

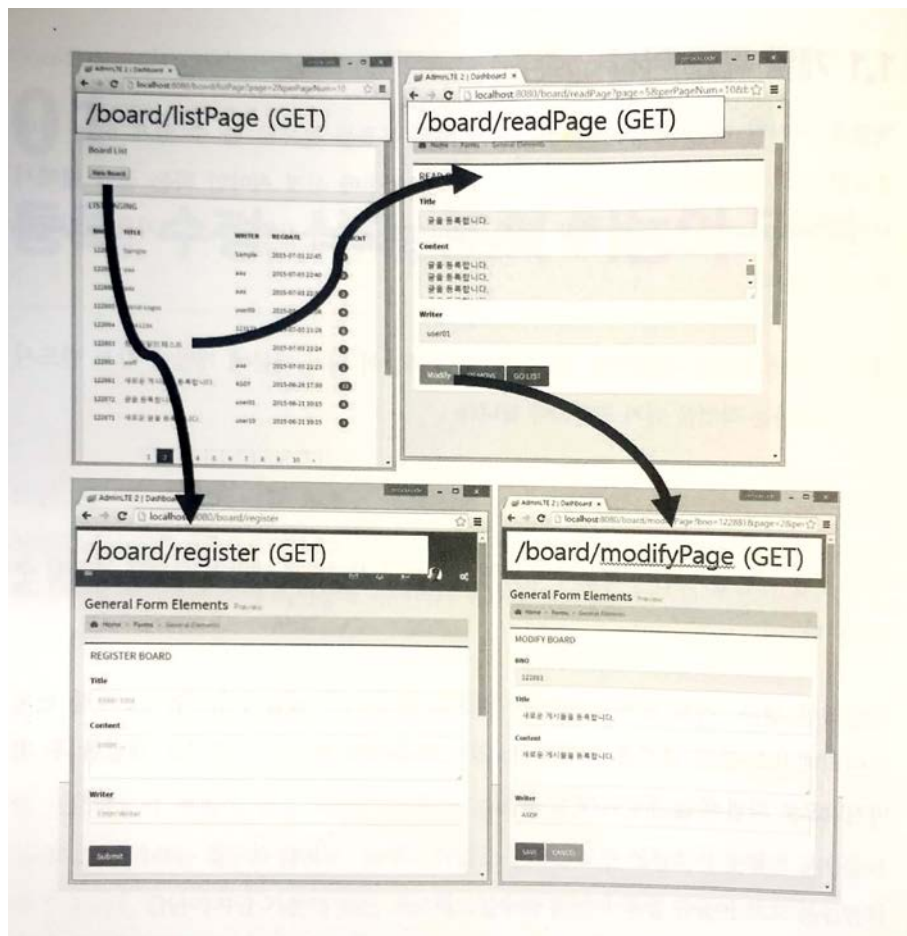


1장 기본적인 기능의 게시물 관리

1.1 등록, 수정, 삭제, 조회 기능의 구현

1.1.1 개발 목표 인식

- 개발을 시작할 때는 항상 현재 만들어야 하는 목표를 눈으로 볼 수 있게 하는 것(화면 설계)이 중요하다. 눈으로 결과를 보는 것은 1) 개발 이전에 신경 써야만 하는 점에 대해서 사전에 파악이 가능하고, 2) 개발 중에는 올바르게 개발이 진행되고 있는지를 확인하는 척도가 된다.



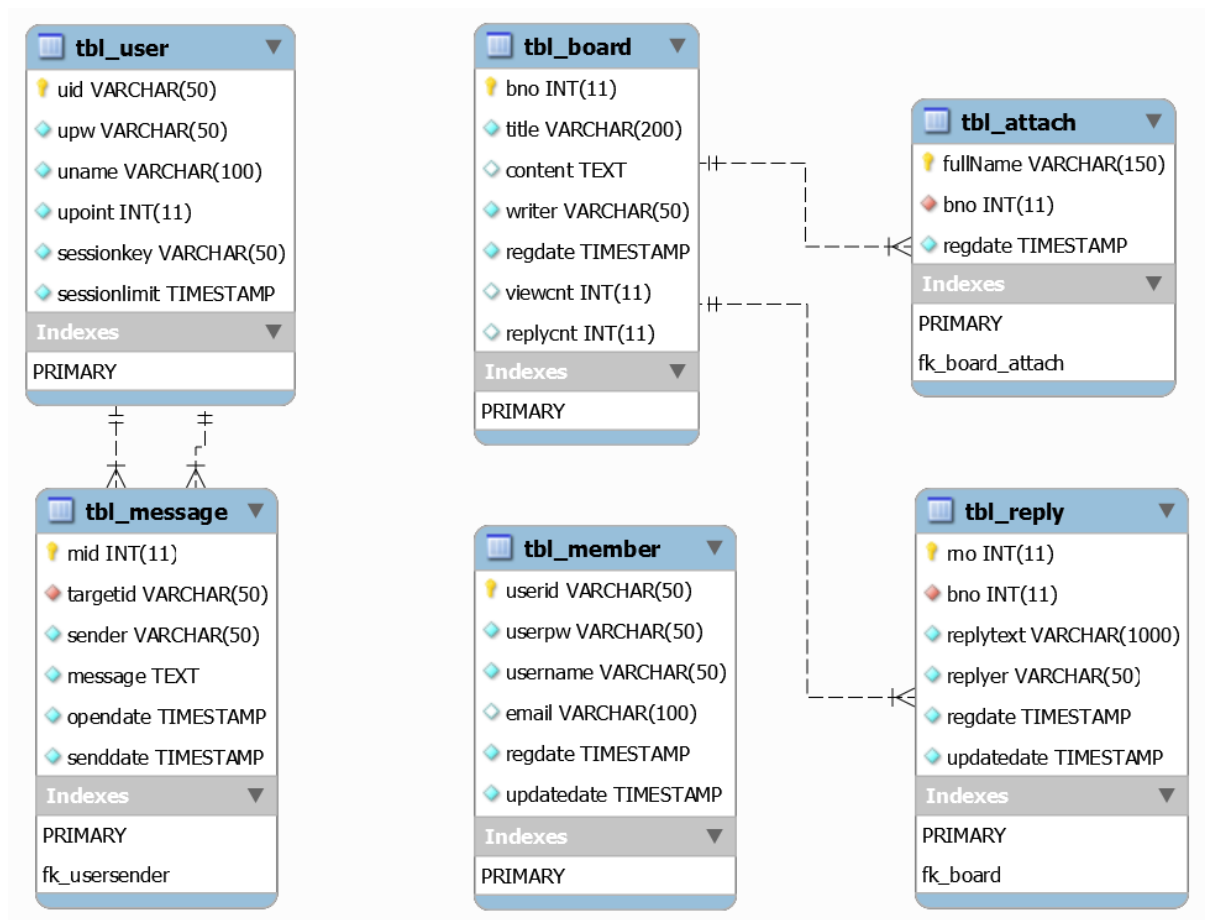
- 일반적인 경우, 개발 순서는 '뒤에서 앞으로'의 순서가 많다. 즉, 1) 가장 먼저 데이터베이스 관련 부분을 개발하고, 2) 컨트롤러와 비즈니스 영역을 개발한 후 3) 마지막으로 화면 쪽을 개발하게 된다.

1.1.2 각 영역에 따른 개발의 준비

영역	준비	설명
데이터베이스 관련	관련 스키마, 계정의 생성	개발에 사용할 스키마를 정의하고, 개발에 필요한 사용자 계정 등을 생성한다.
	테이블의 설계와 생성	실제 작업할 테이블을 설계하고 각 테이블의 관계를 ERD 등을 사용해서 그려둔다.

	더미(dummy) 데이터의 추가	테스트를 위한 가상의 의미 없는 데이터(dummy)를 추가해서 개발 시 결과를 확인할 때 사용한다.
웹 서버 환경 관련 (MVC)	패키지 구조 결정	개발에 사용할 패키지 이름이나 코드 규칙 등을 미리 지정한다.
	개발 역할 분담	팀원들 간의 개발 역할을 어떻게 할 것인지를 결정한다.
	테스트 방법의 수립	개발의 중간에 팀원들의 현재 상황을 올바르게 인식할 수 있는 테스트 방법을 서로 공유하고, 이를 지키도록 한다.
화면 관련	화면 방식의 결정	JSP 위주의 개발인지, HTML 위주와 JavaScript를 주로 사용하는 개발인지를 명확히 정리한다.
	절대 경로, 상대 경로의 결정	페이지에서 사용하는 링크의 처리를 하나로 통일해서 진행한다.

(1) 데이터베이스 관련 (ERD, MySQL)

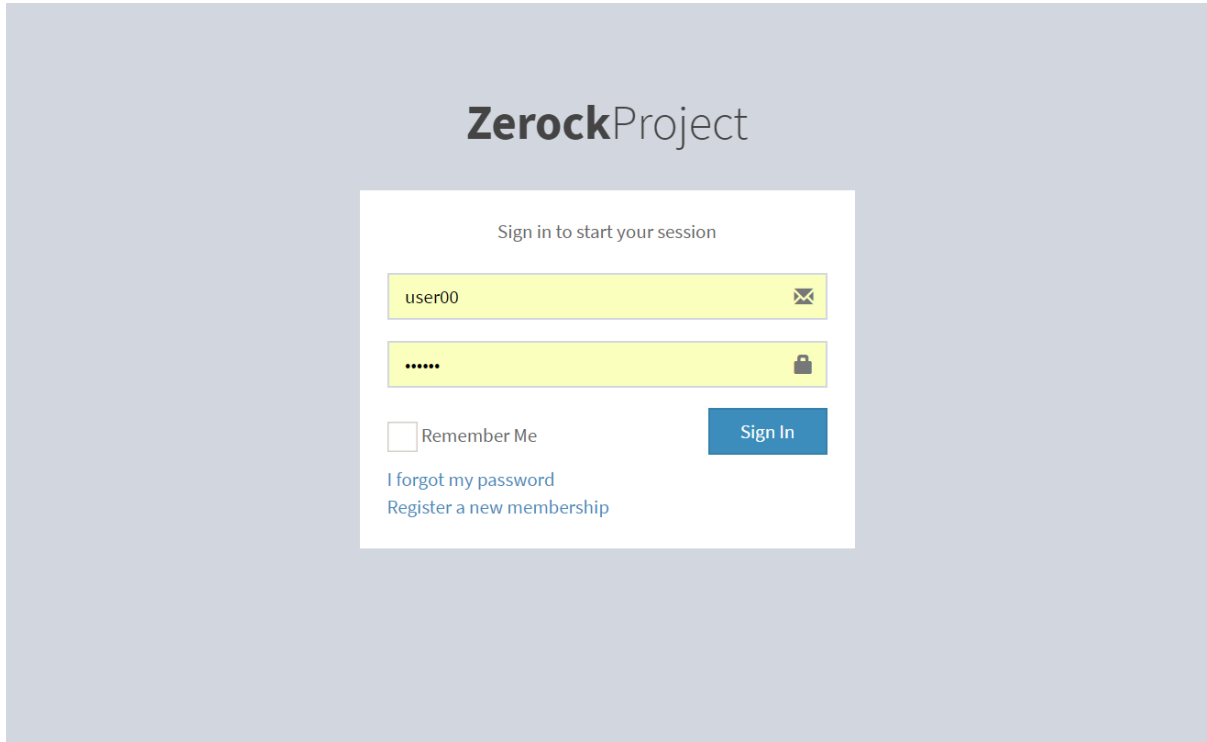


(2) 웹 서버 환경 관련 (Java, Spring, MyBatis)

- 웹 서버 환경은 다음과 같이 구성한다.
 - JDK 1.7 이상
 - Apache Tomcat 8.B.x
 - MySQL Community Server 5.6.25
 - Spring 4.x 이상의 버전
 - MyBatis 3.4.1
 - Bootstrap Theme (AdminLTE-2.4.x)

(3) 화면 관련 (HTML, Javascript, jQuery, JSP)

- 로그인 기능 : HttpSession을 이용한 로그인 처리와 쿠키를 이용하는 자동 로그인 방식을 구현한다.



The image shows a login form for a project named "ZerockProject". The form is centered on a light gray background. It has a title "ZerockProject" at the top. Below the title, the text "Sign in to start your session" is displayed. The form contains two input fields: the first is for the username, labeled "user00", and the second is for the password, labeled with six dots. To the right of the password field is a lock icon. Below the input fields, there is a checkbox labeled "Remember Me" and a blue button labeled "Sign In". At the bottom of the form, there are two links: "I forgot my password" and "Register a new membership".

- 로그아웃 기능 : HttpSession인 경우 login 정보를 삭제하고 쿠키의 유효시간을 변경하여 로그아웃 처리한다.

Zerock PROJECT

Alexander Pierce Online

Search...

MAIN NAVIGATION

- Dashboard
- Layout Options
- Widgets
- Charts
- UI Elements
- Forms
- Tables

General Form Elements Preview

Home > Forms > General Elements

Board List

LIST PAGING

BNO	TITLE	USER	DATE	COUNT
983523	aaa [0]	user00	2017-11-13 21:01	14
983522	aa [1]	aa	2017-11-10 17:09	6
983520	새로운 글을 넣습니다. [-1]	user00	2017-11-02 17:48	6
983519	test3 [2]	user01	2017-10-31 17:42	6
983395	새로운 글을 넣습니다. [1]	user00	2017-10-25 15:17	8
983392	aa [2]	aa	2017-10-25 15:17	6
983391	aa [0]	aa	2017-10-25 15:17	2

Alexander Pierce - Web Developer
Member since Nov. 2012

Followers Sales Friends

Profile Sign out

- 게시물 등록(쓰기) 기능 : 게시물 등록할 수 있는 화면을 만들어주고 데이터베이스까지 등록 되도록 작성한다.

Home > Forms > General Elements

REGISTER BOARD

Title

게시물 쓰기 테스트

Content

내용을 입력합니다

Writer

user00

File DROP Here

Submit

- 게시물 목록 보기 : 등록된 결과를 확인할 수 있도록 전체 목록 기능을 작성한다.

Zerock PROJECT

Alexander Pierce

Online

Search...

MAIN NAVIGATION

Dashboard

Layout Options

Widgets

Charts

UI Elements

Forms

General Elements

Advanced Elements

4

10

9

Alexander Pierce

General Form Elements Preview

Home > Forms > General Elements

Board List

Search

New Board

LIST PAGING

BNO	TITLE	WRITER	REGDATE	VIEWCNT
983524	게시물 등록 테스트 [0]	user00	2017-12-07 16:10	2
983523	aaa [0]	user00	2017-11-13 21:01	13
983522	aa [1]	aa	2017-11-10 17:09	4
983520	새로운 글을 넣습니다. [-1]	user00	2017-11-02 17:48	4
983519	test3 [2]	user01	2017-10-31 17:42	6

- 게시물 기획 기능 : 목록 작업이 완료되면 리스트의 특정 게시물을 선택해서 조회하는 화면을 작성한다.

Zerock PROJECT

Alexander Pierce

Online

Search...

MAIN NAVIGATION

Dashboard

Layout Options

Widgets

Charts

UI Elements

Forms

General Elements

Advanced Elements

Editors

Tables

4

10

9

Alexander Pierce

General Form Elements Preview

Home > Forms > General Elements

READ BOARD

Title

새로운 글을 넣습니다.

Content

새로운 글을 넣습니다.

Writer

user00

Modify

REMOVE

GO LIST

- 게시물 변경 기능 : 조회 화면에서는 'Modify' 버튼을 이용해서 수정 작업 페이지로 이동할 수 있도록 한다.

Online

Search...

MAIN NAVIGATION

Dashboard

Layout Options 4

Widgets new

Charts

UI Elements

Forms

General Elements

Advanced Elements

Editors

Tables

Calendar 3

Mailbox 12

Home > Forms > General Elements

MODIFY BOARD

BNO

983524

Title

게시물 등록 테스트

Content

테스트

Writer

user00

File DROP Here

- 게시물 삭제 기능 : 조회 화면에서 삭제를 선택해서 직접 삭제 처리할 수 있도록 한다.

Zerock PROJECT

Alexander Pierce

Online

Search...

MAIN NAVIGATION

Dashboard

Layout Options 4

Widgets new

Charts

UI Elements

Forms

General Elements

Advanced Elements

192.168.18.188:8080 내용:

처리가 완료되었습니다.

확인

Board List

Title

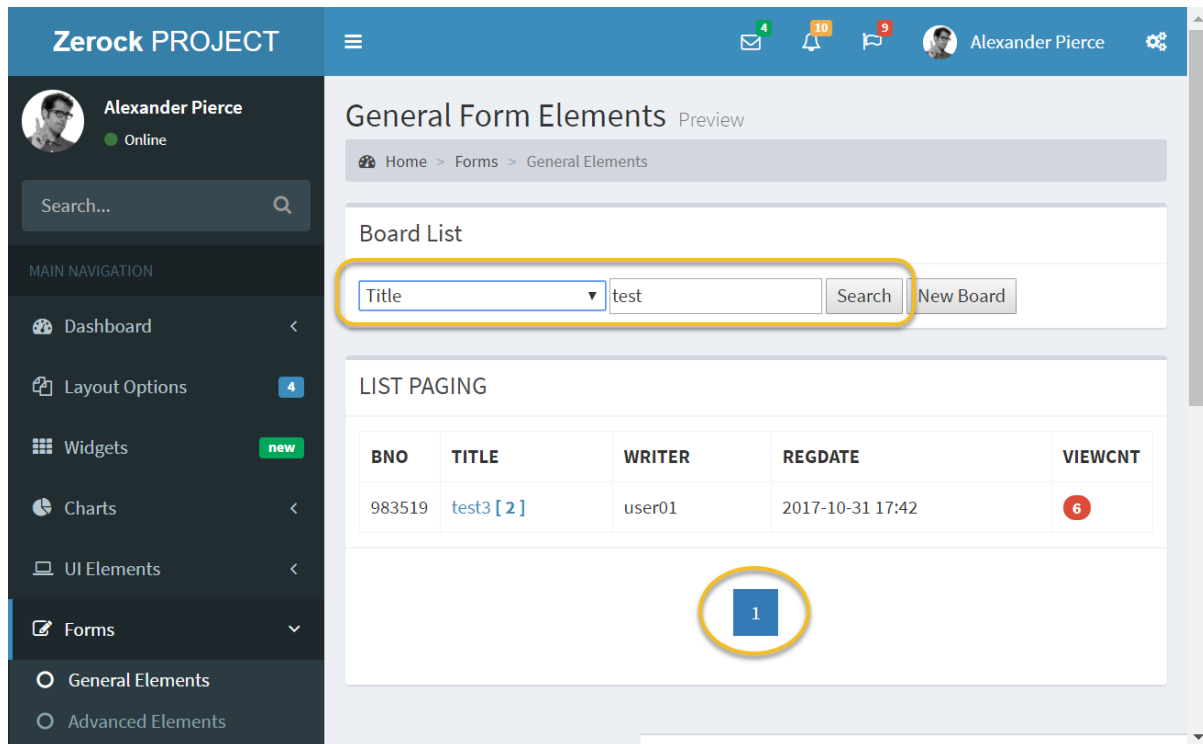
Search

New Board

LIST PAGING

BNO	TITLE	WRITER	REGDATE	VIEWCNT
983522	aa [1]	aa	2017-11-10 17:09	6
983520	새로운 글을 넣습니다. [-1]	user00	2017-11-02 17:48	6
983519	test3 [2]	user01	2017-10-31 17:42	6
983395	새로운 글을 넣습니다. [1]	user00	2017-10-25 15:17	8
983392	aa [2]	aa	2017-10-25 15:17	6

- 게시물 검색과 페이징 기능 : 게시물에 대한 검색 기능과 페이징 기능을 제공한다.



1.1.3 프로젝트의 생성

- 프로젝트는 'ex01Lab'이라는 이름으로 STS 상의 'Spring Project'를 이용한다.
- Please specify the top-level package: org.zerock.controller
- pom.xml 파일 수정

```
// pom.xml 파일 수정
...(생략)...
<properties>
    <java-version>1.8</java-version>
    <org.springframework-version>4.1.7.RELEASE</org.springframework-version>
    <org.aspectj-version>1.6.10</org.aspectj-version>
    <org.slf4j-version>1.6.6</org.slf4j-version>
</properties>
...(생략)...
```

- 프로젝트의 속성 > Project Facets > Java 버전 1.8 으로 수정
- pom.xml에 필요한 라이브러리를 아래와 같이 추가한다.

```
// pom.xml 파일 수정
...(생략)...

<!-- Test -->
<!-- <dependency> <groupId>junit</groupId> <artifactId>junit</artifactId>
    <version>4.7</version> <scope>test</scope> </dependency> -->

<!-- init begin -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
```

```

        <version>${org.springframework-version}</version>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.35</version>
    </dependency>
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis</artifactId>
        <version>3.2.8</version>
    </dependency>
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis-spring</artifactId>
        <version>1.2.2</version>
    </dependency>
    <dependency>
        <groupId>org.bgee.log4jdbc-log4j2</groupId>
        <artifactId>log4jdbc-log4j2-jdbc4</artifactId>
        <version>1.16</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>${org.springframework-version}</version>
    </dependency>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>3.1.0</version>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
    </dependency>
    <!-- init end -->
... (생략) ...

```

- 마지막으로 설정과 관련된 파일들을 추가한다. 파일의 위치는 '/src/main/resources' 폴더를 이용한다.

- log4jdbc.log4j2.properties
- logback.xml
- mybatis-config.xml

1.1.4 개발 전 준비 - 데이터베이스 관련

(1) DataSource의 등록

- DataSource는 MyBatis를 사용하기 위해서 반드시 root-context.xml을 사용해서 등록한다.

```

// src/main/webapp/WEB-INF/spring/root-context.xml 파일 수정
... (생략) ...
    <!-- init begin -->
    <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://127.0.0.1:3306/book_ex"></property>
    </bean>
    <!-- init end -->

```



```

        <property name="username" value="zerock"></property>
        <property name="password" value="zerock"></property>
    </bean>
    <!-- init end -->
... (생략) ...

```

(2) DataSource의 테스트

- DataSourceTest는 설정된 DataSource를 자동으로 주입받을 수 있게 한다.

[ex01/src/test/java/org/zerock/test/DataSourceTest.java]

```

01 package org.zerock.test;
02
03 import java.sql.Connection;
04
05 import javax.inject.Inject;
06 import javax.sql.DataSource;
07
08 import org.junit.Test;
09 import org.junit.runner.RunWith;
10 import org.springframework.test.context.ContextConfiguration;
11 import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
12
13 @RunWith(SpringJUnit4ClassRunner.class)
14 @ContextConfiguration(locations = {"file:src/main/webapp/WEB-INF/spring/**/*.xml"})
15 public class DataSourceTest {
16
17     @Inject
18     private DataSource ds;
19
20     @Test
21     public void testConnection() throws Exception {
22
23         try (Connection con = ds.getConnection()) {
24
25             System.out.println(con);
26
27         } catch (Exception e) {
28             e.printStackTrace();
29         }
30     }
31 }

```

```

// 실행 결과는 정상적으로 Connection 객체가 만들어지는 것을 확인할 수 있어야 한다.
INFO : org.springframework.web.servlet.handler.SimpleUrlHandlerMapping - Mapped URL path [/resources/**] onto
handler 'org.springframework.web.servlet.resource.ResourceHttpRequestHandler#0'
INFO : org.springframework.jdbc.datasource.DriverManagerDataSource - Loaded JDBC driver:
net.sf.log4jdbc.sql.jdbcapi.DriverSpy
INFO : jdbc.connection - 1. Connection opened
INFO : jdbc.audit - 1. Connection.new Connection returned
net.sf.log4jdbc.sql.jdbcapi.ConnectionSpy@494737
INFO : jdbc.connection - 1. Connection closed
INFO : jdbc.audit - 1. Connection.close() returned
INFO : org.springframework.context.support.GenericApplicationContext - Closing
org.springframework.context.support.GenericApplicationContext@1aee2f: startup date [Thu Nov 02 15:24:40 KST
2017]; root of context hierarchy

```

(3) 개발 패키지 구성

- org.zerock.domain : Value Object가 사용하는 패키지
- org.zerock.service : 서비스 인터페이스와 구현 클래스 패키지
- org.zerock.persistence : MyBatis의 DAO 패키지
- org.zerock.controller : 스프링 MVC의 컨트롤러 패키지
- resources/mappers 폴더 : MyBatis의 XML Mapper 위치

(4) 테이블의 생성 작업

```
create table tbl_board (
  bno INT NOT NULL AUTO_INCREMENT,
  title VARCHAR(200) NOT NULL,
  content TEXT NULL,
  writer VARCHAR(50) NOT NULL,
  regdate TIMESTAMP NOT NULL DEFAULT now(),
  viewcnt INT DEFAULT 0,
  PRIMARY KEY (bno)
);
```

(5) 테스트를 위한 SQL 준비

```
insert into tbl_board (title, content, writer)
values('제목입니다', '내용입니다', 'user00');

select * from tbl_board where bno = 1;

select * from tbl_board where bno >0
order by bno desc;

update tbl_board set title='수정된 제목' where bno = 1;

delete from tbl_board where bno = 1;
```

1.1.5 스프링의 UTF-8 처리 필터 등록

(1) web.xml

- 웹 애플리케이션을 제작하다 보면 한글 처리에서 고생을 하는 경우가 많은데, UTF-8이 사용되는 경우는 스프링에서 제공하는 필터를 등록해 주는 것이 좋다.

[ex01/src/main/webapp/WEB-INF/web.xml]

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
03         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
05 app_2_5.xsd">
06
07     <!-- The definition of the Root Spring Container shared by all Servlets and Filters -->
08     <context-param>
```

```

09         <param-name>contextConfigLocation</param-name>
10         <param-value>/WEB-INF/spring/root-context.xml</param-value>
11     </context-param>
12
13     <!-- Creates the Spring Container shared by all Servlets and Filters -->
14     <listener>
15         <listener-class>org.springframework.web.context.ContextLoaderListener</listener-
16 class>
17     </listener>
18
19     <!-- Processes application requests -->
20     <servlet>
21         <servlet-name>appServlet</servlet-name>
22         <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
23         <init-param>
24             <param-name>contextConfigLocation</param-name>
25             <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
26         </init-param>
27         <load-on-startup>1</load-on-startup>
28     </servlet>
29
30     <servlet-mapping>
31         <servlet-name>appServlet</servlet-name>
32         <url-pattern>/</url-pattern>
33     </servlet-mapping>
34
35     <filter>
36         <filter-name>encoding</filter-name>
37         <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
38         <init-param>
39             <param-name>encoding</param-name>
40             <param-value>UTF-8</param-value>
41         </init-param>
42     </filter>
43
44     <filter-mapping>
45         <filter-name>encoding</filter-name>
46         <url-pattern>/*</url-pattern>
47     </filter-mapping>
48
49 </web-app>

```

1.1.6 CSS, JavaScript 준비

- 프로젝트의 템플릿 적용: 'Admin LTE' 템플릿은 <https://almsaeedstudio.com/AdminLTE> 에서 제공하는 오픈 소스로 부트스트랩을 사용해서 작성된 무료 템플릿(bootstrap theme)이다.
- Context Path 변경: 프로젝트 Properties에서 'Web Project Settings > Context Path'를 '/'로 변경한다.
- 프로젝트 실행: <http://localhost:8080/>

1.1.7 작업 내역 정리

- 프로젝트 생성과 JDK 버전 변경
- 프로젝트에 필요한 라이브러리 추가
- 프로젝트에 필요한 설정 파일을 추가
- 개발에 필요한 패키지 생성
- 데이터베이스에 테이블 생성

- DataSource의 연결과 테스트 작업

1.2 영속(persistence) 계층, 비즈니스 계층

1.2.1 BoardVO의 작성

- 가장 먼저 하는 일은 테이블의 구조를 객체화시킬 때 사용하는 BoardVO 클래스를 작성하는 것으로 시작한다.
- org.zerock.domain 패키지 내에 BoardVO 클래스를 생성한다.

[ex01/src/main/java/org.zerock.domain/BoardVO.java]

```
01  package org.zerock.domain;
02
03  import java.util.Date;
04
05  public class BoardVO {
06
07      private Integer bno;
08      private String title;
09      private String content;
10      private String writer;
11      private Date regdate;
12      private int viewcnt;
13
14      public Integer getBno() {
15          return bno;
16      }
17      public void setBno(Integer bno) {
18          this.bno = bno;
19      }
20      public String getTitle() {
21          return title;
22      }
23      public void setTitle(String title) {
24          this.title = title;
25      }
26      public String getContent() {
27          return content;
28      }
29      public void setContent(String content) {
30          this.content = content;
31      }
32      public String getWriter() {
33          return writer;
34      }
35      public void setWriter(String writer) {
36          this.writer = writer;
37      }
38      public Date getRegdate() {
39          return regdate;
40      }
41      public void setRegdate(Date regdate) {
42          this.regdate = regdate;
43      }
44      public int getViewcnt() {
45          return viewcnt;
46      }
47      public void setViewcnt(int viewcnt) {
48          this.viewcnt = viewcnt;
49      }
49  }
```

```

50         @Override
51         public String toString() {
52             return "BoardVO [bno=" + bno + ", title=" + title + ", content="
53                 + content + ", writer=" + writer + ", regdate=" + regdate
54                 + ", viewcnt=" + viewcnt + "]\n";
55         }
56     }
57 }

```

1.2.2 DAO의 생성과 XML Mapper 작업

(1) XML 네임스페이스의 추가

- root-context.xml의 설정을 위해 xml 파일 내부에서 사용하는 태그들에 대한 설정을 위한 XML 네임스페이스를 추가한다.
 - beans
 - context
 - mybatis-spring

(2) BoardDAO의 생성

- org.zerock.persistence 패키지에 BoardDAO 인터페이스를 작성한다.

[ex01/src/main/java/org/zerock/persistence/BoardDAO.java]

```

01 package org.zerock.persistence;
02
03 import java.util.List;
04
05 import org.zerock.domain.BoardVO;
06 import org.zerock.domain.Criteria;
07 import org.zerock.domain.SearchCriteria;
08
09 public interface BoardDAO {
10
11     public void create(BoardVO vo) throws Exception;
12
13     public BoardVO read(Integer bno) throws Exception;
14
15     public void update(BoardVO vo) throws Exception;
16
17     public void delete(Integer bno) throws Exception;
18
19     public List<BoardVO> listAll() throws Exception;
20
21     public List<BoardVO> listPage(int page) throws Exception;
22
23     public List<BoardVO> listCriteria(Criteria cri) throws Exception;
24
25     public int countPaging(Criteria cri) throws Exception;
26
27     //use for dynamic sql
28
29     public List<BoardVO> listSearch(SearchCriteria cri) throws Exception;
30
31     public int listSearchCount(SearchCriteria cri) throws Exception;
32
33 }

```

(3) XML Mapper에서의 SQL 처리

[ex01/src/main/resources/mappers/boardMapper.xml]

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <!DOCTYPE mapper
03 PUBLIC "-//mybatis.org/DTD Mapper 3.0//EN"
04 "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
05
06 <mapper namespace="org.zerock.mapper.BoardMapper">
07
08   <insert id="create">
09     insert into tbl_board (title, content, writer)
10     values(#{title},#{content}, #{writer})
11   </insert>
12
13   <select id="read" resultType="org.zerock.domain.BoardVO">
14     select
15       bno, title, content, writer, regdate, viewcnt
16     from
17       tbl_board
18     where bno = #{bno}
19   </select>
20
21   <update id="update">
22     update tbl_board set title =#{title}, content =#{content}
23     where bno = #{bno}
24   </update>
25
26   <delete id="delete">
27     delete from tbl_board where bno = #{bno}
28   </delete>
29
30   <select id="listAll" resultType="org.zerock.domain.BoardVO">
31     <![CDATA[
32     select
33       bno, title, content, writer, regdate, viewcnt
34     from
35       tbl_board
36     where bno > 0
37     order by bno desc, regdate desc
38   ]]>
39   </select>
40
41   <select id="listPage" resultType="BoardVO">
42     <![CDATA[
43     select
44       bno, title, content, writer, regdate, viewcnt
45     from
46       tbl_board
47     where bno > 0
48     order by bno desc, regdate desc
49     limit #{page}, 10
50   ]]>
51   </select>
52
53   <select id="listCriteria" resultType="BoardVO">
54     <![CDATA[
55     select
56       bno, title, content, writer, regdate, viewcnt
57     from
```

```

58     tbl_board
59     where bno > 0
60     order by bno desc, regdate desc
61     limit #{pageStart}, #{perPageNum}
62 ]]>
63 </select>
64
65
66
67 <select id="countPaging" resultType="int">
68 <![CDATA[
69     select
70         count(bno)
71     from
72         tbl_board
73     where
74         bno > 0
75 ]]>
76 </select>
77
78
79
80 <sql id="search">
81     <if test="searchType != null">
82         <if test="searchType = 't'.toString()">
83             and title like CONCAT('%', #{keyword}, '%')
84         </if>
85         <if test="searchType = 'c'.toString()">
86             and content like CONCAT('%', #{keyword}, '%')
87         </if>
88         <if test="searchType = 'w'.toString()">
89             and writer like CONCAT('%', #{keyword}, '%')
90         </if>
91         <if test="searchType = 'tc'.toString()">
92             and ( title like CONCAT('%', #{keyword}, '%') OR content like CONCAT('%', #{keyword}, '%'))
93         </if>
94         <if test="searchType = 'cw'.toString()">
95             and ( content like CONCAT('%', #{keyword}, '%') OR writer like CONCAT('%', #{keyword}, '%'))
96         </if>
97         <if test="searchType = 'tcw'.toString()">
98             and ( title like CONCAT('%', #{keyword}, '%')
99                 OR
100                 content like CONCAT('%', #{keyword}, '%')
101                 OR
102                 writer like CONCAT('%', #{keyword}, '%'))
103         </if>
104     </if>
105 </sql>
106
107 <select id="listSearch" resultType="BoardVO">
108 <![CDATA[
109     select *
110     from tbl_board
111     where bno > 0
112 ]]>
113
114     <include refid="search"></include>
115
116 <![CDATA[
117     order by bno desc
118     limit #{pageStart}, #{perPageNum}
119 ]]>
120 </select>
121
122 <select id="listSearchCount" resultType="int">

```

```

123 <![CDATA[
124     select count(bno)
125     from tbl_board
126     where bno > 0
127 ]]>
128     <include refid="search"></include>
129
130 </select>
131
132 </mapper>

```

(4) BoardDAO의 구현 클래스 BoardDAOImpl

[ex01/src/main/java/org/zerock/persistence/BoardDAOImpl.java]

```

01 package org.zerock.persistence;
02
03 import java.util.List;
04
05 import javax.inject.Inject;
06
07 import org.apache.ibatis.session.SqlSession;
08 import org.springframework.stereotype.Repository;
09 import org.zerock.domain.BoardVO;
10 import org.zerock.domain.Criteria;
11 import org.zerock.domain.SearchCriteria;
12
13 @Repository
14 public class BoardDAOImpl implements BoardDAO {
15
16     @Inject
17     private SqlSession session;
18
19     private static String namespace = "org.zerock.mapper.BoardMapper";
20
21     @Override
22     public void create(BoardVO vo) throws Exception {
23         session.insert(namespace + ".create", vo);
24     }
25
26     @Override
27     public BoardVO read(Integer bno) throws Exception {
28         return session.selectOne(namespace + ".read", bno);
29     }
30
31     @Override
32     public void update(BoardVO vo) throws Exception {
33         session.update(namespace + ".update", vo);
34     }
35
36     @Override
37     public void delete(Integer bno) throws Exception {
38         session.delete(namespace + ".delete", bno);
39     }
40
41     @Override
42     public List<BoardVO> listAll() throws Exception {
43         return session.selectList(namespace + ".listAll");
44     }
45
46     @Override
47     public List<BoardVO> listPage(int page) throws Exception {

```



```

48
49     if (page <= 0) {
50         page = 1;
51     }
52
53     page = (page - 1) * 10;
54
55     return session.selectList(namespace + ".listPage", page);
56 }
57
58 @Override
59 public List<BoardVO> listCriteria(Criteria cri) throws Exception {
60
61     return session.selectList(namespace + ".listCriteria", cri);
62 }
63
64 @Override
65 public int countPaging(Criteria cri) throws Exception {
66
67     return session.selectOne(namespace + ".countPaging", cri);
68 }
69
70 @Override
71 public List<BoardVO> listSearch(SearchCriteria cri) throws Exception {
72
73     return session.selectList(namespace + ".listSearch", cri);
74 }
75
76 @Override
77 public int listSearchCount(SearchCriteria cri) throws Exception {
78
79     return session.selectOne(namespace + ".listSearchCount", cri);
80 }
81
82
83 }

```

(5) SessionFactory, SqlSessionFactory의 추가

- MyBatis의 SqlSessionFactory, SqlSessionTemplate을 등록한다.
- 설정의 마지막에 org.zerock.persistence 패키지를 자동으로 인식하기 위한 <context:component-scan>이 사용된다.

[ex01/src/main/webapp/WEB-INF/spring/root-context.xml]

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <beans xmlns="http://www.springframework.org/schema/beans"
03        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04        xmlns:context="http://www.springframework.org/schema/context"
05        xmlns:mybatis-spring="http://mybatis.org/schema/mybatis-spring"
06        xsi:schemaLocation="http://mybatis.org/schema/mybatis-spring
07        http://mybatis.org/schema/mybatis-spring-1.2.xsd
08        http://www.springframework.org/schema/beans
09        http://www.springframework.org/schema/beans/spring-beans.xsd
10        http://www.springframework.org/schema/context
11        http://www.springframework.org/schema/context/spring-context-4.1.xsd">
12
13 <!-- Root Context: defines shared resources visible to all other web components -->
14 <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
15     <property name="driverClassName" value="net.sf.log4jdbc.sql.jdbcapi.DriverSpy"></property>
16     <property name="url" value="jdbc:log4jdbc:mysql://127.0.0.1:3306/book_ex"></property>

```

```

17         <property name="username" value="zerock"></property>
18         <property name="password" value="zerock"></property>
19     </bean>
20
21     <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
22         <property name="dataSource" ref="dataSource" />
23         <property name="configLocation" value="classpath:/mybatis-config.xml"></property>
24         <property name="mapperLocations" value="classpath:mappers/**/*.xml"></property>
25     </bean>
26
27     <bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate" destroy-method="clearCache">
28         <constructor-arg name="sqlSessionFactory" ref="sqlSessionFactory"></constructor-arg>
29     </bean>
30
31     <context:component-scan base-package="org.zerock.persistence"></context:component-scan>
32     <context:component-scan base-package="org.zerock.service"></context:component-scan>
33
34 </beans>

```

(6) BoardDAO의 테스트

- 작성된 BoardDAO의 테스트 작업은 junit을 사용해서 진행한다.

[ex01/src/test/java/org/zerock/test/BoardDAOTest.java]

```

01 package org.zerock.test;
02
03 import java.util.List;
04
05 import javax.inject.Inject;
06
07 import org.junit.Test;
08 import org.junit.runner.RunWith;
09 import org.slf4j.Logger;
10 import org.slf4j.LoggerFactory;
11 import org.springframework.test.context.ContextConfiguration;
12 import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
13 import org.springframework.web.util.UriComponents;
14 import org.springframework.web.util.UriComponentsBuilder;
15 import org.zerock.domain.BoardVO;
16 import org.zerock.domain.Criteria;
17 import org.zerock.domain.SearchCriteria;
18 import org.zerock.persistence.BoardDAO;
19
20 @RunWith(SpringJUnit4ClassRunner.class)
21 @ContextConfiguration(locations = { "file:src/main/webapp/WEB-INF/spring/**/*.xml" })
22 public class BoardDAOTest {
23
24     @Inject
25     private BoardDAO dao;
26
27     private static Logger logger = LoggerFactory.getLogger(BoardDAOTest.class);
28
29     @Test
30     public void testCreate() throws Exception {
31
32         BoardVO board = new BoardVO();
33         board.setTitle("새로운 글을 넣습니다. ");
34         board.setContent("새로운 글을 넣습니다. ");
35         board.setWriter("user00");
36         dao.create(board);
37     }

```

```

38
39     @Test
40     public void testRead() throws Exception {
41
42         logger.info(dao.read(1).toString());
43     }
44
45     @Test
46     public void testUpdate() throws Exception {
47
48         BoardVO board = new BoardVO();
49         board.setBno(1);
50         board.setTitle("수정된 글입니다.");
51         board.setContent("수정 테스트 ");
52         dao.update(board);
53     }
54
55     @Test
56     public void testDelete() throws Exception {
57
58         dao.delete(1);
59     }
60
61     @Test
62     public void testListAll() throws Exception {
63
64         logger.info(dao.listAll().toString());
65     }
66
67     @Test
68     public void testListPage() throws Exception {
69
70         int page = 3;
71
72         List<BoardVO> list = dao.listPage(page);
73
74         for (BoardVO boardVO : list) {
75             logger.info(boardVO.getBno() + ":" + boardVO.getTitle());
76         }
77     }
78
79     @Test
80     public void testListCriteria() throws Exception {
81
82         Criteria cri = new Criteria();
83         cri.setPage(2);
84         cri.setPerPageNum(20);
85
86         List<BoardVO> list = dao.listCriteria(cri);
87
88         for (BoardVO boardVO : list) {
89             logger.info(boardVO.getBno() + ":" + boardVO.getTitle());
90         }
91     }
92
93     @Test
94     public void testURI() throws Exception {
95
96         UriComponents uriComponents =
97             UriComponentsBuilder.newInstance().path("/board/read").queryParam("bno", 12)
98                 .queryParam("perPageNum", 20).build();
99
100         logger.info("/board/read?bno=12&perPageNum=20");
101         logger.info(uriComponents.toString());
102

```

```

103
104     }
105
106     @Test
107     public void testURI2() throws Exception {
108
109         UriComponents uriComponents =
110 UriComponentsBuilder.newInstance().path("/{module}/{page}").queryParam("bno", 12)
111                                     .queryParam("perPageNum", 20).build().expand("board",
112 "read").encode();
113
114         logger.info("/board/read?bno=12&perPageNum=20");
115         logger.info(uriComponents.toString());
116     }
117
118     @Test
119     public void testDynamic1() throws Exception {
120
121         SearchCriteria cri = new SearchCriteria();
122         cri.setPage(1);
123         cri.setKeyword("글");
124         cri.setSearchType("t");
125
126         logger.info("=====");
127
128         List<BoardVO> list = dao.listSearch(cri);
129
130         for (BoardVO boardVO : list) {
131             logger.info(boardVO.getBno() + ": " + boardVO.getTitle());
132         }
133
134         logger.info("=====");
135
136         logger.info("COUNT: " + dao.listSearchCount(cri));
137     }
138
139 }

```

(7) <typeAliases>의 적용

- 이 요소들을 이용하면 매번 parameterType이나 resultType에 사용하는 클래스의 이름을 'org.zerock.domain'은 생략한 채로 표현하는 것이 가능하다.

[ex01/src/main/resources/mybatis-config.xml]

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <!DOCTYPE configuration
03     PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
04     "http://mybatis.org/dtd/mybatis-3-config.dtd">
05 <configuration>
06
07     <typeAliases>
08         <package name="org.zerock.domain"/>
09     </typeAliases>
10
11
12 </configuration>

```

1.2.3 비즈니스 계층

(1) 비즈니스 계층의 구현

- 비즈니스 영역을 구분해서 개발하면 개발의 양이 늘어나는 것은 사실이다. 그럼에도, 몇 가지 이유가 있기 때문에 굳이 계층을 분리해서 개발한다.
 - 비즈니스 계층은 고객마다 다른 부분을 처리할 수 있는 완충장치 역할을 한다.
 - 각 회사마다 다른 로직이나 규칙을 데이터베이스에 무관하게 처리할 수 있는 완충 영역으로 존재할 필요가 있다.
 - 컨트롤러와 같은 외부 호출이 영속 계층에 종속적인 상황을 막아준다.
 - 만일 컨트롤러가 직접 영속 계층의 데이터베이스를 이용하게 되면 트랜잭션의 처리나 예외의 처리 등 모든 로직이 컨트롤러로 집중된다. 비즈니스 계층은 컨트롤러로 하여금 처리해야 하는 일을 분업하게 만들어 준다.
- root-context.xml의 설정
 - root-context.xml에 <context>를 사용해서 패키지를 자동으로 인식할 수 있도록 한다.

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

■ BoardService / BoardServiceImpl의 작성

[ex01/src/main/java/org/zerock/service/BoardService.java]

```
01 package org.zerock.service;
02
03 import java.util.List;
04
05 import org.zerock.domain.BoardVO;
06 import org.zerock.domain.Criteria;
07 import org.zerock.domain.SearchCriteria;
08
09 public interface BoardService {
10
11     public void regist(BoardVO board) throws Exception;
12
13     public BoardVO read(Integer bno) throws Exception;
14
15     public void modify(BoardVO board) throws Exception;
16
17     public void remove(Integer bno) throws Exception;
18
19     public List<BoardVO> listAll() throws Exception;
20
21     public List<BoardVO> listCriteria(Criteria cri) throws Exception;
22
23     public int listCountCriteria(Criteria cri) throws Exception;
24
25     public List<BoardVO> listSearchCriteria(SearchCriteria cri)
26         throws Exception;
27
28     public int listSearchCount(SearchCriteria cri) throws Exception;
29
30 }
```

[ex01/src/main/java/org/zerock/service/BoardServiceImpl.java]

```
01 package org.zerock.service;
```

```
02
03 import java.util.List;
04
05 import javax.inject.Inject;
06
07 import org.springframework.stereotype.Service;
08 import org.zerock.domain.BoardVO;
09 import org.zerock.domain.Criteria;
10 import org.zerock.domain.SearchCriteria;
11 import org.zerock.persistence.BoardDAO;
12
13 @Service
14 public class BoardServiceImpl implements BoardService {
15
16     @Inject
17     private BoardDAO dao;
18
19     @Override
20     public void regist(BoardVO board) throws Exception {
21         dao.create(board);
22     }
23
24     @Override
25     public BoardVO read(Integer bno) throws Exception {
26         return dao.read(bno);
27     }
28
29     @Override
30     public void modify(BoardVO board) throws Exception {
31         dao.update(board);
32     }
33
34     @Override
35     public void remove(Integer bno) throws Exception {
36         dao.delete(bno);
37     }
38
39     @Override
40     public List<BoardVO> listAll() throws Exception {
41         return dao.listAll();
42     }
43
44     @Override
45     public List<BoardVO> listCriteria(Criteria cri) throws Exception {
46
47         return dao.listCriteria(cri);
48     }
49
50     @Override
51     public int listCountCriteria(Criteria cri) throws Exception {
52
53         return dao.countPaging(cri);
54     }
55
56     @Override
57     public List<BoardVO> listSearchCriteria(SearchCriteria cri) throws Exception {
58
59         return dao.listSearch(cri);
60     }
61
62     @Override
63     public int listSearchCount(SearchCriteria cri) throws Exception {
64
65         return dao.listSearchCount(cri);
66     }
67 }
```

```
67
68 }
```

(2) 비즈니스 계층의 테스트

- 비즈니스 계층의 서비스는 현재의 경우 별다른 중요한 작업이 없기 때문에 원한다면 테스트 코드를 작성하지 않아도 괜찮다.

1.3 등록 구현 - 컨트롤러와 프레젠테이션 계층

1.3.1 컨트롤러의 구현

(1) 컨트롤러 관련 고민들

- 공통 경로의 처리와 호출 방식
 - 외부나 다른 사람에게 메신저 등으로 보낼 수 있게 하려면 반드시 GET 방식으로 처리한다. 즉, '조회'가 가능하도록 만들어야 하는 모든 경우는 GET 방식으로 설계한다.
 - POST 방식은 현재 사용자가 스스로 작업하는 내용이 있는 경우에 사용한다. 즉 외부에 노출하는 것이 아니라 사용자 본인이 결정해서 어떤 작업이 진행되는 일은 POST 방식으로 처리한다.
 - 스토리보드에 맞는 URI 설계 예

방식	URI	설명
GET	/board/register	게시물의 등록 페이지를 보여준다.
POST	/board/register	게시물을 실제로 등록한다.
GET	/board/red?bno=xxx	특정 번호의 게시물을 조회한다.
GET	/board/mod?bno=xxx	게시물의 수정 화면으로 이동한다.
POST	/board/mod	게시물을 수정한다.
POST	/board/remove	게시물을 삭제한다.
GET	/board/list	게시물의 목록을 확인한다.

- 리다이렉트의 처리 방식
 - 등록 작업이 끝나고 리스트가 보여지기 전에 사용자에게 어떤 식으로 등록이 성공했는지를 알려줄 것인가?
 - 수정 작업이 끝나고 어떤 식으로 결과를 알려주고 리스트로 이동하는가?
 - 삭제 작업이 끝나면 어떻게 결과를 알려줘야 하는가?

(2) 컨트롤러의 선언

- 스프링 MVC를 이용하는 경우에는 컨트롤러 역시 '@Controller'라는 어노테이션을 추가하는 것만으로 설정이 완료된다.
- 등록 작업과 파라미터의 결정
 - 파라미터의 수집은 스프링 MVC에서 자동으로 이루어지므로, 파라미터의 수집이 필요하면 원하는 객체를 파라미터로 선언한다.
 - 특별한 경우가 아니라면 VO 클래스 혹은 DTO 클래스를 파라미터로 사용하는 것이 편리하다.
- 등록 작업의 메소드
 - 등록을 위한 입력 페이지를 보는 경우(GET 방식으로 처리)
 - 실제로 데이터를 처리하는 부분(POST 방식으로 처리)

[ex01/src/main/java/org/zerock/controller/BoardController.java]

```
01 package org.zerock.controller;
02
03 import javax.inject.Inject;
04
05 import org.slf4j.Logger;
06 import org.slf4j.LoggerFactory;
07 import org.springframework.stereotype.Controller;
08 import org.springframework.ui.Model;
09 import org.springframework.web.bind.annotation.ModelAttribute;
10 import org.springframework.web.bind.annotation.RequestMapping;
11 import org.springframework.web.bind.annotation.RequestMethod;
12 import org.springframework.web.bind.annotation.RequestParam;
13 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
14 import org.zerock.domain.BoardVO;
15 import org.zerock.domain.Criteria;
16 import org.zerock.domain.PagerMaker;
17 import org.zerock.service.BoardService;
18
19 @Controller
20 @RequestMapping("/board/*")
21 public class BoardController {
22
23     private static final Logger logger = LoggerFactory.getLogger(BoardController.class);
24
25     @Inject
26     private BoardService service;
27
28     @RequestMapping(value = "/register", method = RequestMethod.GET)
29     public void registerGET(BoardVO board, Model model) throws Exception {
30
31         logger.info("register get .....");
32     }
33
34     @RequestMapping(value = "/register", method = RequestMethod.POST)
35     public String registPOST(BoardVO board, RedirectAttributes rttr) throws Exception {
36
37         logger.info("regist post .....");
38         logger.info(board.toString());
39
40         service.regist(board);
41
42         rttr.addFlashAttribute("msg", "SUCCESS");
43         return "redirect:/board/listAll";
44     }
45
46     @RequestMapping(value = "/listAll", method = RequestMethod.GET)
47     public void listAll(Model model) throws Exception {
48
49         logger.info("show all list.....");
50         model.addAttribute("list", service.listAll());
51     }
52
53     @RequestMapping(value = "/read", method = RequestMethod.GET)
54     public void read(@RequestParam("bno") int bno, Model model) throws Exception {
55
56         model.addAttribute(service.read(bno));
57     }
58
59     @RequestMapping(value = "/remove", method = RequestMethod.POST)
60     public String remove(@RequestParam("bno") int bno, RedirectAttributes rttr) throws Exception {
61
62         service.remove(bno);
63     }
64 }
```



```

64     rttr.addFlashAttribute("msg", "SUCCESS");
65
66     return "redirect:/board/listAll";
67 }
68
69 @RequestMapping(value = "/modify", method = RequestMethod.GET)
70 public void modifyGET(int bno, Model model) throws Exception {
71
72     model.addAttribute(service.read(bno));
73 }
74
75 @RequestMapping(value = "/modify", method = RequestMethod.POST)
76 public String modifyPOST(BoardVO board, RedirectAttributes rttr) throws Exception {
77
78     logger.info("mod post.....");
79
80     service.modify(board);
81     rttr.addFlashAttribute("msg", "SUCCESS");
82
83     return "redirect:/board/listAll";
84 }
85
86 @RequestMapping(value = "/listCri", method = RequestMethod.GET)
87 public void listAll(Criteria cri, Model model) throws Exception {
88
89     logger.info("show list Page with Criteria.....");
90
91     model.addAttribute("list", service.listCriteria(cri));
92 }
93
94 @RequestMapping(value = "/listPage", method = RequestMethod.GET)
95 public void listPage(@ModelAttribute("cri") Criteria cri, Model model) throws Exception {
96
97     logger.info(cri.toString());
98
99     model.addAttribute("list", service.listCriteria(cri));
100     PageMaker pageMaker = new PageMaker();
101     pageMaker.setCri(cri);
102     // pageMaker.setTotalCount(131);
103
104     pageMaker.setTotalCount(service.listCountCriteria(cri));
105
106     model.addAttribute("pageMaker", pageMaker);
107 }
108
109 @RequestMapping(value = "/readPage", method = RequestMethod.GET)
110 public void read(@RequestParam("bno") int bno, @ModelAttribute("cri") Criteria cri, Model model)
111 throws Exception {
112
113     model.addAttribute(service.read(bno));
114 }
115
116 @RequestMapping(value = "/removePage", method = RequestMethod.POST)
117 public String remove(@RequestParam("bno") int bno, Criteria cri, RedirectAttributes rttr) throws
118 Exception {
119
120     service.remove(bno);
121
122     rttr.addAttribute("page", cri.getPage());
123     rttr.addAttribute("perPageNum", cri.getPerPageNum());
124     rttr.addFlashAttribute("msg", "SUCCESS");
125
126     return "redirect:/board/listPage";
127 }
128

```

```

129     @RequestMapping(value = "/modifyPage", method = RequestMethod.GET)
130     public void modifyPagingGET(@RequestParam("bno") int bno, @ModelAttribute("cri") Criteria cri, Model
131     model)
132         throws Exception {
133
134         model.addAttribute(service.read(bno));
135     }
136
137 }

```

1.3.2 컨트롤러의 동작 확인과 루트 경로 지정

- <http://localhost:8080/>

1.3.3 뷰(view)의 구현 - 등록

- <http://localhost:8080/board/register>

(1) 컨트롤러에서의 데이터 전달

- /board/register (GET) -> BoardController -> /WEB-INF/views/board/register.jsp -> BoardController -> /WEB-INF/views/board/listAll.jsp

(2) 결과 페이지의 문제점 - 새로 고침

- 이 문제를 막기 위해서 가장 간단하게 사용할 수 있는 방법은 바로 페이지를 다른 곳으로 이동하는 리다이렉트를 이용하는 방법이다.

(3) 리다이렉트(redirect)와 결과 데이터

- /board/register (POST) -> BoardController -> /board/listAll (GET) -> BoardController

(4) RedirectAttributes를 이용한 숨김 데이터의 전송

- Spring의 RedirectAttributes 객체는 리다이렉트 시점에 한 번만 사용되는 데이터를 전송할 수 있는 addFlashAttribute()라는 기능을 지원한다.

```

// org.zerock.controller.BoardController의 일부

...(생략)...
@RequestMapping(value = "/register", method = RequestMethod.POST)
public String registPOST(BoardVO board, RedirectAttributes rttr) throws Exception {

    logger.info("regist post .....");
    logger.info(board.toString());

    service.regist(board);
}

```

```
rttr.addFlashAttribute("msg", "SUCCESS");
return "redirect:/board/listAll";
}
...(생략)...
```

```
// /WEB-INF/views/board/listAll.jsp의 일부

...(생략)...
<script>
    var result = '${msg}';

    if (result == 'SUCCESS') {
        alert("처리가 완료되었습니다.");
    }
</script>
...(생략)...
```

1.4 전체 목록 구현

- <http://localhost:8080/board/listAll>

1.4.1 컨트롤러의 완성 및 JSP의 완성

- org.zerock.controller.BoardController.java의 일부
- /WEB-INF/views/board/listAll.jsp의 일부

1.4.2 목록에 추가로 구현해야 하는 사항들

- 페이지 처리
- 검색 기능

1.5 조회 구현

- 조회 기능의 구현은 다음과 같은 순서로 구현된다.
 - 조회 기능을 위한 BoardDAO의 처리
 - BoardService, BoardServiceImpl, BoardController의 처리
 - 조회 페이지의 작성
 - 수정, 삭제 링크 처리

1.5.1 BoardController의 기능 추가와 뷰 처리

- 파라미터는 '@RequestParam'을 통해 외부에서 전달될 bno 값을 전달받는다. @RequestParam 어노테이션은 Servlet에서 request.getParameter()의 효과와 유사하다. 다른 점은 문자열, 숫자, 날짜 등의 형 변환이 가능하다는 점이다.
- 조회된 결과 게시물을 JSP로 전달해야 하기 때문에 Model 객체를 사용한다.

- 스프링의 Model은 addAttribute() 작업을 할 때 아무런 이름 없이 데이터를 넣으면 자동으로 클래스의 이름을 소문자로 시작해서 사용하게 된다. 즉 들어오는 데이터는 BoardVO 클래스의 객체이므로, boardVO라는 이름으로 저장된다.

```
// org.zerock.controller.BoardController의 일부

...(생략)...
@RequestMapping(value = "/read", method = RequestMethod.GET)
public void read(@RequestParam("bno") int bno, Model model) throws Exception {

    model.addAttribute(service.read(bno));
    //model.addAttribute("boardVO", service.read(bno));
}
...(생략)...
```

(1) 조회용 페이지 작성

- JSP의 경우 결과 데이터의 변수 이름은 'boardVO' 이다.
- 조회 화면의 경우, 나중에 수정이나 삭제 작업에서 사용되기 때문에 반드시 원래 게시물 번호인 bno를 가지고 있어야만 한다. <input type='hidden'>을 이용해서 bno 값을 처리한다.

```
// /WEB-INF/views/board/read.jsp의 일부

...(생략)...
<form role="form" method="post">

    <input type='hidden' name='bno' value='${boardVO.bno}'>

</form>

<div class="box-body">
    <div class="form-group">
        <label for="exampleInputEmail1">Title</label> <input type="text"
            name='title' class="form-control" value='${boardVO.title}'
            readonly="readonly">
    </div>
    <div class="form-group">
        <label for="exampleInputPassword1">Content</label>
        <textarea class="form-control" name="content" rows="3"
            readonly="readonly">${boardVO.content}</textarea>
    </div>
    <div class="form-group">
        <label for="exampleInputEmail1">Writer</label> <input type="text"
            name="writer" class="form-control" value='${boardVO.writer}'
            readonly="readonly">
    </div>
</div>
<!-- /.box-body -->

<div class="box-footer">
    <button type="submit" class="btn btn-warning">Modify</button>
    <button type="submit" class="btn btn-danger">REMOVE</button>
    <button type="submit" class="btn btn-primary">LIST ALL</button>
</div>
...(생략)...
```

1.5.2 수정, 삭제로의 링크 처리

- 게시물 수정은 별도의 수정이 가능한 페이지로 이동한 후에 처리한다.
- 게시물 삭제는 조회 페이지에서 바로 삭제 처리를 할 수 있도록 한다.

```
// /WEB-INF/views/board/read.jsp의 일부

...(생략)...
<script>
    $(document).ready(function() {

        var formObj = $("form[role='form']");

        console.log(formObj);

        $(".btn-warning").on("click", function() {
            formObj.attr("action", "/board/modify");
            formObj.attr("method", "get");
            formObj.submit();
        });

        $(".btn-danger").on("click", function() {
            formObj.attr("action", "/board/remove");
            formObj.submit();
        });

        $(".btn-primary").on("click", function() {
            self.location = "/board/listAll";
        });

    });
</script>
...(생략)...
```

1.6 삭제/수정 처리

1.6.1 삭제 처리

- 삭제는 등록 기능과 유사한 부분이 많은데, 삭제 후 페이지의 이동을 제대로 처리하지 않으면 브라우저에서 '새로 고침'을 통해 계속해서 동일한 데이터가 재전송될 수 있는 문제가 있다.

```
// org.zerock.controller.BoardController의 일부

@RequestMapping(value = "/remove", method = RequestMethod.POST)
public String remove(@RequestParam("bno") int bno, RedirectAttributes rttr) throws Exception {

    service.remove(bno);

    rttr.addFlashAttribute("msg", "SUCCESS");

    return "redirect:/board/listAll";
}
```

1.6.2 수정 처리

- 경우에 따라서 다를 수 있지만, 이번 장에서는 수정할 수 있는 별도의 화면을 제공하는 형태로 제작한다.

```
// org.zerock.controller.BoardController의 일부

@RequestMapping(value = "/modify", method = RequestMethod.GET)
public void modifyGET(int bno, Model model) throws Exception {

    model.addAttribute(service.read(bno));
}

@RequestMapping(value = "/modify", method = RequestMethod.POST)
public String modifyPOST(BoardVO board, RedirectAttributes rttr) throws Exception {

    logger.info("mod post.....");

    service.modify(board);
    rttr.addFlashAttribute("msg", "SUCCESS");

    return "redirect:/board/listAll";
}
```

```
// /WEB-INF/views/board/modify.jsp의 일부

<form role="form" method="post">

    <div class="box-body">

        <div class="form-group">
            <label for="exampleInputEmail1">BNO</label> <input type="text"
                name='bno' class="form-control" value="${boardVO.bno}"
                readonly="readonly">
        </div>

        <div class="form-group">
            <label for="exampleInputEmail1">Title</label> <input type="text"
                name='title' class="form-control" value="${boardVO.title}">
        </div>
        <div class="form-group">
            <label for="exampleInputPassword1">Content</label>
            <textarea class="form-control" name="content"
                rows="3">${boardVO.content}</textarea>
        </div>
        <div class="form-group">
            <label for="exampleInputEmail1">Writer</label> <input type="text"
                name="writer" class="form-control"
                value="${boardVO.writer}">
        </div>
    </div>
    <!-- /.box-body -->
</form>

<div class="box-footer">
    <button type="submit" class="btn btn-primary">SAVE</button>
    <button type="submit" class="btn btn-warning">CANCEL</button>
</div>
```

1.7 예외 처리

1.7.1 예외 처리에 대한 팁

- Spring MVC를 사용할 때 Controller쪽에서 Exception을 처리하기 위해서는 다음과 같은 방식들을 이용한다.
 - @ExceptionHandler 어노테이션을 이용한 처리
 - @ResponseStatus를 이용한 Http 상태 코드 처리
 - @ControllerAdvice를 이용한 처리: 호출되는 메소드에서 발생한 Exception을 모두 처리하는 역할을 한다.
 - 클래스에 @ControllerAdvice라는 어노테이션 처리
 - 각 메소드에 @ExceptionHandler를 이용해서 적절한 타입의 Exception을 처리

```
// org.zerock.controller.CommonExceptionAdvice의 일부
package org.zerock.controller;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.servlet.ModelAndView;

@ControllerAdvice
public class CommonExceptionAdvice {

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

    //ExceptionHandler(Exception.class)
    public String common(Exception e) {

        logger.info(e.toString());

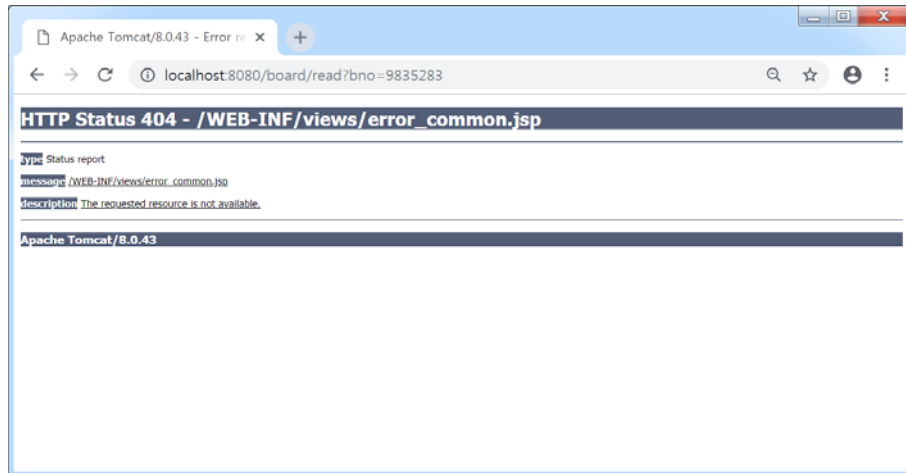
        return "error_common";
    }

    @ExceptionHandler(Exception.class)
    private ModelAndView errorModelAndView(Exception ex) {

        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("/error_common");
        modelAndView.addObject("exception", ex);

        return modelAndView;
    }
}
```

- 예를 들어 위의 코드를 작성한 후에 BoardController에 잘못된 파라미터를 전달해 보면 어떤 식으로 동작하는지 잘 알 수 있다.



1.7.2 Exception을 화면으로 전달하기

- error_common.jsp에서는 BoardController의 실행 중에 발생하는 Exception을 상세하게 볼 수 있게 다음과 같이 작성한다.

```
// /WEB-INF/views/error_common.jsp의 일부

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

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>

    <h4>${exception.getMessage()} </h4>

    <ul>
        <c:forEach items="${exception.getStackTrace()}" var="stack">
            <li>${stack.toString()} </li>
        </c:forEach>
    </ul>

</body>
</html>
```

1.8 페이징 처리-영속 계층, 비즈니스 계층

1.8.1 페이징 처리 방식

- <a> 태그의 href 속성을 이용해서 직접 이동할 URI를 지정하는 방식
- <form> 태그를 이용해서 링크를 클릭하면 여러 정보를 전달하는 방식

(1) 페이징 처리의 원칙

- 페이지는 다른 사람에게 URL로 전달하는 경우가 많기 때문에, 반드시 GET 방식으로만 처리한다.
- 페이지 처리가 되면 조회 화면에서 반드시 '목록 가기'가 필요하다.
- 페이지 처리에는 반드시 필요한 페이지 번호만을 제공한다.

1.8.2 페이징 처리 개발에 필요한 지식

- 페이징 처리를 위한 SQL
- 데이터 개수 파악을 위한 SQL
- 자바스크립트 혹은 <a> 태그를 통한 이벤트 처리

(1) MySQL의 limit를 이용한 페이지 출력 SQL

- SQL을 처리할 때 가장 중요한 점은 빠른 데이터의 출력을 위해서 가능하면 결과의 양을 적게 유지하는 것이 좋다. 즉 페이지당 10개의 데이터를 출력한다면, 특정 페이지 번호에 맞는 데이터를 가져올 때 필요한 10개의 데이터만 가져와야만 한다.

```
select * from tbl_board where bno > 0 order by bno desc limit 0, 10;
```

[꿀팁] 오라클의 ROWNUM를 이용한 페이지 출력 SQL

```
-- 급여(SAL)를 많이 받는 6~10째 사원을 출력하기 위해서는 인라인 뷰안에 또 다른 인라인 뷰를 사용해야 한다.  
-- 또한 ROWNUM 칼럼에 별칭을 부여해야 검색이 가능하다.  
SELECT ROWNUM, RNUM, ENAME, SAL FROM  
(SELECT ROWNUM RNUM, ENAME, SAL FROM  
(SELECT * FROM EMP ORDER BY SAL DESC))  
WHERE RNUM BETWEEN 6 AND 10;
```

(2) 충분한 양의 데이터 넣기

- 아래 SQL 문을 여러 번 실행하면 현재 데이터의 배수만큼 데이터가 들어가므로 100만건의 데이터 역시 쉽게 만들어 낼 수 있다.

```
insert into tbl_board (title, content, writer)  
(select title, content, writer from tbl_board);  
  
select count(*) from tbl_board;
```

1.8.3 MyBatis의 BoardDAO 처리

(1) BoardDAO, XML Mapper, BoardDAOImpl의 처리

```
// org.zerock.persistence.BoardDAO.java

package org.zerock.persistence;

import java.util.List;

import org.zerock.domain.BoardVO;
import org.zerock.domain.Criteria;
import org.zerock.domain.SearchCriteria;

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;

    public List<BoardVO> listPage(int page) throws Exception;

    public List<BoardVO> listCriteria(Criteria cri) throws Exception;

    public int countPaging(Criteria cri) throws Exception;

    //use for dynamic sql

    public List<BoardVO> listSearch(SearchCriteria cri) throws Exception;

    public int listSearchCount(SearchCriteria cri) throws Exception;

}

```

```
// org.zerock.persistence.BoardDAOImpl.java의 일부

@Override
public List<BoardVO> listPage(int page) throws Exception {

    if (page <= 0) {
        page = 1;
    }

    page = (page - 1) * 10;

    return session.selectList(namespace + ".listPage", page);
}

```

```
// /resources/mappers/boardMapper.xml의 일부

<select id="listPage" resultType="BoardVO">
    <![CDATA[
    select
        bno, title, content, writer, regdate, viewcnt
    from
        tbl_board
    where bno > 0
    ]]>

```

```

    order by bno desc, regdate desc
    limit #{page}, 10
  ]]>
</select>

```

(2) 페이징 처리의 SQL 테스트

```

// test 폴더 내의 org.zerock.test.BoardDAOTest.java의 일부

<select id="listPage" resultType="BoardVO">
  <![CDATA[
    select
      bno, title, content, writer, regdate, viewcnt
    from
      tbl_board
    where bno > 0
    order by bno desc, regdate desc
    limit #{page}, 10
  ]]>
</select>

```

1.8.4 DAO 처리를 도와줄 Criteria 클래스 만들기

```

[org.zerock.domain.Criteria.java]

01  package org.zerock.domain;
02
03  public class Criteria {
04
05      private int page;
06      private int perPageNum;
07
08      public Criteria() {
09          this.page = 1;
10          this.perPageNum = 10;
11      }
12
13      public void setPage(int page) {
14
15          if (page <= 0) {
16              this.page = 1;
17              return;
18          }
19
20          this.page = page;
21      }
22
23      public void setPerPageNum(int perPageNum) {
24
25          if (perPageNum <= 0 || perPageNum > 100) {
26              this.perPageNum = 10;
27              return;
28          }
29
30          this.perPageNum = perPageNum;
31      }
32
33      public int getPage() {

```

```

34     return page;
35 }
36
37 // method for MyBatis SQL Mapper -
38 public int getPageStart() {
39
40     return (this.page - 1) * perPageNum;
41 }
42
43 // method for MyBatis SQL Mapper
44 public int getPerPageNum() {
45
46     return this.perPageNum;
47 }
48
49 @Override
50 public String toString() {
51     return "Criteria [page=" + page + ", "
52         + "perPageNum=" + perPageNum + "]\n";
53 }
54 }

```

1.8.5 BoardService 수정하기

```

// org.zerock.service.BoardService.java

package org.zerock.service;

import java.util.List;

import org.zerock.domain.BoardVO;
import org.zerock.domain.Criteria;
import org.zerock.domain.SearchCriteria;

public interface BoardService {

    public void regist(BoardVO board) throws Exception;

    public BoardVO read(Integer bno) throws Exception;

    public void modify(BoardVO board) throws Exception;

    public void remove(Integer bno) throws Exception;

    public List<BoardVO> listAll() throws Exception;

    public List<BoardVO> listCriteria(Criteria cri) throws Exception;

    public int listCountCriteria(Criteria cri) throws Exception;

    public List<BoardVO> listSearchCriteria(SearchCriteria cri)
        throws Exception;

    public int listSearchCount(SearchCriteria cri) throws Exception;

}

```

```

// org.zerock.service.BoardServiceImpl.java의 일부

```

```

@Override
public List<BoardVO> listCriteria(Criteria cri) throws Exception {

    return dao.listCriteria(cri);
}

```

1.9 페이징 처리-컨트롤러와 프레젠테이션 계층

- 스프링 MVC의 컨트롤러는 특정 URL에 해당하는 메소드를 실행할 때, 파라미터의 타입을 보고 해당 객체를 자동으로 생성하고 수집하기 때문에 이전에 만든 Criteria라는 클래스를 그대로 사용한다.

[ex01/src/main/java/org/zerock/controller/BoardController.java]

```

01 package org.zerock.controller;
02
03 import javax.inject.Inject;
04
05 import org.slf4j.Logger;
06 import org.slf4j.LoggerFactory;
07 import org.springframework.stereotype.Controller;
08 import org.springframework.ui.Model;
09 import org.springframework.web.bind.annotation.ModelAttribute;
10 import org.springframework.web.bind.annotation.RequestMapping;
11 import org.springframework.web.bind.annotation.RequestMethod;
12 import org.springframework.web.bind.annotation.RequestParam;
13 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
14 import org.zerock.domain.BoardVO;
15 import org.zerock.domain.Criteria;
16 import org.zerock.domain.PagerMaker;
17 import org.zerock.service.BoardService;
18
19 @Controller
20 @RequestMapping("/board/*")
21 public class BoardController {
22
23     private static final Logger logger = LoggerFactory.getLogger(BoardController.class);
24
25     @Inject
26     private BoardService service;
27
28     @RequestMapping(value = "/register", method = RequestMethod.GET)
29     public void registerGET(BoardVO board, Model model) throws Exception {
30
31         logger.info("register get .....");
32     }
33
34     @RequestMapping(value = "/register", method = RequestMethod.POST)
35     public String registPOST(BoardVO board, RedirectAttributes rttr) throws Exception {
36
37         logger.info("regist post .....");
38         logger.info(board.toString());
39
40         service.regist(board);
41
42         rttr.addFlashAttribute("msg", "SUCCESS");
43         return "redirect:/board/listAll";
44     }
45
46     @RequestMapping(value = "/listAll", method = RequestMethod.GET)
47     public void listAll(Model model) throws Exception {

```

```

48
49     logger.info("show all list.....");
50     model.addAttribute("list", service.listAll());
51 }
52
53 @RequestMapping(value = "/read", method = RequestMethod.GET)
54 public void read(@RequestParam("bno") int bno, Model model) throws Exception {
55
56     model.addAttribute(service.read(bno));
57 }
58
59 @RequestMapping(value = "/remove", method = RequestMethod.POST)
60 public String remove(@RequestParam("bno") int bno, RedirectAttributes rttr) throws Exception {
61
62     service.remove(bno);
63
64     rttr.addFlashAttribute("msg", "SUCCESS");
65
66     return "redirect:/board/listAll";
67 }
68
69 @RequestMapping(value = "/modify", method = RequestMethod.GET)
70 public void modifyGET(int bno, Model model) throws Exception {
71
72     model.addAttribute(service.read(bno));
73 }
74
75 @RequestMapping(value = "/modify", method = RequestMethod.POST)
76 public String modifyPOST(BoardVO board, RedirectAttributes rttr) throws Exception {
77
78     logger.info("mod post.....");
79
80     service.modify(board);
81     rttr.addFlashAttribute("msg", "SUCCESS");
82
83     return "redirect:/board/listAll";
84 }
85
86 @RequestMapping(value = "/listCri", method = RequestMethod.GET)
87 public void listAll(Criteria cri, Model model) throws Exception {
88
89     logger.info("show list Page with Criteria.....");
90
91     model.addAttribute("list", service.listCriteria(cri));
92 }
93
94 @RequestMapping(value = "/listPage", method = RequestMethod.GET)
95 public void listPage(@ModelAttribute("cri") Criteria cri, Model model) throws Exception {
96
97     logger.info(cri.toString());
98
99     model.addAttribute("list", service.listCriteria(cri));
100     PageMaker pageMaker = new PageMaker();
101     pageMaker.setCri(cri);
102     // pageMaker.setTotalCount(131);
103
104     pageMaker.setTotalCount(service.listCountCriteria(cri));
105
106     model.addAttribute("pageMaker", pageMaker);
107 }
108
109 @RequestMapping(value = "/readPage", method = RequestMethod.GET)
110 public void read(@RequestParam("bno") int bno, @ModelAttribute("cri") Criteria cri, Model model)
111 throws Exception {
112

```

```

113     model.addAttribute(service.read(bno));
114 }
115
116 @RequestMapping(value = "/removePage", method = RequestMethod.POST)
117 public String remove(@RequestParam("bno") int bno, Criteria cri, RedirectAttributes rttr) throws
118 Exception {
119
120     service.remove(bno);
121
122     rttr.addAttribute("page", cri.getPage());
123     rttr.addAttribute("perPageNum", cri.getPerPageNum());
124     rttr.addFlashAttribute("msg", "SUCCESS");
125
126     return "redirect:/board/listPage";
127 }
128
129 @RequestMapping(value = "/modifyPage", method = RequestMethod.GET)
130 public void modifyPagingGET(@RequestParam("bno") int bno, @ModelAttribute("cri") Criteria cri, Model
131 model)
132     throws Exception {
133
134     model.addAttribute(service.read(bno));
135 }
136
137 }

```

1.9.1 1차 화면 테스트

- 이전에 만들어진 listAll.jsp 페이지를 복사해서 listCri.jsp 페이지로 제작한다.
- 실행: <http://localhost:8080/board/listCri?page=3&perPageNum=5>

1.9.2 화면 하단의 페이징 처리

- 화면에서 페이징 처리한 결과를 보여주기 위해서는 반드시 몇 개의 데이터가 필요하다.
 - 시작 페이지 번호(startPage)
 - 끝 페이지 번호(endPage)
 - 전체 데이터의 개수(totalCount)
 - 이전 페이지 링크(prev)
 - 이후 페이지 링크(next)

(1) endPage 구하기

- endPage는 현재의 페이지 번호를 기준으로 계산한다.
- $endPage = (int) (Math.ceil(cri.getPage() / (double)displayPageNum) * displayPageNum);$
- cri.getPage()는 현재 페이지의 번호를 의미하며 아래와 계산할 수 있다.
 - 현재 페이지가 3일 때, $Math.ceil(3/10)*10=10$
 - 현재 페이지가 1일 때, $Math.ceil(1/10)*10=10$
 - 현재 페이지가 20일 때, $Math.ceil(20/10)*10=20$
 - 현재 페이지가 21일 때, $Math.ceil(21/10)*10=30$

(2) startPage 구하기

- endPage가 구해졌다면 startPage는 단순한 빼기의 문제이다.
- $startPage = (endPage - displayPageNum) + 1;$

(3) totalCount와 endPage의 재계산

- totalCount: 전체 데이터의 개수
- perPageNum: 한 번에 보여지는 데이터의 수
- 예를 들어 100개의 데이터(totalCount)를 10개씩 보여준다면(perPageNum) endPage는 10이 되어야 하지만, 20개씩 보여주는 경우에는 endPage는 5가 되어야 한다.

```
int tempEndPage = (int)(Math.ceil(totalCount / (double)cri.getPerPageNum()));
if(endPage > tempEndPage) {
    endPage = tempEndPage;
}
```

(4) prev와 next의 계산

- pre의 경우는 startPage가 1이 아닌지를 검사하는 것으로 충분하다.
- next의 경우는 뒤에 더 데이터가 남아 있는지에 대한 처리하므로 $endPage * perPageNum$ 이 totalCount보다 작은지를 확인해 줘야 한다.

```
pre = startPage==1 ? false : true;
next = endPage * cri.getPerPageNum() >= totalCount ? false : true;
```

1.9.3 페이징 처리용 클래스 설계하기

- 페이지의 계산을 매번 JSP 등에서 처리할 수도 있겠지만, 좀 더 편하게 사용하기 위해서는 별도의 클래스를 설계해서 처리하는 것이 좋다.
- 해당 클래스를 통해 아래와 같은 필요한 데이터를 구한다.
 - page: 현재 조회하는 페이지의 번호
 - perPageNum: 한 페이지당 출력하는 데이터의 개수
 - totalCount: SQL의 결과로 나온 데이터의 전체 개수
 - startPage
 - endPage
 - prev
 - next

[org.zerock.domain.PageMaker.java]

```
01 package org.zerock.domain;
02
03 import org.springframework.web.util.UriComponents;
04 import org.springframework.web.util.UriComponentsBuilder;
05
06 public class PageMaker {
07
08     private int totalCount;
09     private int startPage;
```



```

10     private int endPage;
11     private boolean prev;
12     private boolean next;
13
14     private int displayPageNum = 10;
15
16     private Criteria cri;
17
18     public void setCri(Criteria cri) {
19         this.cri = cri;
20     }
21
22     public void setTotalCount(int totalCount) {
23         this.totalCount = totalCount;
24
25         calcData();
26     }
27
28     private void calcData() {
29
30         endPage = (int) (Math.ceil(cri.getPage() / (double) displayPageNum) *
31 displayPageNum);
32
33         startPage = (endPage - displayPageNum) + 1;
34
35         int tempEndPage = (int) (Math.ceil(totalCount / (double) cri.getPerPageNum()));
36
37         if (endPage > tempEndPage) {
38             endPage = tempEndPage;
39         }
40
41         prev = startPage == 1 ? false : true;
42
43         next = endPage * cri.getPerPageNum() >= totalCount ? false : true;
44
45     }
46
47     public int getTotalCount() {
48         return totalCount;
49     }
50
51     public int getStartPage() {
52         return startPage;
53     }
54
55     public int getEndPage() {
56         return endPage;
57     }
58
59     public boolean isPrev() {
60         return prev;
61     }
62
63     public boolean isNext() {
64         return next;
65     }
66
67     public int getDisplayPageNum() {
68         return displayPageNum;
69     }
70
71     public Criteria getCri() {
72         return cri;
73     }
74

```

```

75         public String makeQuery(int page) {
76
77             UriComponents uriComponents = UriComponentsBuilder.newInstance().queryParam("page",
78 page)
79                                     .queryParam("perPageNum", cri.getPerPageNum()).build();
80
81             return uriComponents.toUriString();
82         }
83
84         public String makeSearch(int page) {
85
86             UriComponents uriComponents = UriComponentsBuilder.newInstance().queryParam("page",
87 page)
88                                     .queryParam("perPageNum", cri.getPerPageNum())
89                                     .queryParam("searchType", ((SearchCriteria)
90 cri).getSearchType())
91                                     .queryParam("keyword", ((SearchCriteria)
92 cri).getKeyword()).build();
93
94             return uriComponents.toUriString();
95         }
96     }

```

1.9.4 BoardController와 뷰 처리

```

// org.zerock.controller.BoardController의 일부

@RequestMapping(value = "/listPage", method = RequestMethod.GET)
public void listPage(@ModelAttribute("cri") Criteria cri, Model model) throws Exception {

    logger.info(cri.toString());

    model.addAttribute("list", service.listCriteria(cri));
    PageMaker pageMaker = new PageMaker();
    pageMaker.setCri(cri);
    // pageMaker.setTotalCount(131);

    pageMaker.setTotalCount(service.listCountCriteria(cri));

    model.addAttribute("pageMaker", pageMaker);
}

```

(1) listPage.jsp의 처리

- 페이징 처리에 필요한 모든 데이터가 담긴 PageMaker는 최종적으로 listPage.jsp를 이용해서 보여지게 한다.

```

// /WEB-INF/views/board/listPage.jsp의 일부

<div class="box-footer">
    <div class="text-center">
        <ul class="pagination">

            <c:if test="${pageMaker.prev}">
                <li><a href="${pageMaker.startPage - 1}">&laquo;</a></li>
            </c:if>

            <c:forEach begin="${pageMaker.startPage }"

```

```

end="${pageMaker.endPage }" var="idx">
<li
    <c:out value="${pageMaker.cri.page == idx?'class
=active':''}"/>
    <a href="${idx}">${idx}</a>
</li>
</c:forEach>
<c:if test="${pageMaker.next && pageMaker.endPage > 0}">
    <li><a href="${pageMaker.endPage +1}">&raquo;</a></li>
</c:if>
</ul>
</div>
</div>

```

- 실행: <http://localhost:8080/board/listPage?page=6&perPageNum=10>

1.9.5 페이징을 위한 SQL 문의 처리

- 실제 게시물의 숫자를 이용해서 페이지를 처리할 수 있게 SQL을 작성한다.

(1) BoardDAO의 수정

- 우선 BoardDAO에서는 totalCount를 반환할 수 있게 처리하기 위해서 countPaging()을 추가한다.

```

// org.zerock.persistence.BoardDAO의 일부

public int countPaging(Criteria cri) throws Exception;

```

(2) XML Mapper의 수정

- BoardDAO에서 사용하게 될 SQL 문을 boardMapper.xml에 추가한다.

```

// /resources/mappers/boardMapper.xml의 일부

<select id="countPaging" resultType="int">
    <![CDATA[
        select
            count(bno)
        from
            tbl_board
        where
            bno > 0
    ]]>
</select>

```

(3) BoardDAOImpl의 수정

- MyBatis를 이용해서 처리하는 BoardDAOImpl은 다음과 같이 수정한다.

```
// org.zerock.persistence.BoardDAOImpl의 일부

@Override
public int countPaging(Criteria cri) throws Exception {

    return session.selectOne(namespace + ".countPaging", cri);
}
```

(4) BoardService/BoardServiceImpl의 수정

- 비즈니스 계층 역시 아래와 같이 Criteria를 이용해서 전체 게시물의 숫자를 계산하도록 변경한다.

```
// org.zerock.service.BoardService.java의 일부

public int listCountCriteria(Criteria cri) throws Exception;
```

```
// org.zerock.service.BoardServiceImpl.java의 일부

@Override
public List<BoardVO> listCriteria(Criteria cri) throws Exception {

    return dao.listCriteria(cri);
}
```

(5) BoardController의 수정

- BoardController의 listPage()에서 제대로 된 totalCount가 나올 수 있도록 수정한다.

```
// org.zerock.controller.BoardController의 일부

@RequestMapping(value = "/listPage", method = RequestMethod.GET)
public void listPage(@ModelAttribute("cri") Criteria cri, Model model) throws Exception {

    logger.info(cri.toString());

    model.addAttribute("list", service.listCriteria(cri));
    PageMaker pageMaker = new PageMaker();
    pageMaker.setCri(cri);
    // pageMaker.setTotalCount(131);

    pageMaker.setTotalCount(service.listCountCriteria(cri));

    model.addAttribute("pageMaker", pageMaker);
}
```

- 실행: <http://localhost:8080/board/listPage?page=22>

1.10 검색 처리와 동적 SQL

1.10.1 검색에 필요한 데이터와 SearchCriteria

- 목록 페이지를 만들려면 적어도 다음과 같은 정보가 필요하다.
 - 현재 페이지의 번호(page)
 - 페이지당 보여지는 데이터의 수(perPageNum)
 - 검색의 종류(searchType)
 - 검색의 키워드(keyword)

[org.zerock.domain.SearchCriteria.java]

```
01 package org.zerock.domain;
02
03 public class SearchCriteria extends Criteria{
04
05     private String searchType;
06     private String keyword;
07
08     public String getSearchType() {
09         return searchType;
10     }
11     public void setSearchType(String searchType) {
12         this.searchType = searchType;
13     }
14     public String getKeyword() {
15         return keyword;
16     }
17     public void setKeyword(String keyword) {
18         this.keyword = keyword;
19     }
20     @Override
21     public String toString() {
22         return super.toString() + " SearchCriteria "
23             + "[searchType=" + searchType + ", keyword="
24             + keyword + "]\n";
25     }
26 }
```

1.10.2 SearchBoardController의 작성

- 기존 소스와 구별이 가능하도록 새로운 컨트롤러를 작성해서 처리하도록 한다.

[org.zerock.controller.SearchBoardController.java]

```
01 package org.zerock.controller;
02
03 import javax.inject.Inject;
04
05 import org.slf4j.Logger;
06 import org.slf4j.LoggerFactory;
07 import org.springframework.stereotype.Controller;
08 import org.springframework.ui.Model;
09 import org.springframework.web.bind.annotation.ModelAttribute;
10 import org.springframework.web.bind.annotation.RequestMapping;
11 import org.springframework.web.bind.annotation.RequestMethod;
12 import org.springframework.web.bind.annotation.RequestParam;
13 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
14 import org.zerock.domain.BoardVO;
15 import org.zerock.domain.PagerMaker;
16 import org.zerock.domain.SearchCriteria;
```

```

17 import org.zerock.service.BoardService;
18
19 @Controller
20 @RequestMapping("/sboard/*")
21 public class SearchBoardController {
22
23     private static final Logger logger = LoggerFactory.getLogger(SearchBoardController.class);
24
25     @Inject
26     private BoardService service;
27
28     @RequestMapping(value = "/list", method = RequestMethod.GET)
29     public void listPage(@ModelAttribute("cri") SearchCriteria cri, Model model) throws Exception {
30
31         logger.info(cri.toString());
32
33         // model.addAttribute("list", service.listCriteria(cri));
34         model.addAttribute("list", service.listSearchCriteria(cri));
35
36         PageMaker pageMaker = new PageMaker();
37         pageMaker.setCri(cri);
38
39         // pageMaker.setTotalCount(service.listCountCriteria(cri));
40         pageMaker.setTotalCount(service.listSearchCount(cri));
41
42         model.addAttribute("pageMaker", pageMaker);
43     }
44
45     @RequestMapping(value = "/readPage", method = RequestMethod.GET)
46     public void read(@RequestParam("bno") int bno, @ModelAttribute("cri") SearchCriteria cri, Model model)
47         throws Exception {
48
49         model.addAttribute(service.read(bno));
50     }
51
52     @RequestMapping(value = "/removePage", method = RequestMethod.POST)
53     public String remove(@RequestParam("bno") int bno, SearchCriteria cri, RedirectAttributes rttr) throws
54     Exception {
55
56         service.remove(bno);
57
58         rttr.addAttribute("page", cri.getPage());
59         rttr.addAttribute("perPageNum", cri.getPerPageNum());
60         rttr.addAttribute("searchType", cri.getSearchType());
61         rttr.addAttribute("keyword", cri.getKeyword());
62
63         rttr.addFlashAttribute("msg", "SUCCESS");
64
65         return "redirect:/sboard/list";
66     }
67
68     @RequestMapping(value = "/modifyPage", method = RequestMethod.GET)
69     public void modifyPagingGET(int bno, @ModelAttribute("cri") SearchCriteria cri, Model model) throws
70     Exception {
71
72         model.addAttribute(service.read(bno));
73     }
74
75     @RequestMapping(value = "/modifyPage", method = RequestMethod.POST)
76     public String modifyPagingPOST(BoardVO board, SearchCriteria cri, RedirectAttributes rttr) throws
77     Exception {
78
79         logger.info(cri.toString());
80         service.modify(board);
81

```

```

82     rttr.addAttribute("page", cri.getPage());
83     rttr.addAttribute("perPageNum", cri.getPerPageNum());
84     rttr.addAttribute("searchType", cri.getSearchType());
85     rttr.addAttribute("keyword", cri.getKeyword());
86
87     rttr.addFlashAttribute("msg", "SUCCESS");
88
89     logger.info(rttr.toString());
90
91     return "redirect:/sboard/list";
92 }
93
94 @RequestMapping(value = "/register", method = RequestMethod.GET)
95 public void registGET() throws Exception {
96
97     logger.info("regist get .....");
98 }
99
100 @RequestMapping(value = "/register", method = RequestMethod.POST)
101 public String registPOST(BoardVO board, RedirectAttributes rttr) throws Exception {
102
103     logger.info("regist post .....");
104     logger.info(board.toString());
105
106     service.regist(board);
107
108     rttr.addFlashAttribute("msg", "SUCCESS");
109
110     return "redirect:/sboard/list";
111 }
112
113 }

```

1.10.3 검색에 필요한 JSP 수정

- SearchBoardController의 list()는 SearchCriteria를 @ModelAttribute로 사용하여 list.jsp에 자동으로 전달한다.
- 실행: <http://localhost:8080/sboard/list>

```

// /WEB-INF/views/sboard/list.jsp의 일부

<div class='box-body'>

    <select name="searchType">
        <option value="n"
            <c:out value="${cri.searchType == null?'selected':''}"/>
            ---</option>
        <option value="t"
            <c:out value="${cri.searchType eq 't'? 'selected':''}"/>
            Title</option>
        <option value="c"
            <c:out value="${cri.searchType eq 'c'? 'selected':''}"/>
            Content</option>
        <option value="w"
            <c:out value="${cri.searchType eq 'w'? 'selected':''}"/>
            Writer</option>
        <option value="tc"
            <c:out value="${cri.searchType eq 'tc'? 'selected':''}"/>
            Title OR Content</option>
        <option value="cw"

```

```

        <c:out value="${cri.searchType eq 'cw'? 'selected':''}"/>>
        Content OR Writer</option>
        <option value="tcw"
        <c:out value="${cri.searchType eq 'tcw'? 'selected':''}"/>>
        Title OR Content OR Writer</option>
    </select> <input type="text" name="keyword" id="keywordInput"
        value='${cri.keyword }'>
    <button id='searchBtn'>Search</button>
    <button id='newBtn'>New Board</button>

</div>

```

(1) searchType과 keyword 링크 처리

- PageMaker에서 적당한 URI에 사용할 문자열(query string)을 생성해 준다.

```

// org.zerock.domain.PageMaker.java의 일부

public String makeSearch(int page){

    UriComponents uriComponents =
        UriComponentsBuilder.newInstance()
            .queryParams("page", page)
            .queryParams("perPageNum", cri.getPerPageNum())
            .queryParams("searchType", ((SearchCriteria)cri).getSearchType())
            .queryParams("keyword", ((SearchCriteria)cri).getKeyword())
            .build();

    return uriComponents.toUriString();
}

```

- list.jsp 페이지 내에서는 페이징 처리 부분을 아래와 같이 만들어진 문자열이 적용될 수 있도록 작성한다.

```

// /WEB-INF/views/sboard/list.jsp의 일부

<ul class="pagination">

    <c:if test="${pageMaker.prev}">
        <li><a
            href="list${pageMaker.makeSearch(pageMaker.startPage -
1) }">&laquo;</a></li>
    </c:if>

    <c:forEach begin="${pageMaker.startPage }" end="${pageMaker.endPage }"
        var="idx">
        <li <c:out value="${pageMaker.cri.page == idx?'class =active':''}"/>>
            <a href="list${pageMaker.makeSearch(idx)}">${idx}</a>
        </li>
    </c:forEach>

    <c:if test="${pageMaker.next && pageMaker.endPage > 0}">
        <li><a href="list${pageMaker.makeSearch(pageMaker.endPage
+1) }">&raquo;</a></li>
    </c:if>

</ul>

```

- <http://localhost:8080/sboard/list?page=1&perPageNum=10&searchType=t&keyword=test>

(2) 검색 버튼의 동작 처리

- 검색 버튼의 클릭에는 다음과 같은 의미가 있다.
 - 사용자는 새로운 검색을 시도한다. 기존의 페이징이나 검색이 아닌 새로운 검색을 시도하는 경우에만 검색 버튼을 사용한다.
 - 검색 버튼의 클릭은 1페이지의 데이터를 의미한다.

```
// /WEB-INF/views/sboard/list.jsp의 일부

<script>
    $(document).ready(
        function() {
            $('#searchBtn').on(
                "click",
                function(event) {
                    self.location = "list"
                    +
                    '${pageMaker.makeQuery(1)}'
                    + "&searchType="
                    + "${select
                    option:selected").val()
                    + "&keyword=" +
                    $('#keywordInput').val();

                });

            $('#newBtn').on("click", function(evt) {
                self.location = "register";
            });
        });
</script>
```

1.10.4 MyBatis 동적 SQL

- 화면에서의 검색 조건에 따라 검색해야 하는 SQL 문이 달라지기 때문에 이를 처리하기 위해서는 MyBatis의 동적 SQL 기능을 활용한다.

(1) BoardDAO의 수정

```
// org.zerock.persistence.BoardDAO.java의 일부

public List<BoardVO> listSearch(SearchCriteria cri)throws Exception;

public int listSearchCount(SearchCriteria cri)throws Exception;
```

(2) XML Mapper 수정

```
// resources/mappers/boardMapper.xml의 일부

<select id="listSearch" resultType="BoardV0">
<![CDATA[
    select *
    from tbl_board
    where bno > 0
]]>

    <include refid="search"></include>

<![CDATA[
    order by bno desc
    limit #{pageStart}, #{perPageNum}
]]>
</select>

<select id="listSearchCount" resultType="int">
<![CDATA[
    select count(bno)
    from tbl_board
    where bno > 0
]]>
    <include refid="search"></include>

</select>
```

(3) BoardDAOImpl의 수정

```
// org.zerock.persistence.BoardDAOImpl.java의 일부

@Override
public List<BoardV0> listSearch(SearchCriteria cri) throws Exception {

    return session.selectList(namespace + ".listSearch", cri);
}

@Override
public int listSearchCount(SearchCriteria cri) throws Exception {

    return session.selectOne(namespace + ".listSearchCount", cri);
}
```

(4) BoardDAO의 테스트

```
// test파일의 org.zerock.test.BoardDAOTest.java의 일부

@Test
public void testDynamic1() throws Exception {

    SearchCriteria cri = new SearchCriteria();
    cri.setPage(1);
    cri.setKeyword("글");
    cri.setSearchType("t");
```

```

        logger.info("=====");

        List<BoardVO> list = dao.listSearch(cri);

        for (BoardVO boardVO : list) {
            logger.info(boardVO.getBno() + ": " + boardVO.getTitle());
        }

        logger.info("=====");

        logger.info("COUNT: " + dao.listSearchCount(cri));
    }

```

(5) 동적 SQL 문의 추가

- 재사용을 위해서 분리된 <sql>은 필요한 경우에 <include>를 이용해서 사용할 수 있다.

```

// resources/mappers/boardMapper.xml의 일부

<sql id="search">
    <if test="searchType != null" >
        <if test="searchType = 't'.toString()">
            and title like CONCAT('%', #{keyword}, '%')
        </if>
        <if test="searchType = 'c'.toString()">
            and content like CONCAT('%', #{keyword}, '%')
        </if>
        <if test="searchType = 'w'.toString()">
            and writer like CONCAT('%', #{keyword}, '%')
        </if>
        <if test="searchType = 'tc'.toString()">
            and ( title like CONCAT('%', #{keyword}, '%') OR content like CONCAT('%', #{keyword}, '%'))
        </if>
        <if test="searchType = 'cw'.toString()">
            and ( content like CONCAT('%', #{keyword}, '%') OR writer like CONCAT('%', #{keyword}, '%'))
        </if>
        <if test="searchType = 'tcw'.toString()">
            and ( title like CONCAT('%', #{keyword}, '%')
                OR
                content like CONCAT('%', #{keyword}, '%')
                OR
                writer like CONCAT('%', #{keyword}, '%'))
        </if>
    </if>
</sql>

<select id="listSearch" resultType="BoardVO">
    <![CDATA[
        select *
        from tbl_board
        where bno > 0
    ]]>

    <include refid="search"></include>

    <![CDATA[
        order by bno desc
        limit #{pageStart}, #{perPageNum}
    ]]>
</select>

```

```

<select id="listSearchCount" resultType="int">
<![CDATA[
    select count(bno)
    from tbl_board
    where bno > 0
]]>
    <include refid="search"></include>
</select>

```

1.10.5 BoardService와 SearchBoardController의 수정

- BoardService 인터페이스는 아래와 같이 수정한다.

```

// org.zerock.service.BoardService.java의 일부

public List<BoardVO> listSearchCriteria(SearchCriteria cri) throws Exception;

public int listSearchCount(SearchCriteria cri) throws Exception;

```

- 인터페이스의 구현 클래스 역시 아래와 같이 수정한다.

```

// org.zerock.service.BoardServiceImpl.java의 일부

@Override
public List<BoardVO> listSearchCriteria(SearchCriteria cri) throws Exception {

    return dao.listSearch(cri);
}

@Override
public int listSearchCount(SearchCriteria cri) throws Exception {

    return dao.listSearchCount(cri);
}

```

- 컨트롤러는 아래와 같이 변경해야 한다.

```

// org.zerock.controller.SearchBoardController.java의 일부

@RequestMapping(value = "/list", method = RequestMethod.GET)
public void listPage(@ModelAttribute("cri") SearchCriteria cri, Model model) throws Exception {

    logger.info(cri.toString());

    // model.addAttribute("list", service.listCriteria(cri));
    model.addAttribute("list", service.listSearchCriteria(cri));

    PageMaker pageMaker = new PageMaker();
    pageMaker.setCri(cri);

    // pageMaker.setTotalCount(service.listCountCriteria(cri));
    pageMaker.setTotalCount(service.listSearchCount(cri));

    model.addAttribute("pageMaker", pageMaker);
}

```

- 실행: <http://localhost:8080/sboard/list?page=1&perPageNum=10&searchType=t&keyword=test>

1.10.6 조회, 수정, 삭제 페이지의 처리

```
//org.zerock.controller.SearchBoardController의 일부
@RequestMapping(value = "/modifyPage", method = RequestMethod.POST)
public String modifyPagingPOST(BoardVO board, SearchCriteria cri, RedirectAttributes rttr) throws Exception {
    logger.info(cri.toString());
    service.modify(board);

    rttr.addAttribute("page", cri.getPage());
    rttr.addAttribute("perPageNum", cri.getPerPageNum());
    rttr.addAttribute("searchType", cri.getSearchType());
    rttr.addAttribute("keyword", cri.getKeyword());

    rttr.addFlashAttribute("msg", "SUCCESS");

    logger.info(rttr.toString());

    return "redirect:/sboard/list";
}
```