapping("/register")  
public void register() {  
  
}
```



SB Admin v2.0

Board Register

Board Register

Title
테스트

Text area
테스트

Writer
user00

Submit Button Reset Button

```
views  
└─ board  
    ├── list.jsp  
    └── register.jsp  
includes  
└─ home.jsp
```

한글 깨짐과 필터 설정

SB Admin 2 - Bootstrap

localhost:8080/board/register

SB Admin v2.0

Board Register

Board Register

Title

테스트

Text area

테스트

Writer

user00

Submit Button

Reset Button

Board List Page

#번호	제목	작성자	작성일	수정일
1	테스트 제목	user00	2018-06-22	2018-06-22
2	테스트 제목	user00	2018-06-22	2018-06-22
3	테스트 제목	user00	2018-06-22	2018-06-22
4	테스트 제목	user00	2018-06-22	2018-06-22
5	테스트 제목	user00	2018-06-22	2018-06-22
21	새로 작성하는 글	newbie	2018-06-23	2018-06-23
22	새로 작성하는 글 select key	newbie	2018-06-23	2018-06-23
23	새로 작성하는 글 select key	newbie	2018-06-23	2018-06-23
24	새로 작성하는 글	newbie	2018-06-25	2018-06-25
25	테스트 새글 제목	user00	2018-06-25	2018-06-25
26	이이이이이	user00	2018-06-25	2018-06-25

게시글이 등록되지만 한글이 깨지는 문제가 발생하는 경우

UTF-8 필터 처리

- web.xml을 이용한 필터 설정

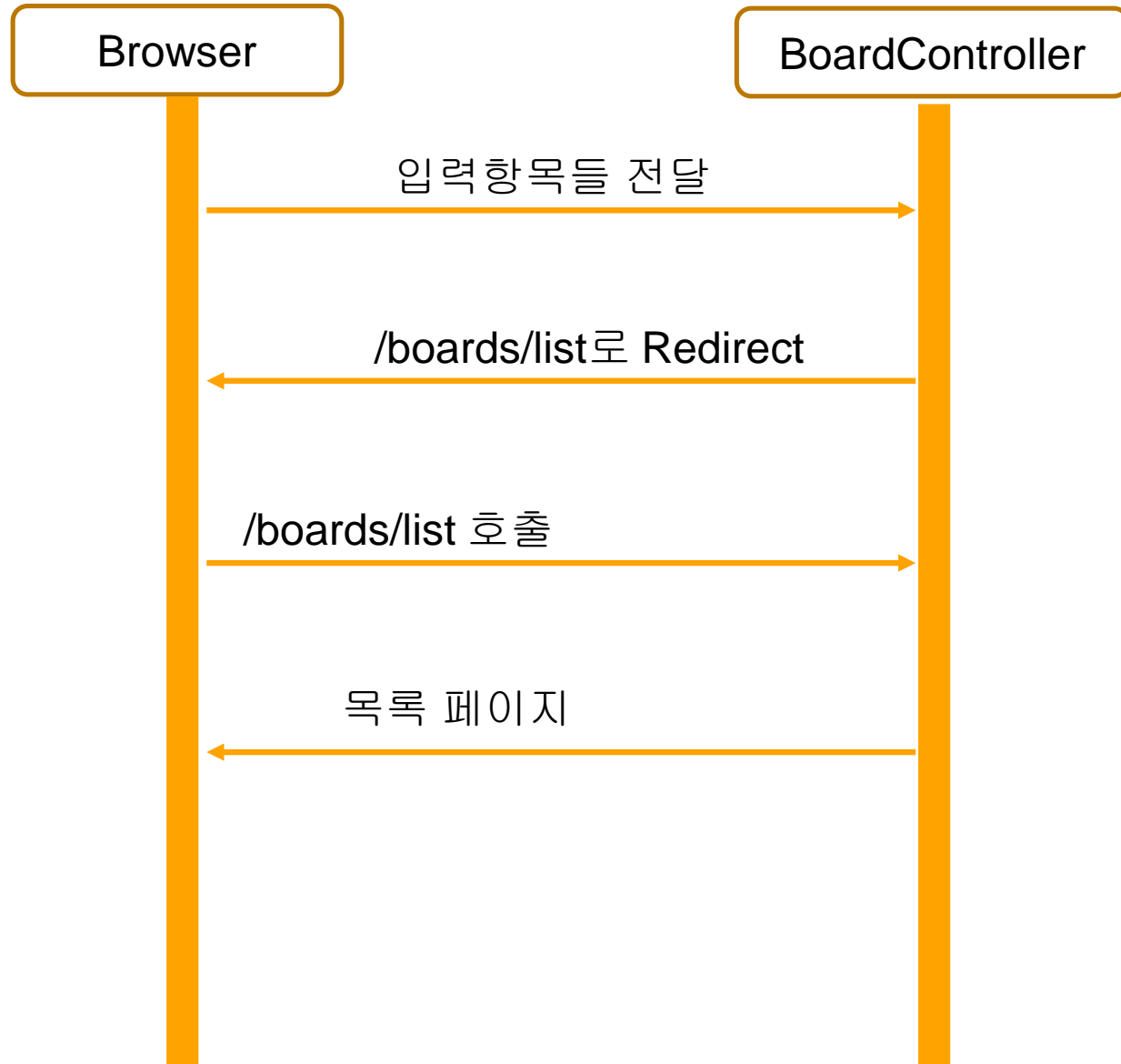
```
<filter>
  <filter-name>encoding</filter-name>
  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
  <init-param>
    <param-name>encoding</param-name>
    <param-value>UTF-8</param-value>
  </init-param>
</filter>

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

@Override

```
protected Filter[] getServletFilters() {
    CharacterEncodingFilter characterEncodingFilter = new CharacterEncodingFilter();
    characterEncodingFilter.setEncoding("UTF-8");
    characterEncodingFilter.setForceEncoding(true);
    return new Filter[] { characterEncodingFilter };
}
```

재전송(redirect) 처리



게시물 작업 이후 재전송

- 게시물의 등록, 수정, 삭제의 경우에 해당
- 작업이 완료된 후에는 리스트 페이지로 다시 이동
- BoardController에서는 RedirectAttributes의 addFlashAttribute()를 이용해서 단 한번만 전송되는 데이터 저장후 전송
- JSP등의 화면에서는 JavaScript를 이용해서 경고창이나 모달창등을 보여주는 형식

```
@PostMapping("/register")  
public String register(BoardVO board, RedirectAttributes rttr) {  
    log.info("register: " + board);  
    service.register(board);  
    rttr.addFlashAttribute("result", board.getBno());  
    return "redirect:/board/list";  
}
```


화면의 처리

```
<script type="text/javascript">
$(document).ready(function(){
    var result = '<c:out value="${result}"/>';
});
</script>
```

작업후 생성되는 결과

```
<script type="text/javascript">
$(document).ready(function(){
    var result = '15';
});
</script>
```

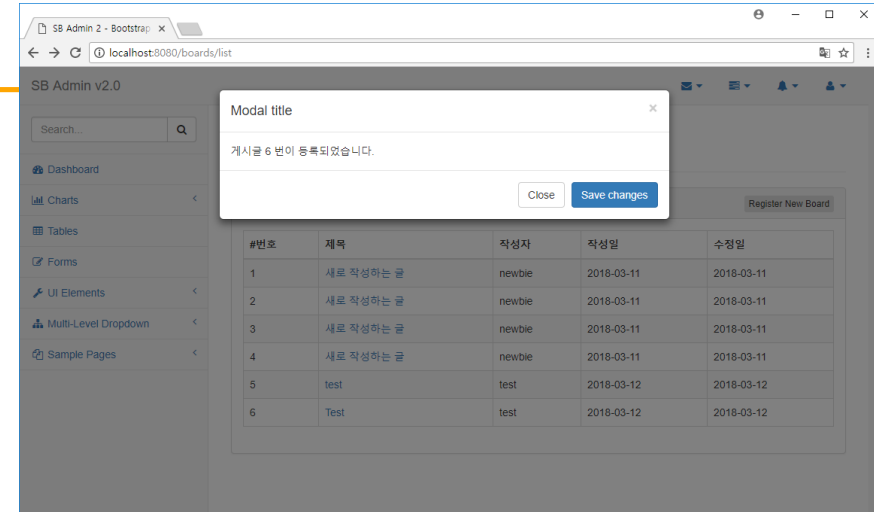
일반 호출의 결과

```
<script type="text/javascript">
$(document).ready(function(){
    var result = '';
});
</script>
```

모달창 보여주기

- BoardController에서 특정한 데이터가 RedirectAttribute에 포함된 경우에 모달창을 보여주기

```
<script type="text/javascript">
$(document).ready(function() {
    var result = '<:out value="${result}"/>';
    checkModal(result);
    function checkModal(result) {
        if (result === '') {
            return;
        }
        if (parseInt(result) > 0) {
            $(".modal-body").html(
                "게시글 " + parseInt(result) + " 번이 등록되었습니다.");
        }
        $("#myModal").modal("show");
    }
});
</script>
```



조회 페이지와 이동

- 목록에서 특정 게시물 선택후 이동 처리
- 조회의 경우는 반드시 GET방식으로만 처리

```
<tr>
<td><c:out value="${board.bno}"/></td>
<td><a href="/board/get?bno=<c:out value="${board.bno}"/>'><c:out value="${board.title}"/></a></td>
<td><c:out value="${board.writer}"/></td>
<td><fmt:formatDate pattern = "yyyy-MM-dd" value = "${board.regdate}" /></td>
<td><fmt:formatDate pattern = "yyyy-MM-dd" value = "${board.updateDate}" /></td>
</tr>
```

뒤로 가기와 windo의 history객체



1

/boards/list



2

/boards/register

/boards/list

3

/boards/list



/boards/register

/boards/list



4

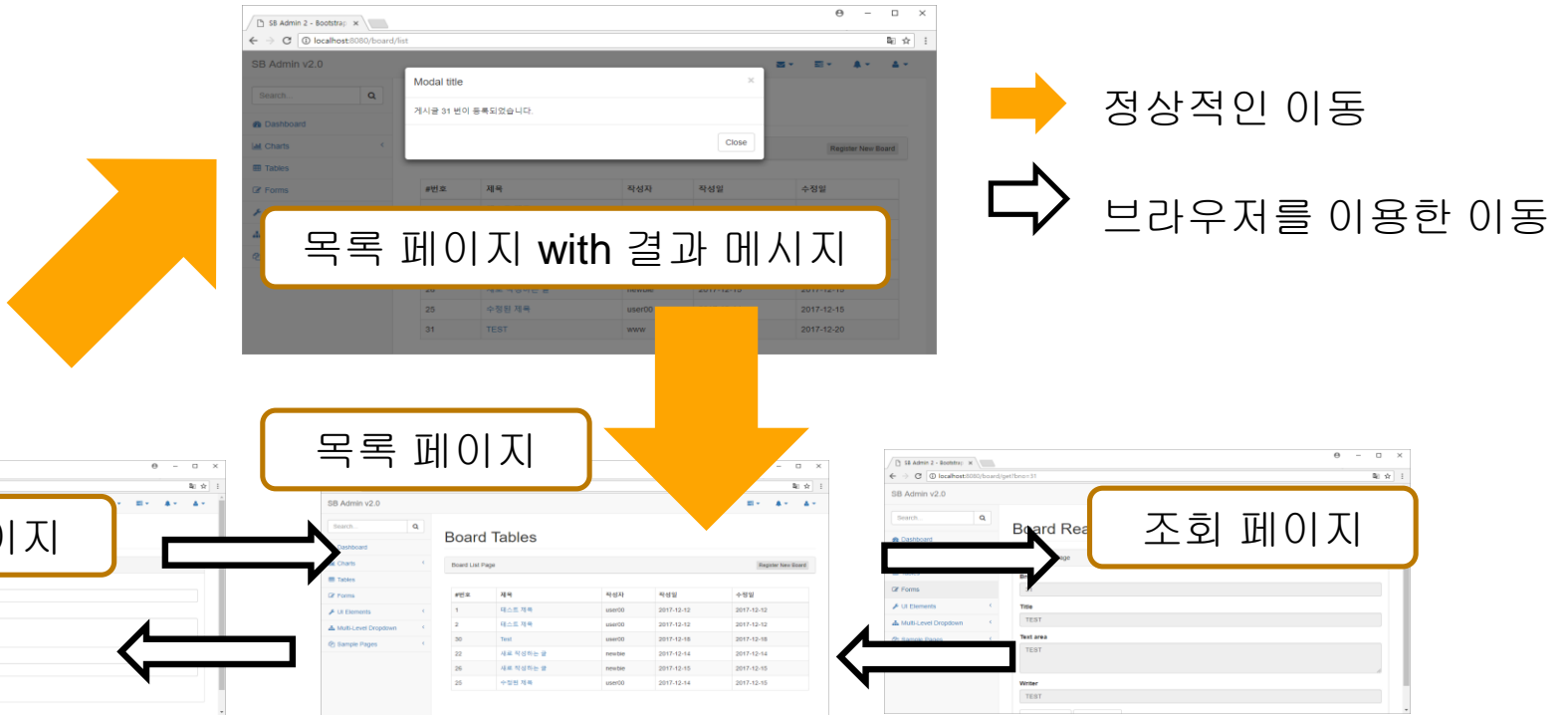
/boards/register

/boards/list



/boards/register

/boards/list



게시물의 수정/삭제 처리

- 게시물 조회이후 수정/삭제 페이지로 이동
- 사용자의 선택에 따라서 작업을 처리
- 수정/삭제는 POST방식으로 처리

SB Admin v2.0

Board Modify Page

Board Modify Page

Bno
19

Title
Test

Text area
test

Writer
Test

Modify Remove List

```
<script type="text/javascript">
$(document).ready(function() {
    var formObj = $("form");
    $('button').on("click", function(e){
        e.preventDefault();
        var operation = $(this).data("oper");
        console.log(operation);
        if(operation === 'remove'){
            formObj.attr("action", "/board/remove");
        }else if(operation === 'list'){
            //move to list
            self.location= "/board/list";
            return;
        }
        formObj.submit();
    });
});
</script>
```

6. 오라클 DB 페이지 처리

order by의 고민

- SQL의 실행 과정
 - SQL 파싱
 - SQL 최적화
 - SQL 실행
- SQL의 처리 과정에서 SQL의 실행 계획(execution plan)이 수립
- 데이터의 정렬이 많은 경우에는 order by가 성능에 나쁜 영향

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			7	4
SORT		ORDER BY	7	4
TABLE ACCESS	TBL_BOARD	FULL	7	3
Other XML				
{info}				
info type="db_version"				

order by 대신에 index

```
select /*+INDEX_DESC(tbl_board pk_board) */
  rownum rn, bno, title, content
from
  tbl_board
where rownum <= 10;
```



	↕ RN	↕ BNO	↕ TITLE	↕ CONTENT
1	1	3670041	Test	TestTestTest
2	2	3670040	테스트 제목	테스트 내용
3	3	3670039	테스트 제목	테스트 내용
4	4	3670038	TEST	TEST
5	5	3670037	수정된 제목	수정된 내용
6	6	3670036	TEST	TEST
7	7	3670035	수정된 제목	수정된 내용
8	8	3670034	새로 작성하는 글	새로 작성하는 내용
9	9	3670033	새로 작성하는 글	새로 작성하는 내용
10	10	3670032	Test	TestTestTest

식별키(PK)와 인덱스

- PK를 생성하면 자동으로 인덱스가 생성

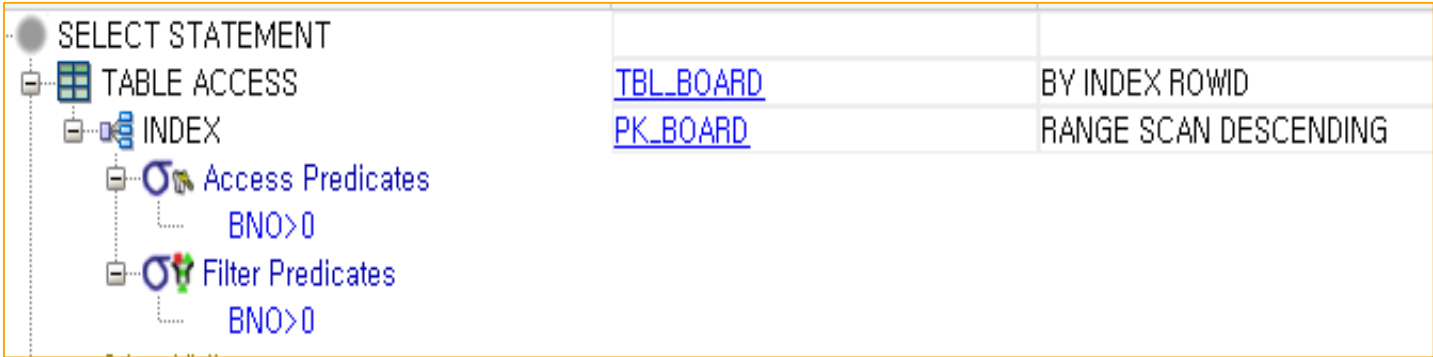
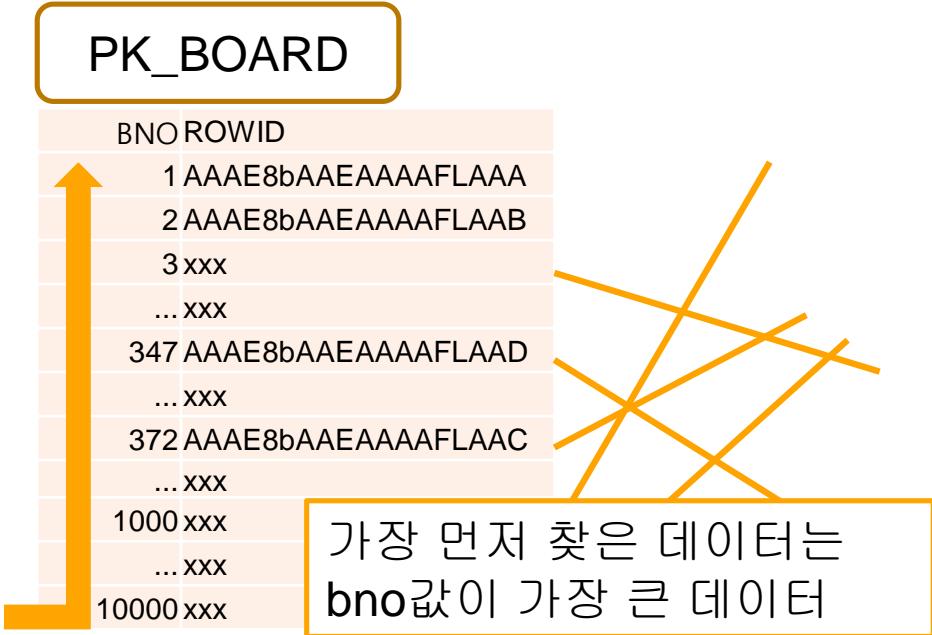
식별키로 만들어진 인덱스

BNO	ROWID
1	AAAE8bAAEAAAAFLAAA
2	AAAE8bAAEAAAAFLAAB
3	xxx
...	xxx
347	AAAE8bAAEAAAAFLAAD
...	xxx
372	AAAE8bAAEAAAAFLAAC
...	xxx
1000	xxx
...	xxx
10000	xxx

테이블의 데이터

ROWID	BNO	TITLE	CONTENT
AAAE8bAAEAAAAFLAAA	1	테스트 제목	테스트 내용
AAAE8bAAEAAAAFLAAB	2	테스트 제목	테스트 내용
AAAE8bAAEAAAAFLAAC	372	테스트 제목	테스트 내용
AAAE8bAAEAAAAFLAAD	347	새로 작성하는 글	새로 작성하는 내용
AAAE8bAAEAAAAFLAAE	348	테스트 제목	테스트 내용
AAAE8bAAEAAAAFLAAF	349	테스트 제목	테스트 내용
AAAE8bAAEAAAAFLAAG	350	Test	Test
AAAE8bAAEAAAAFLAAH	351	새로 작성하는 글	새로 작성하는 내용
AAAE8bAAEAAAAFLAAI	352	새로 작성하는 글	새로 작성하는 내용
AAAE8bAAEAAAAFLAAJ	353	수정된 제목	수정된 내용
AAAE8bAAEAAAAFLAAK	354	TEST	TEST
AAAE8bAAEAAAAFLAAL	355	수정된 제목	수정된 내용
AAAE8bAAEAAAAFLAAM	356	TEST	TEST
AAAE8bAAEAAAAFLAAN	357	TEST	TEST

인덱스를 이용한 정렬



인덱스를 이용하기 위한 힌트

- 개발자의 의도를 힌트를 이용해서 전달
- 힌트 구문은 잘못되어도 SQL처리에는 지장을 주지 않음
- 여러 종류의 힌트가 존재
 - FULL
 - INDEX_DESC, INDEX_ASC

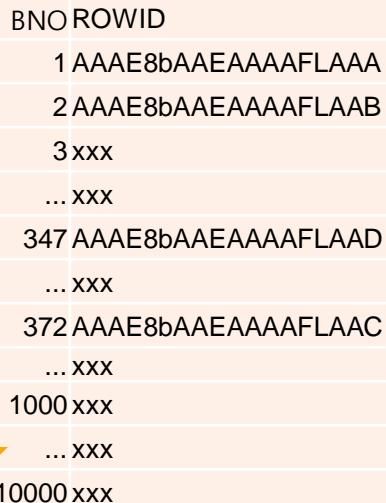
```
SELECT
  /*+ Hint name ( param...) */ column name, ....
FROM
  table name
....
```

ROWNUM과 인덱스

- 테이블에서 최종적으로 나오면서 붙이는 컬럼
- 인덱스를 통해서 나오는지, FULL 스캔을 통하는지에 따라서 ROWNUM이 다르게 나옴

```
select /*+ INDEX_ASC(tbl_board pk_board) */  
       rownum rn, bno, title, content  
from tbl_board;
```

PK_BOARD



BNO	ROWID
1	AAAE8bAAEAAAAFLAAA
2	AAAE8bAAEAAAAFLAAB
3	xxx
...	xxx
347	AAAE8bAAEAAAAFLAAD
...	xxx
372	AAAE8bAAEAAAAFLAAC
...	xxx
1000	xxx
...	xxx
10000	xxx

```
select /*+ INDEX_ASC(tbl_board pk_board) */  
       rownum rn, bno, title, content  
from tbl_board;
```

가장 먼저 찾은 데이터부터 ROWNUM이 1부터 시작

RN	BNO	TITLE	CONTENT
208	211	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다.
209	212	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다.
210	213	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다.
211	214	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다.
212	215	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다.
213	216	수정테스트	test수정테스트수정테스트
214	217	새로 작성하는 글	새로 작성하는 내용
215	218	새로 작성하는 글	새로 작성하는 내용
216	219	테스트 제목	테스트 내용
217	220	테스트 제목	테스트 내용
218	221	테스트 제목	테스트 내용
219	222	테스트 제목	테스트 내용
220	223	수정된 제목	수정된 내용
221	224	새로 작성하는 글 select key	새로 작성하는 내용 select key
222	225	새로 작성하는 글	새로 작성하는 내용

인라인뷰와 페이징

```
select /*+INDEX_DESC(tbl_board pk_board) */
       rownum rn, bno, title, content
from
       tbl_board
where rownum <= 10;
```

RN	BNO	TITLE	CONTENT
1	2359299	테스트 게시물입니다.	테스트 게시물입니다.
2	2359298	테스트 게시물입니다.	테스트 게시물입니다.
3	2359297	테스트 게시물입니다.	테스트 게시물입니다.
4	2359296	테스트 게시물입니다.	테스트 게시물입니다.
5	2359295	테스트 게시물입니다.	테스트 게시물입니다.
6	2359294	테스트 게시물입니다.	테스트 게시물입니다.
7	2359293	테스트 게시물입니다.	테스트 게시물입니다.
8	2359292	테스트 게시물입니다.	테스트 게시물입니다.
9	2359291	테스트 게시물입니다.	테스트 게시물입니다.
10	2359290	수정된 테스트 게시물입니다.	테스트 게시물입니다.

SELECT STATEMENT		
COUNT		STOPKEY
Filter Predicates ROWNUM<=10		
TABLE ACCESS INDEX	TBL_BOARD PK_BOARD	BY INDEX ROWID FULL SCAN DESCENDING

ROWNUM은 1이 포함된 조건으로

```
select /*+INDEX_DESC(tbl_board pk_board) */
  rownum rn, bno, title, content
from
  tbl_board
where rownum <= 20;
```

RN	BNO	TITLE	CONTENT
1	2359299	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다.
2	2359298	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다.
3	2359297	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다.
4	2359296	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다.
5	2359295	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다.
6	2359294	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다.
7	2359293	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다.
8	2359292	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다.
9	2359291	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다.
10	2359290	수정된 테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다.
11	2359289	ㅇㅇㅇㅇㅇㅇㅇㅇ	ㅇㅇㅇㅇㅇㅇㅇㅇ
12	2359288	Test	Test
13	2359287	테스트 새글 제목	테스트 새글 내용
14	2359286	새로 작성하는 글	새로 작성하는 내용
15	2359285	새로 작성하는 글 select key	새로 작성하는 내용 select key
16	2359284	수정된 제목	수정된 내용
17	2359283	테스트 제목	테스트 내용
18	2359282	테스트 제목	테스트 내용
19	2359281	테스트 제목	테스트 내용
20	2359280	테스트 제목	테스트 내용

SELECT

FROM (

SELECT
FROM.... 인라인뷰

)


```

select bno, title, content
from (
    select /*+INDEX_DESC(tbl_board pk_board) */
        rownum rn, bno, title, content
    from
        tbl_board
    where rownum <= 20
)
where rn > 10;

```

RN	BNO	TITLE	CONTENT
1	2359299	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다. 테스트 게시물입니다.
2	2359298	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다. 테스트 게시물입니다.
3	2359297	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다. 테스트 게시물입니다.
4	2359296	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다. 테스트 게시물입니다.
5	2359295	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다. 테스트 게시물입니다.
6	2359294	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다. 테스트 게시물입니다.
7	2359293	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다. 테스트 게시물입니다.
8	2359292	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다. 테스트 게시물입니다.
9	2359291	테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다. 테스트 게시물입니다.
10	2359290	수정된 테스트 게시물입니다.	테스트 게시물입니다. 테스트 게시물입니다. 테스트 게시물입니다.
11	2359289	ㅇㅇㅇㅇㅇㅇㅇㅇ	ㅇㅇㅇㅇㅇㅇㅇㅇ
12	2359288	Test	Test
13	2359287	테스트 새글 제목	테스트 새글 내용
14	2359286	새로 작성하는 글	새로 작성하는 내용
15	2359285	새로 작성하는 글 select key	새로 작성하는 내용 select key
16	2359284	수정된 제목	수정된 내용
17	2359283	테스트 제목	테스트 내용
18	2359282	테스트 제목	테스트 내용
19	2359281	테스트 제목	테스트 내용
20	2359280	테스트 제목	테스트 내용

인라인뷰의 내용

인라인뷰에서 필요한 부분만 추출

6. MyBatis와 스프링 페이징 처리

검색에 필요한 내용을 담는 Criteria클래스

- 검색에 사용되는 여러 종류의 데이터를 하나의 객체로 묶기 위한 용도

```
@Getter
@Setter
@ToString
public class Criteria {
    private int pageNum;
    private int amount;

    public Criteria() {
        this(1,10);
    }

    public Criteria(int pageNum, int amount) {
        this.pageNum = pageNum;
        this.amount = amount;
    }
}
```

MyBatis의 처리

```
public interface BoardMapper {  
    public List<BoardVO> getList();  
    public List<BoardVO> getListWithPaging(Criteria cri);  
    public void insert(BoardVO board);  
    public Integer insertSelectKey(BoardVO board);  
    public BoardVO read(Long bno);  
    public int delete(Long bno);  
    public int update(BoardVO board);  
}
```

```
<select id="getListWithPaging"  
    resultType="org.zerock.domain.BoardVO">  
    <![CDATA[  
        select  
            bno, title, content, writer, regdate, updatedate  
        from ( select /*+INDEX_DESC(tbl_board pk_board) */  
            rownum rn, bno, title, content, writer, regdate, updatedate  
        from  
            tbl_board  
        where rownum <= #{pageNum} * #{amount}  
        )  
        where rn > (#{pageNum} - 1) * #{amount}  
    ]]>  
</select>
```

페이징 처리 테스트

@Test

```
public void testGetList() {  
    // service.getList().forEach(board -> log.info(board));  
    service.getList(new Criteria(2, 10)).forEach(board -> Log.info(board));  
}
```

cri.setPageNum(1);
cri.setAmount(10);

cri.setPageNum(2);
cri.setAmount(10);

cri.setPageNum(3);
cri.setAmount(10);

2359299
2359298
2359297
2359296
2359295
2359294
2359293
2359292
2359291
2359290

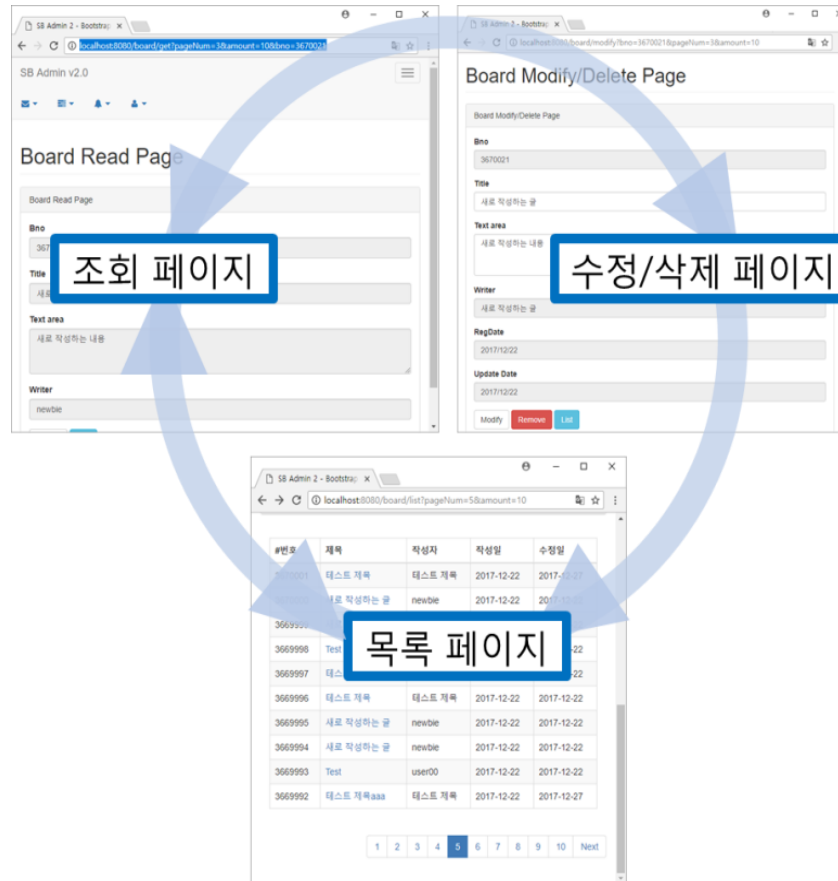
2359289
2359288
2359287
2359286
2359285
2359284
2359283
2359282
2359281
2359280

2359279
2359278
2359277
2359276
2359275
2359274
2359273
2359272
2359271
2359270

7. 화면 페이징 처리

JSP 처리

- BoardController/BoardService/BoardServiceImpl 처리
- list.jsp 처리



페이지 처리에 필요한 정보들

- 현재 페이지 번호(page)
- 이전과 다음으로 이동 가능한 링크의 표시 여부(prev, next)
- 화면에서 보여지는 페이지의 시작 번호와 끝 번호(startPage, endPage)

페이징의 끝 번호(endPage) 계산

```
this.endPage = (int)(Math.ceil(페이지번호 / 10.0)) * 10;
```

페이징의 시작 번호(startPage) 계산

```
this.startPage = this.endPage - 9;
```

total을 통한 endPage의 재계산

```
realEnd = (int) (Math.ceil((total * 1.0) / amount) );  
if(realEnd < this.endPage) {  
    this.endPage = realEnd;  
}
```


이전(prev) 계산

```
this.prev = this.startPage > 1;
```

다음(next) 계산

```
this.next = this.endPage < realEnd;
```

@Getter

@ToString

```
public class PageDTO {  
    private int startPage;  
    private int endPage;  
    private boolean prev, next;  
    private int total;  
    private Criteria cri;  
  
    public PageDTO(Criteria cri, int total) {  
        this.cri = cri;  
        this.total = total;  
        this.endPage = (int) (Math.ceil(cri.getPageNum() / 10.0)) * 10;  
        this.startPage = this.endPage - 9;  
        int realEnd = (int) (Math.ceil((total * 1.0) / cri.getAmount()));  
        if (realEnd <= this.endPage) {  
            this.endPage = realEnd;  
        }  
  
        this.prev = this.startPage > 1;  
        this.next = this.endPage < realEnd;  
    }  
}
```

JSP에서 화면 번호 출력

```
<div class='pull-right'>
    <ul class="pagination">
        <c:if test="${pageMaker.prev}">
            <li class="paginate_button previous"><a href="#">Previous</a>
        </li>
        </c:if>
        <c:forEach var="num" begin="${pageMaker.startPage}"
            end="${pageMaker.endPage}">
            <li class="paginate_button"><a href="#">${num}</a></li>
        </c:forEach>
        <c:if test="${pageMaker.next}">
            <li class="paginate_button next"><a href="#">Next</a></li>
        </c:if>
    </ul>
</div>
<!-- end Pagination -->
</div>
```

SB Admin 2 - Bootstrap x

localhost:8080/board/list?pageNum=5

Tables

Board List Page Register New Board

#번호	제목	작성자	작성일	수정일
2359259	새로 작성하는 글 select key	newbie	2018-04-03	2018-04-03
2359258	수정된 제목	user00	2018-04-03	2018-04-03
2359257	테스트 제목	user00	2018-04-03	2018-04-03
2359256	새로 작성하는 글	newbie	2018-04-03	2018-04-03
2359255	수정테스트	Test	2018-04-03	2018-04-03
2359251	Test	Test	2018-04-03	2018-04-03
2359250	테스트 새글 제목	user00	2018-04-03	2018-04-03

<http://localhost:8080/board/list?pageNum=5>

1 2 3 4 5 6 7 8 9 10 Next

SB Admin 2 - Bootstrap x

localhost:8080/board/list?pageNum=5&amount=20

2359214	테스트 게시물입니다.	user00	2018-04-03	2018-04-03
2359213	테스트 게시물입니다.	user00	2018-04-03	2018-04-03
2359212	테스트 게시물입니다.	user00	2018-04-03	2018-04-03
2359211	테스트 게시물입니다.	user00	2018-04-03	2018-04-03
2359210	수정된 테스트 게시물입니다.	user00	2018-04-03	2018-04-03
2359209	□ ◁ ○ ▷ ◁ ○ ▷	□ ◁ ○ ▷ ◁ ○ ▷	2018-04-03	2018-04-03
2359208	Test	Test	2018-04-03	2018-04-03
2359207	테스트 제목	user00	2018-04-03	2018-04-03
2359206	테스트 제목	user00	2018-04-03	2018-04-03
2359205	테스트 제목	user00	2018-04-03	2018-04-03
2359204	새로 작성하는 글 select key	newbie	2018-04-03	2018-04-03
2359203	수정된 제목	user00	2018-04-03	2018-04-03
2359202	테스트 제목	user00	2018-04-03	2018-04-03
2359201	테스트 제목	user00	2018-04-03	2018-04-03
2359200	테스트 제목	user00	2018-04-03	2018-04-03

<http://localhost:8080/board/list?pageNum=5&amount=20>

1 2 3 4 5 6 7

페이지 번호 이벤트 처리

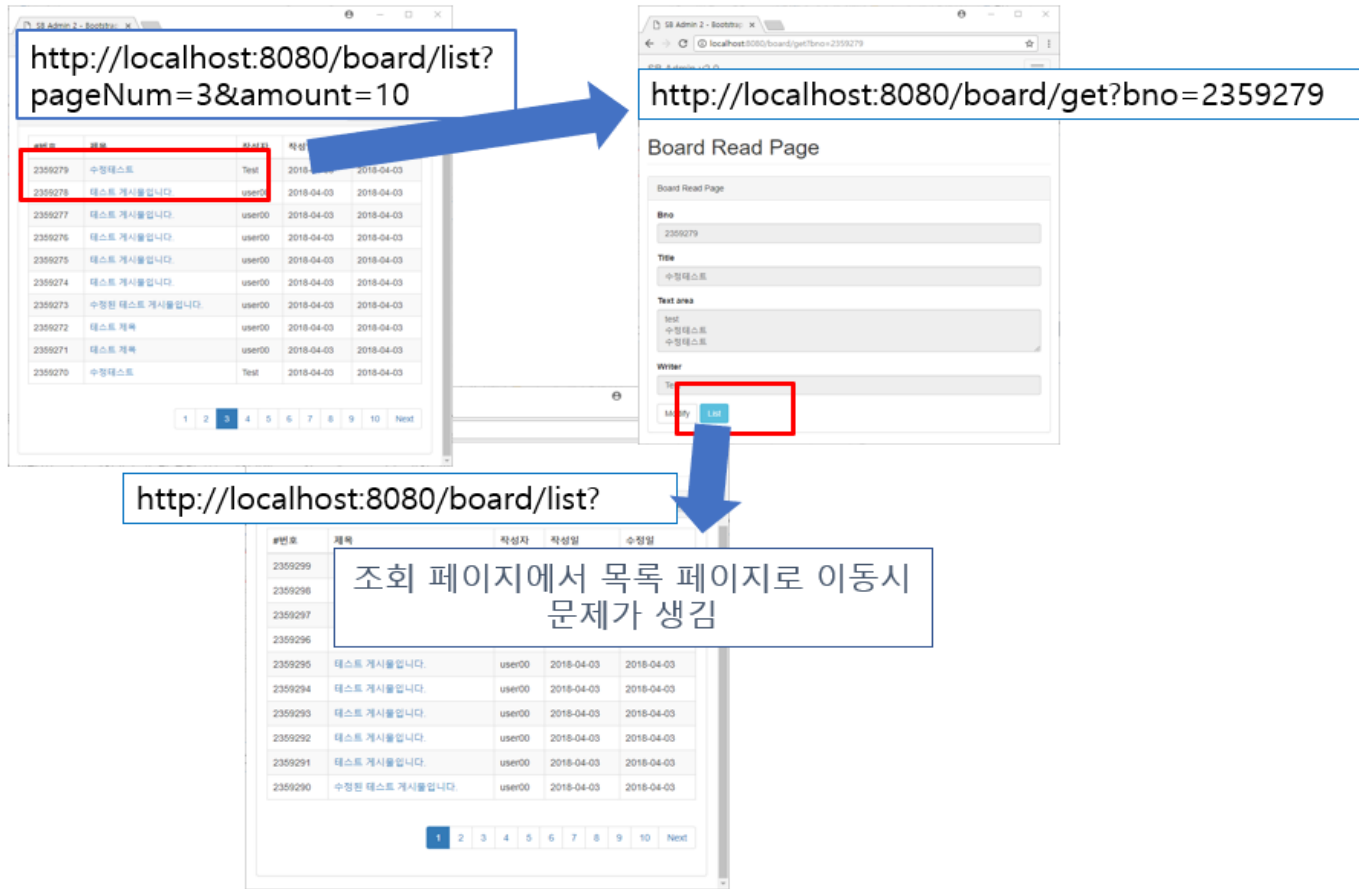
- 페이지번호의 링크는 페이지 번호만 가지도록 하고
- 별도의 <form>태그와 이벤트를 통해서 이동

```
<form id='actionForm' action="/board/list" method='get'>  
  <input type='hidden' name='pageNum' value = '${pageMaker.cri.pageNum}'>  
  <input type='hidden' name='amount' value = '${pageMaker.cri.amount}'>  
</form>
```

```
var actionForm = $("#actionForm");  
$(".paginate_button a").on("click", function(e) {  
  e.preventDefault(); //기본 동작 제한  
  console.log( ' click ' );  
  //<form>태그의 내용 변경 후 submit  
  actionForm.find("input[name='pageNum']").val($(this).attr("href"));  
  actionForm.submit();  
});
```

조회 페이지로 이동

- 조회 페이지 이동시 현재 페이지 번호를 같이 전달하는 방식으로 처리되어야 함



```

<td>
  <a class='move' href='<c:out value="\${board.bno}"/>'>
    <c:out value="\${board.title}" /></a>
</td>

```

2359291	테스트 게시물입니다.
2359290	수정된 테스트 게시물입니다.

1 2 3

localhost:8080/board/2359291

list.jsp 게시물 조회를 위한 이벤트 처리 추가

```

$(".move").on("click", function(e){
  e.preventDefault();
  actionForm.append("<input type='hidden' name='bno' value='"+ $(this).attr("href")+ "'>");
  actionForm.attr("action", "/board/get");
  actionForm.submit();
});

```

http://localhost:8080/board/list?pageNum=2&amount=10

Board List Page

#번호	제목	작성자	작성일	수정일
2359289	테스트 제목	user00	2018-04-03	2018-04-03
2359288	Test	Test	2018-04-03	2018-04-03
2359287	테스트 제목 제목	user00	2018-04-03	2018-04-03
2359286	새로 작성하는 글	newbie	2018-04-03	2018-04-03
2359285	새로 작성하는 글 select key	newbie	2018-04-03	2018-04-03
2359284	수정된 제목	user00	2018-04-03	2018-04-03
2359283	테스트 제목	user00	2018-04-03	2018-04-03
2359282	테스트 제목	user00	2018-04-03	2018-04-03
2359281	테스트 제목	user00	2018-04-03	2018-04-03
2359280	테스트 제목	user00	2018-04-03	2018-04-03

1 2 3 4 5 6 7 8 9 10 Next

http://localhost:8080/board/get?pageNum=2&amount=10&bno=2359285

Board Read Page

Board Read Page

Bno
2359285

Title
새로 작성하는 글 select key

Text area
새로 작성하는 내용 select key

Writer
newbie

Modify List

조회 페이지에서 목록페이지로

- 전달받은 페이지 번호를 이용해서 원래 페이지로 이동

views/board/get.jsp의 일부

```
<form id='operForm' action="/board/modify" method="get">
  <input type='hidden' id='bno' name='bno' value='<c:out value="\${board.bno}"/>'>
  <input type='hidden' name='pageNum' value='<c:out value="\${cri.pageNum}"/>'>
  <input type='hidden' name='amount' value='<c:out value="\${cri.amount}"/>'>
</form>
```

The left screenshot shows the 'Board Detail' page at `http://localhost:8080/board/get?pageNum=3&amount=10&bno=2359270`. It contains a form with fields for Bno, Title, Text area, and Writer. At the bottom, there are 'Modify' and 'List' buttons. The 'List' button is highlighted with a red box.

The right screenshot shows the 'Board List' page at `http://localhost:8080/board/list?pageNum=3&amount=10`. It displays a table of board entries. A blue arrow points from the 'List' button in the left screenshot to the URL in the right screenshot.

#번호	제목	작성자	작성일	수정일
2359275	테스트 게시물입니다.	user00	2018-04-03	2018-04-03
2359277	테스트 게시물입니다.	user00	2018-04-03	2018-04-03
2359276	테스트 게시물입니다.	user00	2018-04-03	2018-04-03
2359275	테스트 게시물입니다.	user00	2018-04-03	2018-04-03
2359274	테스트 게시물입니다.	user00	2018-04-03	2018-04-03
2359273	수정된 테스트 게시물입니다.	user00	2018-04-03	2018-04-03
2359272	테스트 제목	user00	2018-04-03	2018-04-03
2359271	테스트 제목	user00	2018-04-03	2018-04-03
2359270	수정테스트	Test	2018-04-03	2018-04-03

수정/삭제

- 수정/삭제 페이지에서도 목록으로 이동할 수 있도록 전달

http://localhost:8080/board/get?pageNum=2&amount=10&bno=2359285

Board Register

Board Register

Bno
3342327

Title
새로 작성하는 글

Text area
새로 작성하는 내용

Writer
newbie

Modify List

http://localhost:8080/board/modify?bno=2359285&pageNum=2&amount=10

Board Modify Page

Board Modify Page

Bno
2359285

Title
새로 작성하는 글 select key

Text area
새로 작성하는 내용 select key

Writer
newbie

Modify Remove List

수정/삭제 후 이동

```
@PostMapping("/remove")
public String remove(@RequestParam("bno") Long bno,
    @ModelAttribute("cri") Criteria cri, RedirectAttributes rttr) {
    Log.info("remove..." + bno);
    if (service.remove(bno)) {
        rttr.addFlashAttribute("result", "success");
    }
    rttr.addAttribute("pageNum", cri.getPageNum());
    rttr.addAttribute("amount", cri.getAmount());
    return "redirect:/board/list";
}
```

http://localhost:8080/boards/list?pageNum=5&amount=10

http://localhost:8080/board/get?pageNum=5&amount=10&bno=2359251

http://localhost:8080/board/modify?bno=2359251&pageNum=5&amount=10

번호	제목	작성자	작성일	수정일
2359257	테스트 제목	user00	2018-04-03	2018-04-03
2359256	테스트 제목	user00	2018-04-03	2018-04-03
2359255	테스트 제목	user00	2018-04-03	2018-04-03
2359254	테스트 제목	user00	2018-04-03	2018-04-03
2359253	새로 작성하는 글	newbie	2018-04-03	2018-04-03
2359252	수정테스트	Test	2018-04-03	2018-04-03
2359251	Test	Test	2018-04-03	2018-04-03
2359250	테스트 새글 제목	user00	2018-04-03	2018-04-03

Board Read Page

Bno: 2359251

Title: Test

Text area: Test

Writer: Test

Board Modify Page

Bno: 2359251

Title: Test

Text area: Test

Writer: Test

Modify Remove List

게시물의 숫자 처리

```
public interface BoardMapper {
```

...생략...

```
public int getTotalCount(Criteria cri);
```

```
}
```

```
<select id="getTotalCount" resultType="int">
  select count(*) from tbl_board where bno > 0
</select>
```

BoardController 클래스의 일부

```
@GetMapping("/list")
```

```
public void list(Criteria cri, Model model) {
```

```
    log.info("list: " + cri);
```

```
    model.addAttribute("list", service.getList(cri));
```

```
    //model.addAttribute("pageMaker", new PageDTO(cri, 123));
```

```
    int total = service.getTotal(cri);
```

```
    log.info("total: "+ total);
```

```
    model.addAttribute("pageMaker", new PageDTO(cri, total));
```

```
}
```

8. 검색 처리

검색의 유형

- 제목, 내용, 작성자와 같은 단일 항목
- 제목 + 내용, 제목 + 작성자와 같은 복합 항목
- 검색 항목에 따라서 매번 다른 SQL이 처리될 필요가 있는 상황
- MyBatis의 동적쿼리기능을 이용해서 처리
 - <http://www.mybatis.org/mybatis-3/ko/dynamic-sql.html>

MyBatis의 동적 태그들

- if
 - if 는 test라는 속성과 함께 특정한 조건이 true가 되었을 때 포함된 SQL을 사용하고자 할 때 작성
- choose (when, otherwise)
 - if와 달리 choose는 여러 상황들 중 하나의 상황에서만 동작
- trim (where, set)
 - trim, where, set은 단독으로 사용되지 않고, <if>, <choose>와 같은 태그들을 내포하여 SQL들을 연결해 주고, 앞 뒤에 필요한 구문들(AND, OR, WHERE 등)을 추가하거나 생략하는 역할
- foreach
 - foreach는 List, 배열, 맵 등을 이용해서 루프를 처리

foreach태그

```
Map<String, String> map = new HashMap<>();  
map.put("T", "TTTT");  
map.put("C", "CCCC");
```

```
select * from tbl_board  
<trim prefix="where (" suffix=)" prefixOverrides="OR" >  
  <foreach item="val" index="key" collection="map">  
    <trim prefix="OR" >  
      <if test="key == 'C'.toString()">  
        content = #{val}  
      </if>  
      <if test="key == 'T'.toString()">  
        title = #{val}  
      </if>  
      <if test="key == 'W'.toString()">  
        writer = #{val}  
      </if>  
    </trim>  
  </foreach>  
</trim>
```

검색조건처리를 위한 Criteria클래스 변경

- 검색항목(type)과 검색 키워드(keyword)추가

```
public class Criteria {  
  
    private int pageNum;  
    private int amount;  
  
    private String type;  
    private String keyword;  
  
}
```

BoardMapper.xml의 변경

```
<select id="getListWithPaging" resultType="org.zerock.domain.BoardVO">
...생략..
    <trim prefix="(" suffix=")" AND " prefixOverrides="OR ">
        <foreach item='type' collection="typeArr">
            <trim prefix="OR ">
                <choose>
                    <when test="type == 'T'.toString()">
                        title like '% ' ||#{keyword}|| ' % '
                    </when>
                    <when test="type == 'C'.toString()">
                        content like ' % ' ||#{keyword}|| ' % '
                    </when>
                    <when test="type == 'W'.toString()">
                        writer like ' % ' ||#{keyword}|| ' % '
                    </when>
                </choose>
            </trim>
        </foreach>
    </trim>
...생략..
</select>
```


테스트

@Test

```
public void testSearch() {  
    Criteria cri = new Criteria();  
    cri.setKeyword("새로");  
    cri.setType("TC");  
    List<BoardVO> list = mapper.getListWithPaging(cri);  
    list.forEach(board -> Log.info(board));  
}
```

```
Criteria cri = new Criteria();  
cri.setKeyword("키워드");  
cri.setType("T");
```

```
pk_board) */ rownum rn, bno, title, content, writer, regdate, updatedate from  
( title like '%' || '키워드' || '%' ) AND rownum <= 1 * 10 ) where rn > (1 -1) * 10
```

```
Criteria cri = new Criteria();  
cri.setKeyword("키워드");  
cri.setType("TC");
```

```
INFO : jdbc.sqltiming - select bno, title, content, writer, regdate, updatedate from ( select  
pk_board) */ rownum rn, bno, title, content, writer, regdate, updatedate from tbl_board where  
( title like '%' || '키워드' || '%' OR content like '%' || '키워드' || '%' ) AND rownum <= 1 * 10 ) where  
rn > (1 -1) * 10
```

```
Criteria cri = new Criteria();  
cri.setKeyword("키워드");  
cri.setType("TCW");
```

```
INFO : jdbc.sqltiming - select bno, title, content, writer, regdate, updatedate from ( select /*  
pk board) */ rownum rn, bno, title, content, writer, regdate, updatedate from tbl board where  
( title like '%' || '키워드' || '%' OR content like '%' || '키워드' || '%' OR writer like '%' || '키워드' || '%' )  
AND rownum <= 1 * 10 ) where rn > (1 -1) * 10
```

<sql>조각과 <include>

- 게시물의 검색과 게시물의 숫자 카운트에 공통으로 사용되므로 <sql>조각으로 분리하고, 필요한 곳에서 <include>하는 방식으로 적용

```
<sql id="criteria">
  <trim prefix="(" suffix=") AND " prefixOverrides="OR">
    ...생략...
  </trim>
</sql>
```

```
<select id="getListWithPaging"
resultType="org.zerock.domain.BoardVO">
  <![CDATA[
...생략...
  <include refid="criteria"></include>
  <![CDATA[
    rownum <= #{pageNum} * #{amount}
  )
  where rn > (#{pageNum} -1) * #{amount}
]]>
</select>
```

화면에서의 검색처리

```
<form id='searchForm' action="/board/list" method='get'>
  <select name='type'>
    <option value="">--</option>
    <option value="T">제목</option>
    <option value="C">내용</option>
    <option value="W">작성자</option>
    <option value="TC">제목 or 내용</option>
    <option value="TW">제목 or 작성자</option>
    <option value="TWC">제목 or 내용 or 작성자</option>
  </select>
  <input type='text' name='keyword' />
  <input type='hidden' name='pageNum' value='${pageMaker.cri.pageNum}'>
  <input type='hidden' name='amount' value='${pageMaker.cri.amount}'>
  <button class='btn btn-default'>Search</button>
</form>
```

The diagram illustrates the search process. On the left, a form with a dropdown menu (currently showing '제목'), a text input field (containing '작성하는'), and a 'Search' button is shown. A blue arrow points from this form to a browser window on the right. The browser window displays a table of search results. A text box in the center of the diagram contains the URL: `http://localhost:8080/board/list?type=T&keyword=%EC%9E%91%EC%84%B1%ED%95%98%EB%8A%94&pageNum=1&amount=10`. The table in the browser window has the following structure:

#번호	제목	작성자	작성일	수정일
3670052	새로 작성하는 글 select key	newbie	2017-12-30	2017-12-30
3670051	새로 작성하는 글	newbie	2017-12-30	2017-12-30
3670034	새로 작성하는 글	newbie	2017-12-22	2017-12-22
3670033	새로 작성하는 글	newbie	2017-12-22	2017-12-22
3670016	새로 작성하는 글	newbie	2017-12-22	2017-12-22

Below the table, there is a pagination bar with buttons for 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, and a 'Next' button.

검색이벤트 처리

```
$("#searchForm button").on("click", function(e){
    if(!searchForm.find("option:selected").val()){
        alert("검색종류를 선택하세요");
        return false;
    }

    if(!searchForm.find("input[name='keyword']").val()){
        alert("키워드를 입력하세요");
        return false;
    }

    searchForm.find("input[name='pageNum']").val("1");
    e.preventDefault();

    searchForm.submit();

});
```

검색후 <select> 처리

- 'Search'를 클릭하면 무조건 1 페이지로
- 이후에는 검색 항목 유지

검색 항목이 선택되지 않은 경우

localhost:8080/board/list?type=C&keyword=테스트&pageNum=1&amount=10

2359294	테스트	출처: localhost:8080	2018-04-03
2359293	테스트	검색종류를 선택하세요	2018-04-03
2359292	테스트		2018-04-03
2359291	테스트	게시물입니다.	2018-04-03
2359290	수정된 테스트 게시물입니다.	user00	2018-04-04

-- Search

1 2 3 4 5 6 7 8 9 10 Next

키워드를 입력하지 않은 경우

localhost:8080/board/list?type=C&keyword=테스트&pageNum=1&amount=10

2359294	테스트	출처: localhost:8080	2018-04-03
2359293	테스트	키워드를 입력하세요	2018-04-03
2359292	테스트		2018-04-03
2359291	테스트	게시물입니다.	2018-04-03
2359290	수정된 테스트 게시물입니다.	user00	2018-04-04

제목 Search

1 2 3 4 5 6 7 8 9 10 Next

검색시에는 무조건 1 페이지로

localhost:8080/board/list?type=C&keyword=테스트&pageNum=1&amount=10

2359162	Test	Test	2018-04-03	2018-04-03
2359145	Test	Test	2018-04-03	2018-04-03
2359118	Test	Test	2018-04-03	2018-04-03
2359114	Test	Test	2018-04-03	2018-04-03

제목 Test Search

1 2 3 4 5 6 7 8 9 10 Next

기타 검색 처리

- 검색후 조회,수정,삭제 페이지 이동시 검색 항목 유지
 - RedirectAttribute에 검색 관련 내용을 추가하는 방식
 - UriComponentsBuilder를 이용해서 Criteria에서 링크를 생성하는 방식

Criteria 클래스의 일부

```
public String getListLink() {  
  
    UriComponentsBuilder builder = UriComponentsBuilder.fromPath("")  
        .queryParam("pageNum", this.pageNum)  
        .queryParam("amount", this.getAmount())  
        .queryParam("type",this.getType())  
        .queryParam("keyword", this.getKeyword());  
  
    return builder.toUriString();  
  
}
```