

Spring과 Mybatis를 활용한 회원게시판

안지혁

01

서론

프로젝트 목적

프로젝트 개요

요구사항

02

본론

기능 및 주요 기술 분석

게시판 시스템의 기본구조

프로그램 실행

03

결론

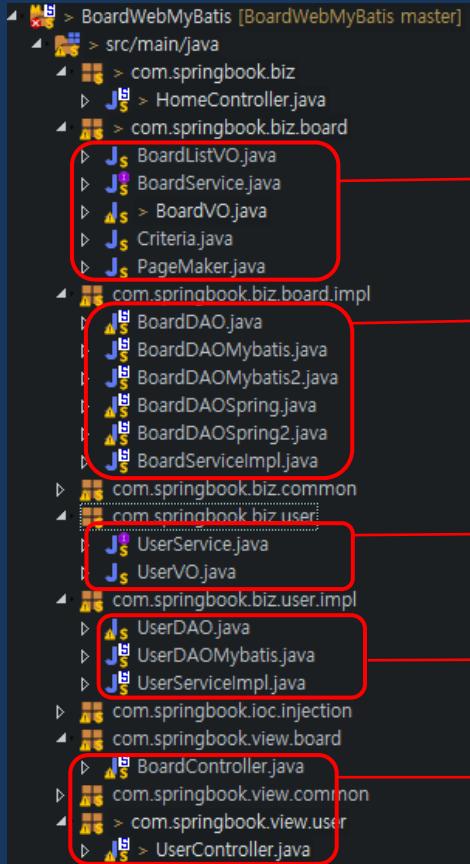
향후 개선 방향

프로젝트 후기

Index

- 회원가입 게시판을 구현하면서 Spring의 기본 개념인 IoC와 DI, AOP 개념을 이해하고 활용
- Spring MVC패턴을 이해하고 활용
- Spring의 3-tier 구조를 이해하고 활용
(Presentation Layer, Business Layer, Persistence Layer)
- Spring과 Mybatis와 연동을 통해 영속성을 구현

- 회원가입을 하여 로그인 하도록 한다.
- 회원은 글 등록, 수정 삭제를 할 수 있도록 한다.
- 회원은 자신의 정보를 수정 및 삭제(탈퇴) 가능 구현 한다.
- 회원가입 및 로그인은 3-tier 구조로 구현 한다.
- 게시판의 글 등록, 수정, 삭제, 검색 3-tier 구현한다.
- Presentation Layer는 Controller를 활용하여 구현한다.
- Business Layer는 DI, AOP를 활용하여 구현 한다.
- Persistence Layer는 Spring 과 Mybatis 활용 하여 구현

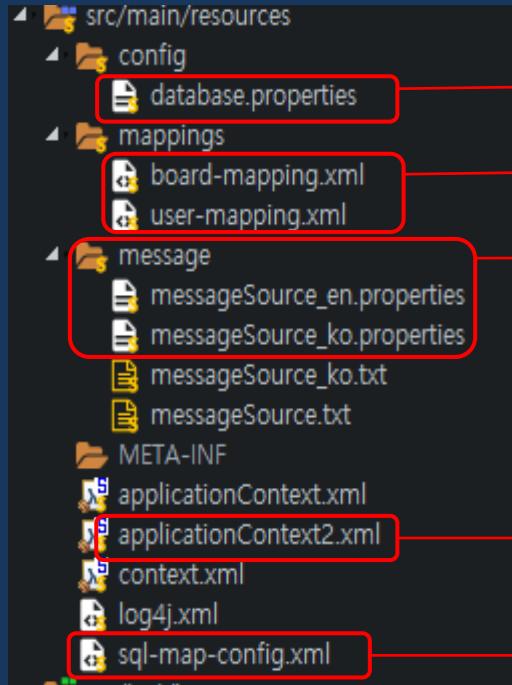


- BoardVO : 게시판 VO 클래스
- Criteria : 한 페이지에 표현할 개수
- PageMaker : 화면에 페이지 개수 설정
- BoardService : 게시판 Service Interface

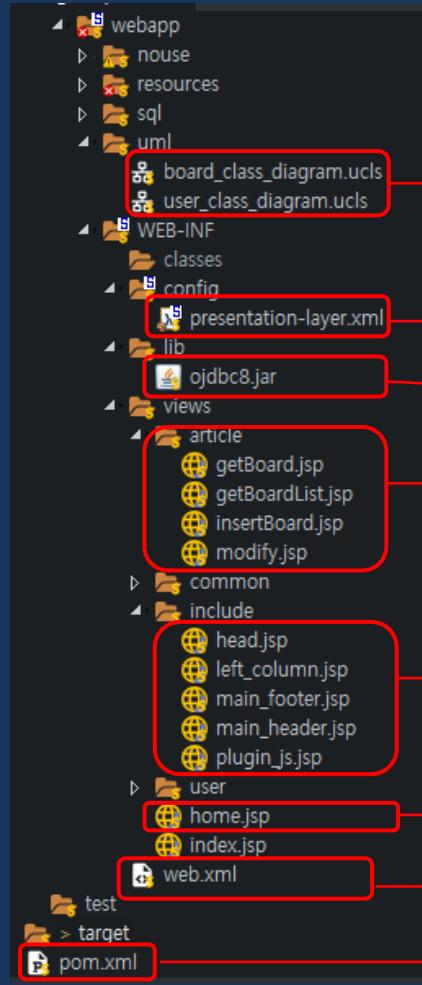
- BoardServiceImpl : BoardService 구현 클래스
- BoardDAOMybatis2 : 게시판 DAO 클래스(mybatis)

- UserVO : 회원 정보 VO 클래스
- UserService : 회원 가입, 조회, 탈퇴 Service 인터페이스
- UserServiceImpl : UserService 구현 클래스
- UserDAOMybatis : 회원 DAO 클래스(mybatis)

- UserController, BoardController : 회원, 게시판 Controller



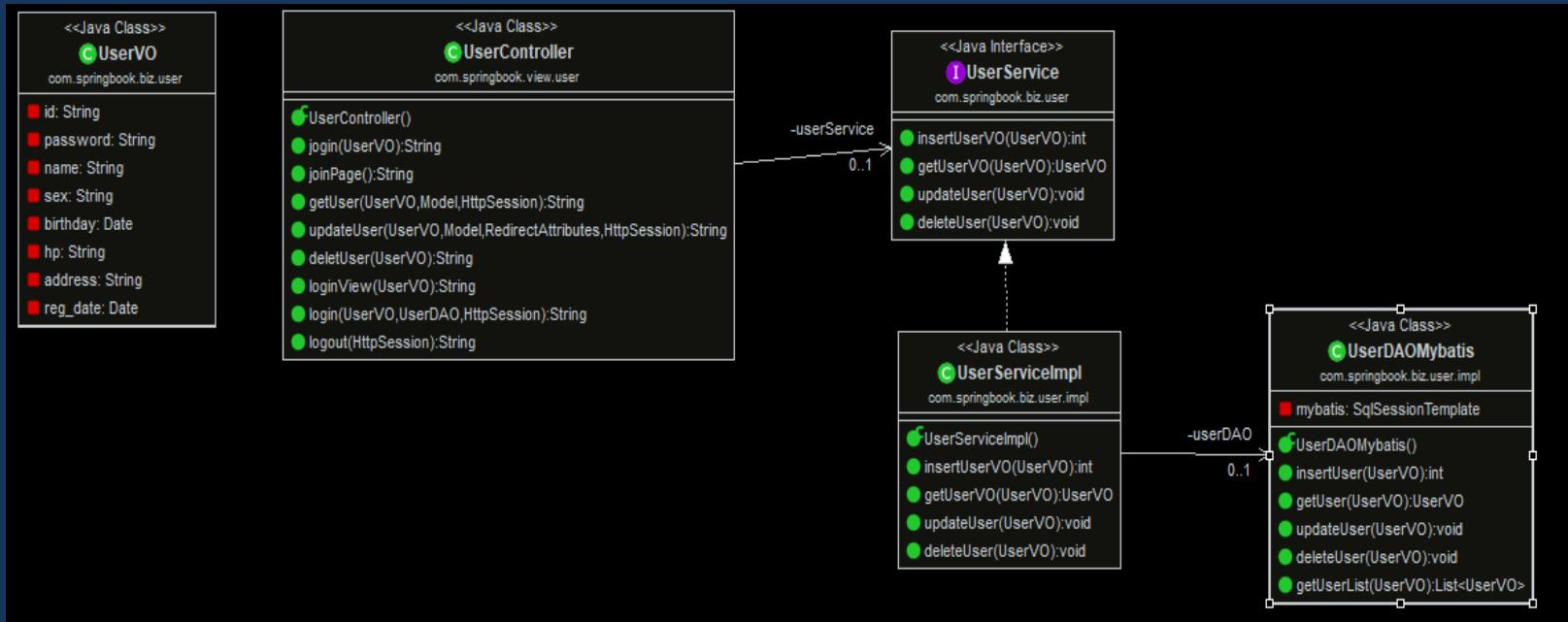
- Database.properties : applicationContext2.xml의 jdbc 설정
- *mapping.xml : ***DAOMybatis.java**에서 사용하는 mapper 파일
- 다국어 처리를 위한 설정 파일
- applicationContext2.xml : JDBC, Datasource, Transaction, Advice, AOP, Mybatis설정
- sql-map-config.xml : mybatis mapping설정



- UML 클래스 다이어 그램 생성 파일
- 파일업로드, 예외처리, 다국어 설정, 정적 파일 경로 지정, ViewResolver 설정
- Oracle DBMS를 사용하기 위한 라이브러리
- 게시판 View
- 전체 화면 구성 View
- 게시판 홈 화면 View
- Presentation-layer.xml, applicationContext.xml 위치 설정, encoding 설정 파일
- 각종 라이브러리 사용을 위한 Maven Dependency 설정

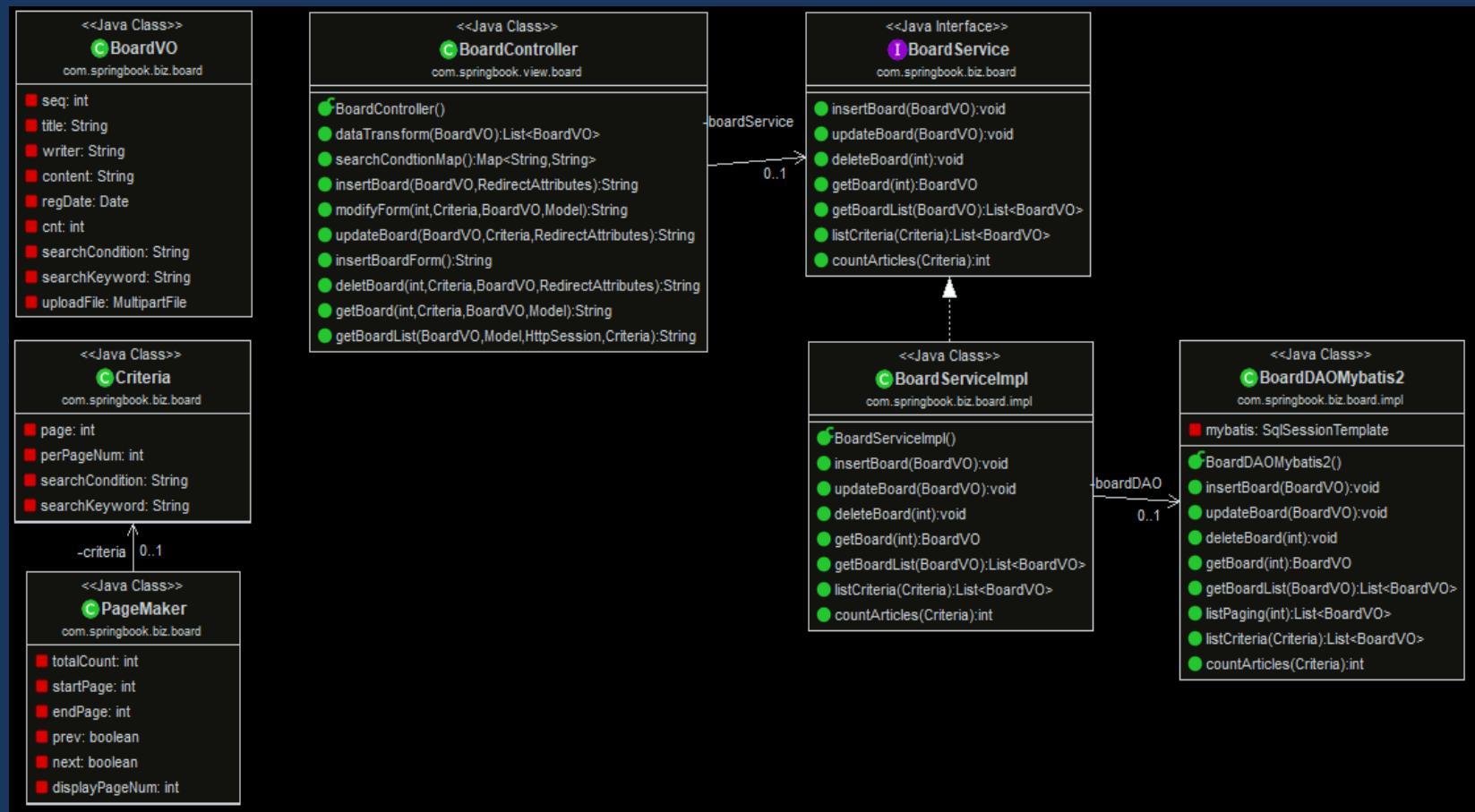
Business Partner

If we meet together, I can be your business partner.



Business Partner

If we meet together, I can be your business partner.



SCOTT.USERS	
P	* ID VARCHAR2 (30 BYTE)
*	PASSWORD VARCHAR2 (30 BYTE)
*	NAME VARCHAR2 (50 BYTE)
*	SEX VARCHAR2 (10 BYTE)
*	BIRTHDAY DATE
*	HP VARCHAR2 (13 BYTE)
*	ADDRESS VARCHAR2 (200 BYTE)
	REG_DATE DATE
	PK USERS_PK (ID)

```
CREATE TABLE users (
    id      VARCHAR2(30 BYTE) PRIMARY KEY,
    password  VARCHAR2(30 BYTE) NOT NULL ENABLE,
    name     VARCHAR2(50 BYTE) NOT NULL ENABLE,
    sex      VARCHAR2(10 BYTE) NOT NULL ENABLE,
    birthday  DATE NOT NULL ENABLE,
    hp       VARCHAR2(13 BYTE) NOT NULL ENABLE,
    address   VARCHAR2(200 BYTE)NOT NULL ENABLE,
    reg_date  DATE DEFAULT SYSDATE
);
```

1	ID	VARCHAR2 (30 BYTE)	No	(null)
2	PASSWORD	VARCHAR2 (30 BYTE)	No	(null)
3	NAME	VARCHAR2 (50 BYTE)	No	(null)
4	SEX	VARCHAR2 (10 BYTE)	No	(null)
5	BIRTHDAY	DATE	No	(null)
6	HP	VARCHAR2 (13 BYTE)	No	(null)
7	ADDRESS	VARCHAR2 (200 BYTE)	No	(null)
8	REG_DATE	DATE	Yes	sysdate

- users 테이블 명세서

SCOTT.BOARD	
P *	SEQ NUMBER (5)
	TITLE VARCHAR2 (200 BYTE)
	WRITER VARCHAR2 (20 BYTE)
	CONTENT VARCHAR2 (2000 BYTE)
	REGDATE DATE
	CNT NUMBER (5)
BOARD_PK(SEQ)	

```
CREATE TABLE board (
    seq      NUMBER(5, 0) PRIMARY KEY,
    title   VARCHAR2(200 BYTE),
    writer  VARCHAR2(20 BYTE),
    content VARCHAR2(2000 BYTE),
    regdate DATE DEFAULT SYSDATE,
    cnt     NUMBER(5, 0) DEFAULT 0
);
```

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT
1 SEQ	NUMBER (5, 0)	No	(null)
2 TITLE	VARCHAR2 (200 BYTE)	Yes	(null)
3 WRITER	VARCHAR2 (20 BYTE)	Yes	(null)
4 CONTENT	VARCHAR2 (2000 BYTE)	Yes	(null)
5 REGDATE	DATE	Yes	sysdate
6 CNT	NUMBER (5, 0)	Yes	0

- board 테이블 명세서

01 | 서론 | DB 테이블 데이터 조회 결과

```
select * from users;
```

ID	PASSWORD	NAME	SEX	BIRTHDAY	HP	ADDRESS	REG_DATE
1	test1	1234	안지혁	female	1988-06-21	010-4197-9322	대구시 북구 호암로 40 삼성파트 101동 1604호 2019-05-20

- users 테이블 검색 결과

```
select * from board;
```

SEQ	TITLE	WRITER	CONTENT	REGDATE	CNT
1	2 교보생명 합격 후기	안지혁	스펙: ITQ한글, ITQ엑셀, ITQ...	2019-05-20	2
2	3 한국전력공사 최종합격 수기입니다.	안지혁	→ 토익 875 쌍기사 컴활1급 한...	2019-05-20	4
3	4 삼성전자 최종합격후기	안지혁	먼저 제 스펙을 말씀 드리겠습니다...	2019-05-20	3
4	5 현대모비스 최종후기입니다. [출처]...	안지혁	현대모비스 면접보고 왔습니다.	2019-05-20	16
5	1 한국농어촌공사 합격 수기입니다~ [...]	안지혁	<p>우선 저의 스펙은 나이 : 25...	2019-05-20	6
6	6 문과 중고신입 2018 LG화학 합격 후기	kim7	1 스펙-환경외시 어문계열-학점 ...	2019-05-21	4

- board 테이블 검색 결과

```

<!-- Spring -->
<!-- lombok -->
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.8</version>
    <scope>provided</scope>
</dependency>

<!-- mybatis-spring -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>2.0.1</version>
</dependency>

<!-- mybatis -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.5.1</version>
</dependency>

```

- mybatis, mybatis-spring : mybatis와 스프링을 연동 하기 위한 API
- Lombok : vo클래스에서 getter,setter 생성해주는 API

```

40      <!-- file upload -->
41      <dependency>
42          <groupId>commons-fileupload</groupId>
43          <artifactId>commons-fileupload</artifactId>
44          <version>1.4</version>
45      </dependency>
46
47      <!-- spring-jdbc -->
48      <dependency>
49          <groupId>org.springframework</groupId>
50          <artifactId>spring-jdbc</artifactId>
51          <version>${org.springframework-version}</version>
52      </dependency>
53
54      <!-- commons-dbc -->
55      <dependency>
56          <groupId>commons-dbc</groupId>
57          <artifactId>commons-dbc</artifactId>
58          <version>1.4</version>
59      </dependency>
60

```

- spring-jdbc : 스프링에서 jdbc를 사용하기 위한 API
- commons-dbc : connection pool을 사용하기 위한 API
- commons-fileupload : 파일 업로드를 위한 API

02 본론 | 프로젝트 설정 파일 – web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  https://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  version="2.5">
  <servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>WEB-INF/config/presentation-layer.xml</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>

  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext2.xml</param-value>
  </context-param>
  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>

  <filter>
    <filter-name>characterEncoding</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>characterEncoding</filter-name>
    <url-pattern>*.do</url-pattern>
  </filter-mapping>
</web-app>
```

- ContextLoaderListner : WebServlet컨테이너 생성 후 스프링에서 DispatcherServlet을 생성하하기 전 applicationContext2.xml파일을 로딩하여 스프링 컨테이너 구동 : Service 구현 객체나 DAO 객체 메모리 생성
- DispatcherServlet : *.do의 요청이 들어 오면 두번째 스프링 컨테이너를 생성 하며 presentation-layer.xml 설정 되어 있는 bean객체들을 생성 : controller 객체를 메모리에 생성
- filter : controller로 들어오는 모든 *.do요청에 대해서 UTF-8 encoding으로 처리하기 위한 설정

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd
                           http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
                           http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- ViewResolver 등록 -->

    <bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/views/" />
        <property name="suffix" value=".jsp" />
    </bean>

    <!-- 2. 어노테이션 기반 설정 -->
    <context:component-scan base-package="com.springbook" />
```

<!-- 파일 업로드 설정 -->

```
<bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
    <property name="maxUploadSize" value="1000000" />
</bean>
```

<!-- 예외 처리 설정 : ControllerAdvice, ExceptionHandler -->

<!-- 어노테이션 기반 설정 -->

```
<mvc:annotation-driven />
```

viewResolver, 어노테이션 인식 설정, 파일 업로드,

```
<!-- xml 기본 예외처리 -->
<bean id="exceptionResolver" class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">
    <property name="exceptionMappings">
        <props>
            <prop key="java.lang.ArithmetricException">
                common/arithmeticError
            </prop>

            <prop key="java.lang.NullPoinerException">
                common/nullPointerError
            </prop>
        </props>
    </property>
    <property name="defaultErrorView" value="common/error"/>
</bean>

<!-- 다국어 설정 -->
<!-- MessageSoruce 등록 -->
<bean id="messageSource" class="org.springframework.context.support.ResourceBundleMessageSource">
    <property name="basenames">
        <list>
            <value>message.messageSource</value>
        </list>
    </property>
</bean>
```

예외처리 설정, 다국어 설정

```
<!-- LocaleResolver 등록 -->
<bean id="localeResolver" class="org.springframework.web.servlet.i18n.SessionLocaleResolver"></bean>

<!-- LocaleChangeInerceptor 등록 -->
<mvc:interceptors>
    <bean class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor">
        <property name="paramName" value="lang"></property>
    </bean>
</mvc:interceptors>

<!-- Handles HTTP GET requests for /resources/** by efficiently serving up static resources in the ${webappRoot}/resources directory -->
<mvc:resources mapping="/bower_components/**" location="/resources/bower_components/" />
<mvc:resources mapping="/plugins/**" location="/resources/plugins/" />
<mvc:resources mapping="/dist/**" location="/resources/dist/" />

</beans>
```

Locale resolver, interceptor 설정, 정적 파일 설정

```

applicationContext2.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xmlns:context="http://www.springframework.org/schema/context"
5   xmlns:tx="http://www.springframework.org/schema/tx"
6   xmlns:aop="http://www.springframework.org/schema/aop"
7   xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/
8     http://www.springframework.org/schema/context http://www.springframework.org/schema/context
9     http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-
10    http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx
11
12
13 <context:component-scan base-package="com.springbook.biz"/>
14 <!-- Datasource 설정 -->
15
16 <context:property-placeholder location="classpath:config/database.properties"/>
17
18 <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
19   <property name="driverClassName" value="${jdbc.driver}"/>
20   <property name="url" value="${jdbc.url}"/>
21   <property name="username" value="${jdbc.username}"/>
22   <property name="password" value="${jdbc.password}"/>
23 </bean>
24
25 <!-- Sping2 jdbc 설정 -->
26 <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
27   <property name="dataSource" ref="dataSource"></property>
28 </bean>
29

```

- Datasource, component-scan 설정
- 스프링과 jdbc설정

```
<!-- Transaction 설정 -->
<bean id="txManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource"></property>
</bean>

<!-- Transaction advice 설정 -->
<tx:advice id="txAdvice" transaction-manager="txManager">
    <tx:attributes>
        <tx:method name="get*" read-only="true"/>
        <tx:method name="*"/>
    </tx:attributes>
</tx:advice>

<!-- Transaction aop(aspect) 설정 -->
<aop:config>
    <aop:pointcut id="txPointcut" expression="execution(* com.springbook.biz..*Impl.*(..))" />

    <aop:advisor pointcut-ref="txPointcut" advice-ref="txAdvice"/>
</aop:config>

<!-- Spring과 Mybatis 연동 설정 -->
<bean id="sqlSession" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource"/>
    <property name="configLocation" value="classpath:sql-map-config.xml"></property>
</bean>

<!-- SqlSessionTemplate Constructor주입(생성자 주입) -->
<bean class="org.mybatis.spring.SqlSessionTemplate">
    <constructor-arg ref="sqlSession"></constructor-arg>
</bean>
</beans>
```

- 트랜잭션 설정
- Spring과 Mybatis 연동 설정
- SqlSessionFactory 설정(Datasource를 참조, sql-map-config.xml 경로 설정)

02 | 본론 | 프로젝트 기술 분석 - UserVO

```
package com.springbook.biz.user;

import java.sql.Date;
import lombok.Data;

@Data
public class UserVO {
    private String id;
    private String password;
    private String name;
    private String sex;
    private Date birthday;
    private String hp;
    private String address;
    private Date reg_date;
}
```

```
‘ C UserVO
■ id : String
■ password : String
■ name : String
■ sex : String
■ birthday : Date
■ hp : String
■ address : String
■ reg_date : Date
● getid() : String
● setId(String) : void
● getPassword() : String
● setPassword(String) : void
● getName() : String
● setName(String) : void
● getSex() : String
● setSex(String) : void
● getBirthday() : Date
● setBirthday(Date) : void
● getHp() : String
● setHp(String) : void
● getAddress() : String
● setAddress(String) : void
● getReg_date() : Date
● setReg_date(Date) : void
● toString() : String
```

- 회원정보 데이터를 처리 시 사용하는 VO(Value Object)
- *Lombok 라이브러리 사용으로 getter/setter 생략 가능

```
package com.springbook.view.user;
import javax.servlet.http.HttpSession;

@Controller
public class UserController {
    // 1. JoinController
    @Autowired
    private UserService userService;

    @RequestMapping(value="/join.do", method=RequestMethod.POST)
    public String jlogin(UserVO vo) {
        System.out.println(vo);
        int join = 0;
        join = userService.insertUserVO(vo);
        if(join == 1) {                                //회원가입 성공
            return "redirect:home.do";
        } else {
            return "/user/join";                      //회원 가입 실패
        }
    }
    //다국어 처리 설정 때문에 Controller를 거쳐야 함.
    @RequestMapping(value="/joinPage.do", method=RequestMethod.GET)
    public String joinPage() {
        return "/user/join";
    }
}
```

- `@Controller` : `applicationContext`에서 `component-scan`에서 읽어 스캔 하기 위한 어노테이션(Annotation)
- `Autowired` : DI(Dependency Injection)를 하기 위한 어노테이션
- 회원가입 버튼을 클릭시 요청하는 controller와 회원가입 페이지를 요청했을 때

```
// 2.loginController
@RequestMapping(value="/login.do", method=RequestMethod.GET)
public String loginView(@ModelAttribute("user") UserVO vo) {
    System.out.println("로그인 화면으로 이동");
    vo.setId("test");
    vo.setPassword("1234");
    return "/user/login";
}

@RequestMapping(value="/login.do", method=RequestMethod.POST)
public String login(UserVO vo, UserDAO userDAO, HttpSession session) {
    System.out.println("로그인 인증 처리");

    if(vo.getId() == null || vo.getId().equals("")) {
        throw new IllegalArgumentException("아이디는 반드시 입력해야 합니다.");
    }
    if(vo.getPassword() == null || vo.getPassword().equals("")) {
        throw new IllegalArgumentException("패스워드는 반드시 입력해야 합니다.");
    }

    UserVO user = userDAO.getUser(vo);

    if(user != null) {
        session.setAttribute("userName", user.getName());
        session.setAttribute("userId", user.getId());
        return "redirect:home.do";
    } else {
        return "redirect:login.do";
    }
}
```

```
// 3. LogoutController
@RequestMapping("/logout.do")
public String logout(HttpSession session) {
    System.out.println("로그아웃 처리");

    session.invalidate();
    return "redirect:home.do";
}
```

- 로그인 페이지로 (get)방식으로 이동할 때 로그인 할 때(post) 요청되는 컨트롤러
- 로그아웃 요청 시 세션 해제

```
@RequestMapping("/getUser.do")
public String getUser(UserVO vo, Model model, HttpSession session) {
    System.out.println("개인정보 조회");
    String auth = (String)session.getAttribute("userName");

    if(auth == null && auth.equals("")) {
        return "redirect:login.do";
    } else {
        model.addAttribute("user", userService.getUserVO(vo));
        return "/user/joinInfo";
    }
}

@RequestMapping(value="/updateUser.do", method=RequestMethod.POST)
public String updateUser(@ModelAttribute("user") UserVO vo, Model model
    , RedirectAttributes redirectAttributes, HttpSession session) {
    System.out.println("개인정보 수정");
    userService.updateUser(vo);
    redirectAttributes.addAttribute("id", session.getAttribute("userId"));
    return "redirect:getUser.do";
}

@RequestMapping("/deleteUser.do")
public String deleteUser(UserVO vo) {
    System.out.println("회원 탈퇴 처리");
    userService.deleteUser(vo);
    return "redirect:login.do";
}
```

- 회원 정보 확인, 수정, 삭제 요청을 받았을때 동작하는 controller

```
package com.springbook.biz.user;

public interface UserService {

    //CRUD 기능의 메소드 구현
    //회원 등록
    public int insertUserVO(UserVO vo);

    //회원 조회
    UserVO getUserVO(UserVO vo);

    //회원 수정
    void updateUser(UserVO vo);

    //회원탈퇴
    void deleteUser(UserVO vo);

}
```

```
@Service("userService")
public class UserServiceImpl implements UserService{

    @Autowired
    private UserDAOMybatis userDAO;

    @Override
    public int insertUserVO(UserVO vo) {
        int joinCheck = 0;
        joinCheck = userDAO.insertUser(vo);
        return joinCheck;
    }

    @Override
    public UserVO getUserVO(UserVO vo) {
        return userDAO.getUser(vo);
    }

    @Override
    public void updateUser(UserVO vo) {
        userDAO.updateUser(vo);
    }

    @Override
    public void deleteUser(UserVO vo) {
        userDAO.deleteUser(vo);
    }
}
```

- 회원 관련 비지니스 로직 구현
- @Service : applicationContext에서 component-scan에서 읽어 스캔하기 위한 어노테이션(Annotation)
- Autowired : DI(Dependency Injection)를 하기 위한 어노테이션

```

@Repository
public class UserDAOMybatis {
    @Autowired
    private SqlSessionTemplate mybatis;

    // crud 기능의 메소드 구현
    // 회원 가입
    public int insertUser(UserVO vo) {
        System.out.println("==> Mybatis JDBC로 insertUser() 기능 처리");
        int check = 0;
        check = mybatis.insert("UserDAO.insertUser", vo);
        System.out.println(check);
        return check;
    }

    // 회원정보 조회
    public UserVO getUser(UserVO vo) {
        System.out.println("==> Mybatis JDBC로 getUser() 기능 처리");
        return (UserVO) mybatis.selectOne("UserDAO.getUser", vo);
    }

    // 회원 정보 수정
    public void updateUser(UserVO vo) {
        System.out.println("==> Mybatis JDBC로 updateUser() 기능 처리");
        System.out.println(vo);
        mybatis.update("UserDAO.updateUser", vo);
    }

    // 회원 탈퇴
    public void deleteUser(UserVO vo) {
        System.out.println("==> Mybatis JDBC로 deleteUser() 기능 처리");
        mybatis.delete("UserDAO.deleteUser", vo);
    }
}

```

- persistence 계층 로직 : 주로 데이터의 생성/수정/삭제/선택(검색)과 같은 CRUD 연산을 수행
- @Repository : applicationContext2.xml에서 component-scan에서 읽어 스캔하기 위한 어노테이션(Annotation)
- Mybatis를 연동하기 위해 SqlSessionTemplate 를 DI(Dependency Injection) 하기 위한 어노테이션
- resources/mapping/user-mapping.xml 에 있는 SQL들과 매핑되는 메소드

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

    <!-- Alias 설정 -->
    <typeAliases>
        <typeAlias alias="board" type="com.springbook.biz.board.BoardVO"/>
        <typeAlias alias="user" type="com.springbook.biz.user.UserVO"/>
    </typeAliases>

    <!-- SQL Mapper 설정 -->
    <mappers>
        <mapper resource="mappings/board-mapping.xml"/>
        <mapper resource="mappings/user-mapping.xml"/>
    </mappers>
</configuration>
```

- typeAliases : BoardVO 클래스의 Alias를 board로 설정, Sql Mapper에서 사용 가능
<typeAlias>를 여러 개 사용 가능
- mappers : 게시판, 회원 테이블을 위한 SQL 파일들, <mapper> 이용해서 등록 가능

```

<?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="UserDAO">
    <!-- ORAM을 위한 설정 -->
    <resultMap type="user" id="boardResult">
        <id property="id" column="ID"/>
        <result property="password" column="PASSWORD"/>
        <result property="name" column="NAME"/>
        <result property="sex" column="SEX"/>
        <result property="birthday" column="BIRTHDAY"/>
        <result property="hp" column="HP"/>
        <result property="address" column="ADDRESS"/>
        <result property="reg_date" column="REG_DATE"/>
    </resultMap>

    <!-- 회원가입 -->
    <insert id="insertUser">
        <![CDATA[
            INSERT INTO USERS(id, password, name, sex, birthday, hp, address, reg_date)
            VALUES(#{id}, #{password}, #{name}, #{sex}, #{birthday}, #{hp}, #{address}, sysdate)
        ]]>
    </insert>

    <!-- 회원 정보 조회 -->
    <select id="getUser" resultType="user">
        <![CDATA[
            SELECT * FROM USERS WHERE ID=#{id}
        ]]>
    </select>

```

sql 명령어들이 저장되는 SQL Mapper XML

- <mapper> 엘리먼트의 namespace는 유일한 SQL 아이디를 만들때 각각의 SQL문의 id 와 조합하여 사용 (DAO클래스)
ex) mybatis.insert("UserDAO.insertUser", vo);
- resultMap : 검색 결과를 특정 자바 객체에 매칭하여 리턴 할 때 join이나 검색 결과의 컬럼 값이 다를 때 매핑 하기 위한 방법
- <insert> 엘리먼트 : insert 구문 작성하는 요소
- <select> 엘리먼트 : select 구문 작성하는 요소
- resultType : SQL결과를 resultMap과 매핑 하기위한 속성

```

<!-- 회원 정보 수정 -->
• <update id="updateUser" parameterType="user">
  <![CDATA[
    UPDATE USERS
      SET PASSWORD=#{password},
          NAME=#{name},
          SEX=#{sex},
          BIRTHDAY=#{birthday},
          HP=#{hp},
          ADDRESS=#{address}
    WHERE ID=#{id}
  ]]>

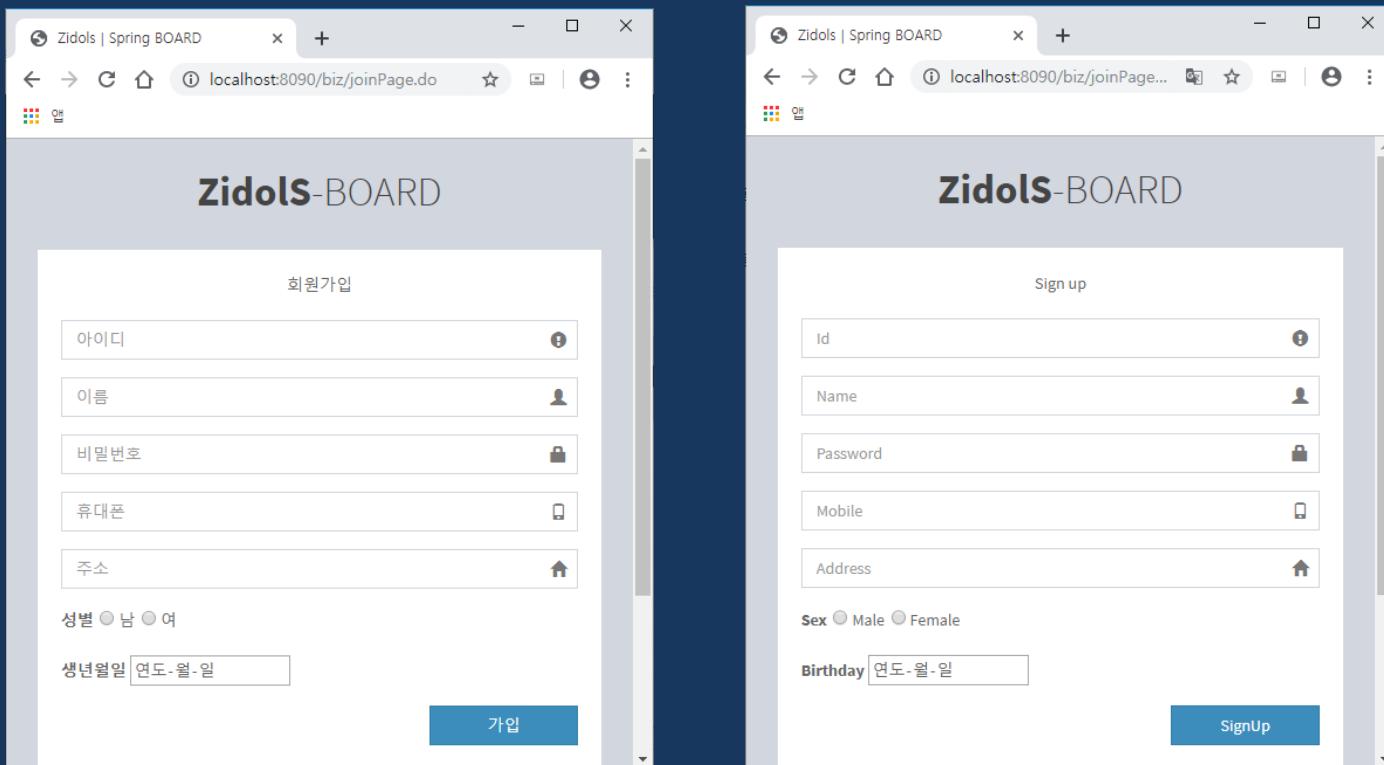
</update>

<!-- 회원 탈퇴 -->
• <delete id="deleteUser">
  <![CDATA[
    DELETE USERS WHERE ID=#{id}
  ]]>
</delete>
</mapper>

```

sql 명령어들이 저장되는 SQL Mapper XML

- <mapper> 엘리먼트의 namespace는 유일한 SQL 아이디를 만들때 각각의 SQL문의 id 와 조합하여 사용 (DAO클래스)
ex) mybatis.insert("UserDAO.insertUser", vo);
- resultMap : 검색 결과를 특정 자바 객체에 매칭하여 리턴 할 때 join이나 검색 결과의 컬럼 값이 다를 때 매핑 하기 위한 방법
- <insert> 엘리먼트 : insert 구문 작성하는 요소



회원가입 페이지 화면(Bootstrap 사용)

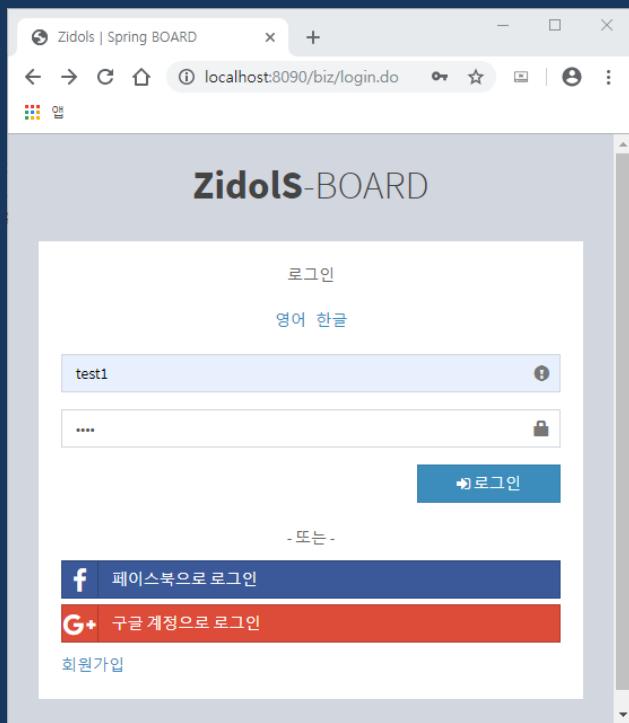
```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<!DOCTYPE html>
<html>
    <%@ include file="../include/head.jsp" %>
    <body class="hold-transition register-page">
        <div class="register-box">
            <div class="register-logo">
                <a href="${path}/home.do">
                    <b>Zidols</b>-BOARD
                </a>
            </div>

            <div class="register-box-body">
                <p class="login-box-msg">회원가입 페이지</p>

                <form action="join.do" method="post"> 회원 가입 처리를 위해 joind.do로 UserController에 호출
                    <div class="form-group has-feedback">
                        <input type="text" name="id" class="form-control" placeholder=<spring:message code="message.board.join.id"/>>
                        <span class="glyphicon glyphicon-exclamation-sign form-control-feedback"></span>
                    </div>
                    <div class="form-group has-feedback">
                        <input type="text" name="name" class="form-control" placeholder=<spring:message code="message.board.join.name"/>>
                        <span class="glyphicon glyphicon-user form-control-feedback"></span>
                    </div>
                    <div class="form-group has-feedback">
                        <input type="password" name="password" class="form-control" placeholder=<spring:message code="message.board.join.password"/>>
                        <span class="glyphicon glyphicon-lock form-control-feedback"></span>
                    </div>
                    <div class="col-xs-4">
                        <input type="submit" class="btn btn-primary btn-block btn-flat" value=<spring:message code="message.board.join.signupBtn"/>>
                    </div>
                <div style="text-align: center;">::
```

02 본론 | 프로젝트 기술 분석 – JSP 로그인



로그인 페이지 화면

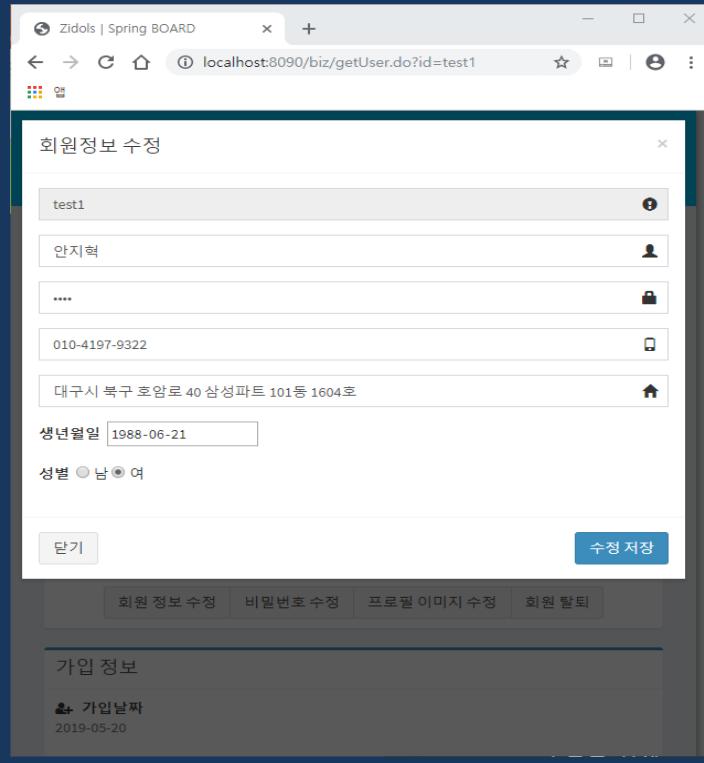
```
<div class="login-box-body">
    <p class="login-box-msg"><spring:message code="message.user.login.title"/></p>
    <p class="login-box-msg">
        <a href="login.do?lang=en"><spring:message code="message.user.login.language.en"/></a>&ampnbsp&ampnbsp
        <a href="login.do?lang=ko"><spring:message code="message.user.login.language.ko"/></a>
    </p>
    <form action="login.do" method="post">    다국어 설정을 위한 a 태그 버튼
        <div class="form-group has-feedback">
            <input type="text" name="id" class="form-control" placeholder=<spring:message code="message.user.login.id"/>>
            <span class="glyphicon glyphicon-exclamation-sign form-control-feedback"></span>
        </div>
        <div class="form-group has-feedback">
            <input type="password" name="password" class="form-control" placeholder=<spring:message code="message.user.login.password"/>>
            <span class="glyphicon glyphicon-lock form-control-feedback"></span>
        </div>
        <div class="row">
            <div class="col-xs-8">
                <div>
                </div>
            </div>
            <div class="col-xs-4">
                <button type="submit" class="btn btn-primary btn-block btn-flat">
                    <i class="fa fa-sign-in"></i> <spring:message code="message.user.login.loginBtn"/>
                </button>
            </div>
        </div>
    </form>
```

로그인 요청시 login.do로 RequestMapping된 메소드로 실행

02 본론 | 프로젝트 기술 분석 – JSP 로그인



회원정보 수정 페이지



회원정보 수정 버튼 클릭 시 팝업 화면에서 수정

게시판에서 사용하는 데이터 VO클래스

```
1 package com.springbook.biz.board;  
2  
3 import java.sql.Date;  
4  
5 public class BoardVO {  
6     private int seq;  
7     private String title;  
8     private String writer;  
9     private String content;  
10    private Date regDate;  
11    private int cnt;  
12    private MultipartFile uploadFile;  
13    private String originalFileName;  
14    private String saveFileName;
```

- seq : 글 번호
- title : 글 제목
- writer : 작성자
- content : 글 내용
- regDate : 작성 날짜
- cnt : 게시글 조회수
- uploadFile : 파일 업로드
- originalFileName : 업로드 파일 이름
- saveFileName : 업로드 파일 이름 변경한 이름

02 | 본론 | 프로젝트 기술 분석 – Criteria

```
BoardController.java  J Criteria.java ✘ J PageMaker.java  Board
1 package com.springbook.biz.board;
2
3 public class Criteria {
4     private int page;
5     private int perPageNum;
6     private String searchCondition;
7     private String searchKeyword;
8
9     public Criteria() {
10         this.page = 1;
11         this.perPageNum = 10;
12     }
13     //처음 페이지번호 설정
14     public void setPage(int page) {
15
16         if (page <= 0) {
17             this.page = 1;
18             return;
19         }
20
21         this.page = page;
22     }
23
24     public int getPage() {
25         return page;
26     }
27     //페이지 당 보여주는 마지막 번호 설정
28     public void setPerPageNum(int perPageNum) {
29         this.perPageNum = page*10;
30     }
31
32     public int getPerPageNum() {
33         return this.perPageNum;
34     }
35     //페이지당 시작 row번호 설정
36     public int getPageStart() {
37         return (this.page - 1) * 10+1;
38     }
39 }
```

게시판에서 페이지 처리시 사용하는 데이터

- page : 페이지 번호
- perPageNum : 페이지당 보여주는 마지막 번호(sql)에서 사용
- searchCondition, searchKeyWord : 검색 조건, 검색 키워드(검색 후 페이지 처리 시 사용)

```

9 public class PageMaker {
10     private int totalCount;
11     private int startPage;
12     private int endPage;
13     private boolean prev;
14     private boolean next;
15
16     private int displayPageNum = 10;
17
18     private Criteria criteria;
19
20     public void setCriteria(Criteria criteria) {
21         this.criteria = criteria;
22     }
23
24     public void setTotalCount(int totalCount) {
25         this.totalCount = totalCount;
26         calcData();
27     }
28
29     private void calcData() {
30         //페이지 마지막 번호 = Math.ceil(현재페이지 / 페이지 번호의 갯수) * 페이지 번호의 갯수
31         endPage = (int) (Math.ceil(criteria.getPage()) / (double) displayPageNum) * displayPageNum;
32         //시작 페이지 번호 = (끝 페이지 번호 - 페이지 번호의 갯수) + 1
33         startPage = (endPage - displayPageNum) + 1;
34
35         int tempEndPage = (int) (Math.ceil(totalCount / (double) 10));
36
37         if (endPage > tempEndPage) {
38             endPage = tempEndPage;
39         }
40         //이전, 다음 버튼 출력시 사용
41         prev = startPage == 1 ? false : true;
42
43         next = endPage * 10 >= totalCount ? false : true;
44

```

*게시판 리스트에 보여줄 페이지 번호 설정 해주는 클래스

DB에서 계산되는 데이터

- totalCount : 전체 게시글의 갯수

계산식을 통해 만들어지는 데이터

- startPage : 시작 페이지 번호
- endPage : 끝 페이지 번호
- prev : 이전 링크
- next : 다음 링크
- displayNum : 페이지 번호 수

- 페이지 계산(예시)

```
int endPage = (int) (Math.ceil(criteria.getPage() / (double) displayPageNum) * displayPageNum);
```

페이지 번호의 갯수	현재 페이지	계산식	끝 페이지 번호
10	3	Math.ceil(3/10) * 10	10
10	1	Math.ceil(1/10) * 10	10
10	20	Math.ceil(20/10) * 10	20
10	21	Math.ceil(21/10) * 10	30
20	20	Math.ceil(20/20) * 20	20
20	21	Math.ceil(21/20) * 20	40
10	3	Math.ceil(3/10) * 10	10

- 페이지 계산(예시)

```
int startPage = (endPage - displaypageNum) + 1;
```

끝 페이지 번호	페이지 번호의 갯수	계산식	시작 페이지 번호
10	10	$(10-10)+1$	1
20	10	$(20-10)+1$	11
30	10	$(30-10)+1$	21
40	20	$(40-20)+1$	21
60	30	$(40-20)+1$	31

```
.. //uri생성 메서드
47     public String makeQuery(int page) {
48         UriComponents uriComponents = UriComponentsBuilder.newInstance()
49             .queryParam("page", page)
50             .queryParam("perPageNum", criteria.getPerPageNum())
51             .queryParam("searchCondition", (((Criteria) criteria).getSearchCondition()))
52             .queryParam("searchKeyword", encoding(((Criteria) criteria).getSearchKeyword())))
53             .build();
54
55         return uriComponents.toUriString();
56     }
57
58
59     private String encoding(String keyword) {
60         if (keyword == null || keyword.trim().length() == 0) {
61             return "";
62         }
63
64         try {
65             return URLEncoder.encode(keyword, "UTF-8");
66         } catch (UnsupportedEncodingException e) {
67             return "";
68         }
69     }
70 }
```

- UriComponents : 스프링에서 제공하는 클래스
- 화면 이동시 가져가는 데이터들을 URI를 생성함

```

@Controller
@SessionAttributes("board")
public class BoardController {
    @Autowired
    private BoardService boardService;

    // 글 등록 화면 호출
    @RequestMapping(value="/insertBoardForm.do", method=RequestMethod.GET)
    public String insertBoardForm(HttpServletRequest session) {
        String userId = (String)session.getAttribute("userId");
        System.out.println(userId);
        return "/article/insertBoard";
    }

    // 글 등록 시 호출
    @RequestMapping(value="/insertBoard.do")      //value=생략 가능
    public String insertBoard(BoardVO vo, RedirectAttributes redirectAttributes
            , HttpServletRequest request) throws IOException {
        System.out.println("글 등록 처리");

        Map<String, String> fileName = null;
        if(vo.getUploadFile() != null) {
            fileName= boardService.fileUpload(vo, request);
            vo.setOriginalFileName(fileName.get("originalFileName"));
            vo.setSaveFileName(fileName.get("saveFileName"));
        }

        String temp = vo.getContent();
        temp = temp.replaceAll("\r\n", "");
        vo.setContent(temp);
        boardService.insertBoard(vo);
        redirectAttributes.addFlashAttribute("msg", "regSuccess");
        return "redirect:getBoardList.do";
    }
}

```

- @Controller : applicationContext에서 component-scan에서 읽어 스캔 하기 위한 어노테이션(Annotation)
- Autowired : DI(Dependency Injection)를 하기 위한 어노테이션
- /insertBoardForm.do : 글 등록 화면으로 이동 시 다국어 처리로 인해 화면을 요청 받는 컨트롤러가 필요
- /insertBoard.do : 글 등록 요청 시 동작 하는 컨트롤러
- 파일 업로드, 글을 작성함
- temp.replaceAll("\r\n", ""); : 에디터로 인해 생기는 공백을 제거 해주기위해 사용
- redirectAttritiutes.addFlashAttrivute : 글 작성 후 메시지를 전달 하기 위해 사용, 리다이렉트 방식으로 전송을 해도 데이터가 전송 됨

```

BoardController.java ✘ Criteria.java PageMaker.java BoardService.java BoardServiceImpl.java BoardDAOMybat
 53
 54 // 글 수정 화면 호출
 55 @RequestMapping(value="/modifyForm.do", method=RequestMethod.GET)
 56 public String modifyForm(@RequestParam("seq") int seq,
 57     @ModelAttribute("criteria") Criteria criteria, BoardVO vo,
 58     Model model, HttpSession session) {
 59     session.getAttribute("userId");
 60     model.addAttribute("board", boardService.getBoard(seq));
 61     return "/article/modify";
 62 }
 63
 64 // 글 수정 작업 호출
 65 @RequestMapping(value="/modify.do", method=RequestMethod.POST)
 66 public String updateBoard(@ModelAttribute("board") BoardVO vo, Criteria criteria,
 67     RedirectAttributes redirectAttributes) {
 68     String temp = vo.getContent();
 69     temp = temp.replaceAll("\r\n", "");
 70     vo.setContent(temp);
 71
 72     boardService.updateBoard(vo);
 73     redirectAttributes.addAttribute("page", criteria.getPage());
 74     redirectAttributes.addAttribute("perPageNum", criteria.getPerPageNum());
 75     redirectAttributes.addAttribute("searchCondition", criteria.getSearchCondition());
 76     redirectAttributes.addAttribute("searchKeyword", criteria.getSearchKeyword());
 77     redirectAttributes.addFlashAttribute("msg", "modSuccess");
 78     return "redirect:getBoardList.do";
 79 }
 80
 81 // 글 삭제 처리
 82 @RequestMapping("/deleteBoard.do")
 83 public String deleteBoard(@RequestParam("seq") int seq, Criteria criteria,
 84     BoardVO vo, RedirectAttributes redirectAttributes) {
 85     System.out.println("글 삭제 처리");
 86     redirectAttributes.addAttribute("page", criteria.getPage());
 87     redirectAttributes.addAttribute("perPageNum", criteria.getPerPageNum());
 88     redirectAttributes.addAttribute("searchCondition", criteria.getSearchCondition());
 89     redirectAttributes.addAttribute("searchKeyword", criteria.getSearchKeyword());
 90     redirectAttributes.addFlashAttribute("msg", "delSuccess");
 91     boardService.deleteBoard(seq);
 92     return "redirect:getBoardList.do";
 93 }
 94
 95
 96
 97
 98
 99
100
101
102
103

```

- /modifyForm.do : 글 등록 화면으로 이동 시 다국어 처리로 인해 화면을 요청 받는 컨트롤러가 필요
- @RequestParam("seq") : jsp에서 seq값을 받아 올 수 있는 어노테이션 작성 글을 다시 불러오기 위한 글 번호
- @ModelAttribute("criteria") : 스프링 컨테이너가 세션에 critetira라는 이름으로 데이터가 있는지 확인, 있다면 해당 객체를 세션에서 꺼내서 매개 변수 criteria에 할당
- temp.replaceAll("\r\n", ""); : 에디터로 인해 생기는 공백을 제거 해주기위해 사용
- redirectAttrriutes.addAttribute() : 글 상세 조회, 삭제, 수정 후 다시 목록으로 돌아 갈 때 기존에 있던 페이지로 돌아 가기 위해 정보를 전달(검색 목록에서도 적용)
- redirectAttrriutes.addFlashAttribute : 성공 메시지를 보내 줌

```

1 *BoardController.java @@
101    // 글 상세 보기
102    @RequestMapping(value="/getBoard.do", method = RequestMethod.GET)
103    public String getBoard(@RequestParam("seq") int seq,
104                           @ModelAttribute("criteria") Criteria criteria, BoardVO vo, Model model) {
105        System.out.println("글 상세 조회 처리");
106        model.addAttribute("board", boardService.getBoard(seq));
107        return "article/getBoard";
108    }
109
110   // 검색 조건 목록 설정, @RequestMapping 보다 먼저 호출됨
111   @ModelAttribute("conditionMap")
112   public Map<String, String> searchConditionMap() {
113       Map<String, String> conditionMap = new HashMap<String, String>;
114       conditionMap.put("제목", "TITLE");
115       conditionMap.put("내용", "CONTENT");
116       conditionMap.put("작성자", "WRITER");
117       return conditionMap;
118   }
119
120   // 글 목록 화면 호출
121   @RequestMapping(value="/getBoardList.do", method = RequestMethod.GET)
122   public String getBoardList(BoardVO vo, Model model, HttpSession session,
123                             Criteria criteria) throws Exception {
124       System.out.println("글 목록 검색 처리");
125       String auth = (String)session.getAttribute("userName");
126
127       // Null Check
128       if(vo.getSearchCondition() == null) vo.setSearchCondition("TITLE");
129       if(vo.getSearchKeyword() == null) vo.setSearchKeyword("");
130       if(auth == null && auth.equals("")) {
131           return "redirect:login.do";
132       } else {
133           PageMaker pageMaker = new PageMaker();
134           pageMaker.setCriteria(criteria);
135           pageMaker.setTotalCount(boardService.countArticles(criteria));
136
137           model.addAttribute("boardList", boardService.listCriteria(criteria));
138           model.addAttribute("pageMaker", pageMaker);
139           System.out.println(pageMaker.makeQuery(criteria.getPage()));
140           return "/article/getBoardList";
141       }
142   }
143 }
```

- `@ModelAttribute("conditionMap")` : 검색 조건 목록에 키와 값을 보냄. `/getBoardList.do` 가 요청 되어도 먼저 실행되어서 데이터를 전송함
- `auth` : 세션에 저장 되어 있는 로그인 정보를 저장하고 값이 비어 있으면 로그인 상태가 아니므로 로그인 창으로 이동
- `pageMaker`에 페이지 설정 필요한 값들 설정

```
//첨부 파일 다운로드 시 호출  
@RequestMapping("/download.do")  
@ResponseBody  
public byte[] downProcess(HttpServletRequest response, HttpServletResponse request,  
    @RequestParam String originalFileName, @RequestParam int seq) throws Exception{  
  
    BoardVO vo = new BoardVO();  
    vo.setOriginalFileName(originalFileName);  
    vo.setSeq(seq);  
    byte[] bytes = boardService.fileDownload(vo, response, request);  
    return bytes;  
}
```

- 파일 다운로드 클릭 시 요청 하는 controller

02 본론 | 프로젝트 기술 분석 – BoardService

```
package com.springbook.biz.board;

import java.io.IOException;

public interface BoardService {

    //글 등록
    void insertBoard(BoardVO vo);

    //글 수정
    void updateBoard(BoardVO vo);

    //글 삭제
    void deleteBoard(int seq);

    //글 상세 조회
    BoardVO getBoard(int seq);

    //글 목록 조회
    List<BoardVO> getBoardList(BoardVO vo);

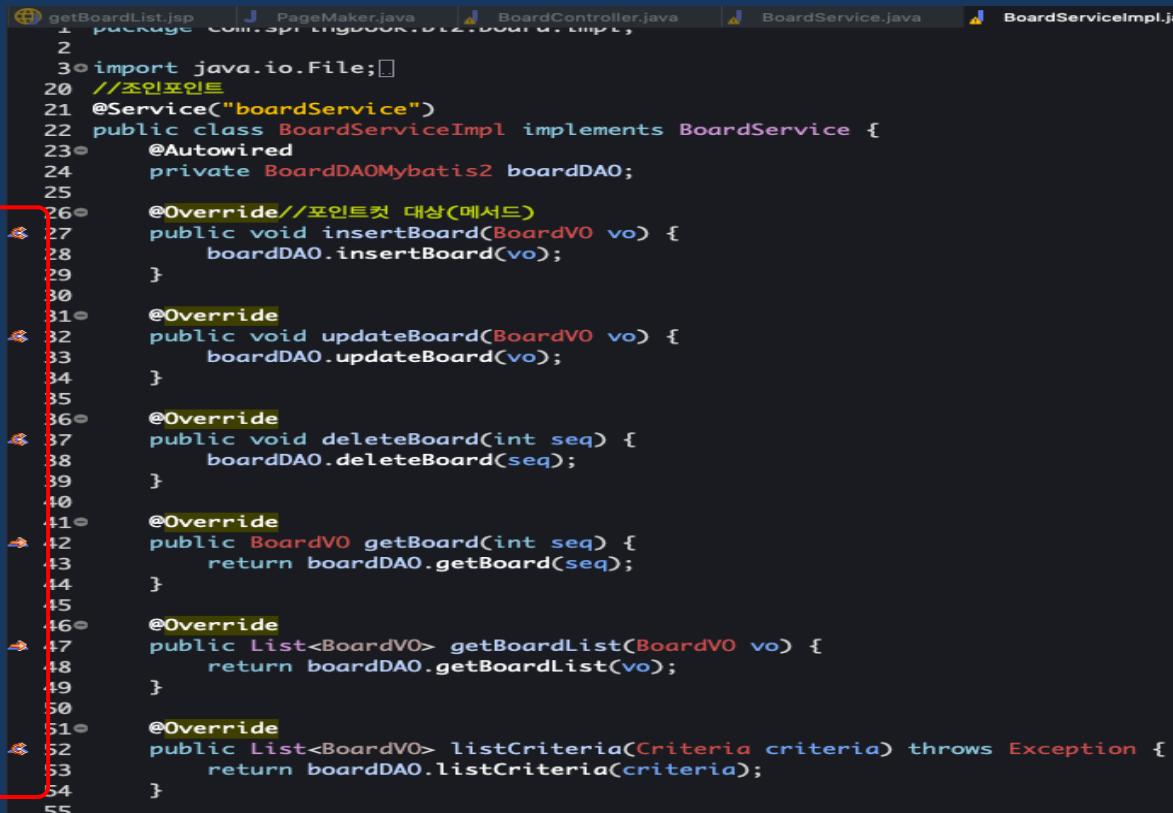
    List<BoardVO> listCriteria(Criteria criteria) throws Exception;

    int countArticles(Criteria criteria) throws Exception;

    Map<String, String> fileUpload(BoardVO vo, HttpServletRequest request);

    byte[] fileDownload(BoardVO vo, HttpServletResponse response, HttpServletRequest request) throws Exception;
}
```

- 비지니스로직이 있는 인터페이스



```
1 package com.springbook.biz.board.impl;
2
3 import java.io.File;
4
5 //조인포인트
6 @Service("boardService")
7 public class BoardServiceImpl implements BoardService {
8     @Autowired
9     private BoardDAOMybatis2 boardDAO;
10
11     @Override//포인트컷 대상(메서드)
12     public void insertBoard(BoardVO vo) {
13         boardDAO.insertBoard(vo);
14     }
15
16     @Override
17     public void updateBoard(BoardVO vo) {
18         boardDAO.updateBoard(vo);
19     }
20
21     @Override
22     public void deleteBoard(int seq) {
23         boardDAO.deleteBoard(seq);
24     }
25
26     @Override
27     public BoardVO getBoard(int seq) {
28         return boardDAO.getBoard(seq);
29     }
30
31     @Override
32     public List<BoardVO> getBoardList(BoardVO vo) {
33         return boardDAO.getBoardList(vo);
34     }
35
36     @Override
37     public List<BoardVO> listCriteria(Criteria criteria) throws Exception {
38         return boardDAO.listCriteria(criteria);
39     }
40
41     @Override
42     public void insertBoardWithAOP(BoardVO vo) {
43         boardDAO.insertBoard(vo);
44     }
45
46     @Override
47     public void updateBoardWithAOP(BoardVO vo) {
48         boardDAO.updateBoard(vo);
49     }
50
51     @Override
52     public void deleteBoardWithAOP(int seq) {
53         boardDAO.deleteBoard(seq);
54     }
55 }
```

- 비지니스로직이 있는 인터페이스
- 트랜잭션 처리를 위한 AOP 설정

| 프로젝트 기술 분석 – BoardServiceImpl(2)

```

@Override
public Map<String, String> fileUpload(BoardVO vo, HttpServletRequest request) {
    MultipartFile uploadFile = vo.getUploadFile();
    boolean isSuccess = false;
    String uploadPath = request.getSession().getServletContext().getRealPath("/upload/");

    File dir = new File(uploadPath);

    if (dir.isDirectory()) {
        dir.mkdir();
    }

    String originalFileName = uploadFile.getOriginalFilename();
    String saveFileName = originalFileName;
    Map<String, String> fileName = new HashMap<String, String>();

    if (saveFileName != null && !saveFileName.equals("")) {
        if (new File(uploadPath + saveFileName).exists()) {
            saveFileName = System.currentTimeMillis() + " " + saveFileName;
        }
        try {
            uploadFile.transferTo(new File(uploadPath + saveFileName));
            isSuccess = true;
            fileName.put("originalFileName", originalFileName);
            fileName.put("saveFileName", saveFileName);
        } catch (IllegalStateException e) {
            e.printStackTrace();
            isSuccess = false;
        } catch (IOException e) {
            e.printStackTrace();
            isSuccess = false;
        }
    }
    return fileName;
}

```

- 파일 업로드 가능

```
102* @Override  
103 public byte[] fileDownload(BoardVO vo, HttpServletResponse response, HttpServletRequest request) throws Exception {  
104     String uploadPath = request.getSession().getServletContext().getRealPath("/upload/");  
105     BoardVO saveFileName = selectFileName(vo);  
106     String save_fileName = saveFileName.getSaveFileName();  
107     File file = new File(uploadPath + save_fileName);  
108     byte[] bytes = FileCopyUtils.copyToByteArray(file);  
109     String fn = new String(vo.getOriginalFileName().getBytes("utf-8"), "iso_8859_1");  
110     response.setHeader("Content-Disposition", "attachment;filename=\"" + fn + "\"");  
111     response.setContentLength(bytes.length);  
112     return bytes;  
113 }  
114  
115* public BoardVO selectFileName(BoardVO vo) throws Exception {  
116     return boardDAO.selectSaveFileName(vo);  
117 }  
118 }
```

- 파일 다운로드 가능

```

package com.springbook.biz.board.impl;

import java.util.List;

@Repository
public class BoardDAOMybatis2 {

    @Autowired
    private SqlSessionTemplate mybatis;

    // crud 기능의 메소드 구현
    // 글 등록
    public void insertBoard(BoardVO vo) {
        System.out.println("==> Mybatis2 JDBC로 insertBoard() 기능 처리");
        mybatis.insert("BoardDAO.insertBoard", vo);
    }

    // 글 수정
    public void updateBoard(BoardVO vo) {
        System.out.println("==> Mybatis2 JDBC로 updateBoard() 기능 처리");
        mybatis.update("BoardDAO.updateBoard", vo);
    }

    // 글 삭제
    public void deleteBoard(int seq) {
        System.out.println("==> Mybatis2 JDBC로 deleteBoard() 기능 처리");
        mybatis.delete("BoardDAO.deleteBoard", seq);
    }
}

```

- persistence 계층 로직 : 주로 데이터의 생성/수정/삭제/선택(검색)과 같은 CRUD 연산을 수행
- @Repository : applicationContext2.xml에서 component-scan에서 읽어 스캔하기 위한 어노테이션(Annotation)
- Mybatis를 연동하기 위해 SqlSessionTemplate 를 DI(Dependency Injection) 하기 위한 어노테이션
- resources/mapping/user-mapping.xml 에 있는 SQL들과 매핑되는 메소드

```

// 글 삭제
public void deleteBoard(int seq) {
    System.out.println("====> Mybatis2 JDBC로 deleteBoard() 기능 처리");
    mybatis.delete("BoardDAO.deleteBoard", seq);
}

// 글 상세 조회
public BoardVO getBoard(int seq) {
    System.out.println("====> Mybatis2 JDBC로 getBoard() 기능 처리");
    mybatis.update("BoardDAO.updateHitCnt", seq);
    return (BoardVO) mybatis.selectOne("BoardDAO.getBoard", seq);
}

// 게시판 리스트 조회
public List<BoardVO> listCriteria(Criteria criteria) throws Exception {
    System.out.println("====> Mybatis2 JDBC 페이징 기능 처리");
    return mybatis.selectList("BoardDAO.listCriteria", criteria);
}

// 게시판 글 총 수
public int countArticles(Criteria criteria) throws Exception {
    return mybatis.selectOne("BoardDAO.countArticles", criteria);
}

// 업로드 한 파일 이름 조회
public BoardVO selectSaveFileName(BoardVO vo) throws Exception {
    return mybatis.selectOne("BoardDAO.getSaveFileName", vo);
}

```

- persistence 계층 로직 : 주로 데이터의 생성/수정/삭제/선택(검색)과 같은 CRUD 연산을 수행
- @Repository : applicationContext2.xml에서 component-scan에서 읽어 스캔하기 위한 어노테이션(Annotation)
- Mybatis를 연동하기 위해 SqlSessionTemplate 를 DI(Dependency Injection) 하기 위한 어노테이션
- resources/mapping/user-mapping.xml 에 있는 SQL들과 매핑되는 메소드

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
3 "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
4<mapper namespace="BoardDAO">
5<resultMap type="board" id="boardResult">
6    <id property="seq" column="SEQ"/>
7    <result property="title" column="TITLE"/>
8    <result property="writer" column="WRITER"/>
9    <result property="content" column="CONTENT"/>
10   <result property="regDate" column="REGDATE"/>
11   <result property="cnt" column="CNT"/>
12   <result property="originalFileName" column="ORIGINALFILENAME"/>
13   <result property="saveFileName" column="SAVEFILENAME"/>
14</resultMap>
15
16<insert id="insertBoard">
17    <![CDATA[
18      INSERT INTO BOARD(SEQ
19          , TITLE
20          , WRITER
21          , CONTENT
22          , ORIGINALFILENAME
23          , SAVEFILENAME)
24      VALUES((SELECT NVL(MAX(SEQ), 0) + 1 FROM BOARD)
25          , #{title}
26          , #{writer}
27          , #{content}
28          , #{originalFileName,jdbcType=VARCHAR}
29          , #{saveFileName,jdbcType=VARCHAR})
30    ]]>
31
32</insert>
33
34<update id="updateBoard">
35    <![CDATA[
36      UPDATE BOARD
37      SET TITLE=#{title}
38      , CONTENT=#{content}
39      WHERE SEQ=#{seq}
40    ]]>
41
42</update>

```

- boardResult : select SQL의 결과들을 매팅
- insertBoard: 게시판 입력 SQL
- updateBoard : 게시판 수정 SQL

```
43
44@    <update id="updateHitCnt">
45        <!CDATA[
46            UPDATE BOARD
47                SET CNT = NVL(cnt, 0) + 1
48                WHERE SEQ = #{seq}
49            ]]>
50    </update>
51
52@    <delete id="deleteBoard">
53        <!CDATA[
54            DELETE BOARD WHERE SEQ=#{seq}
55        ]]>
56    </delete>
57
58@    <select id="getBoard" resultType="board">
59        <!CDATA[
60            SELECT * FROM BOARD WHERE SEQ=#{seq}
61        ]]>
62    </select>
63
64@    <select id="getSaveFileName" resultType="board">
65        <!CDATA[
66            SELECT SAVEFILENAME
67            FROM BOARD
68            WHERE ORIGINALFILENAME=#{originalFileName}
69            AND SEQ=#{seq}
70        ]]>
71    </select>
```

- updateHiCnt : 조회수 증가 SQL
- deleteBoard: 게시 글 삭제 SQL
- getBoard : 게시글 상세 보기 SQL
- getSaveFileName : 업로드 한 파일 변경된 이름 조회 SQL

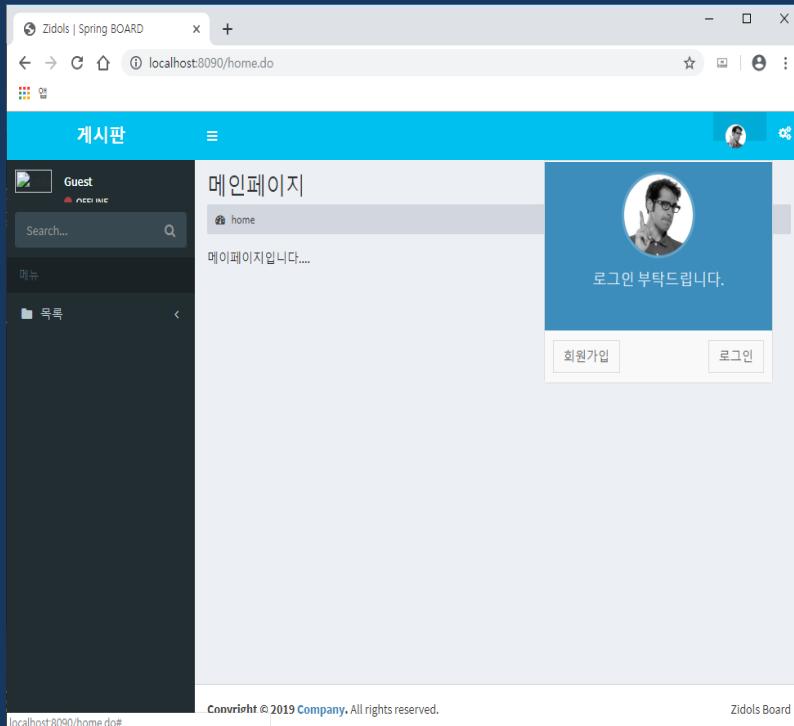
```
33
34@    <select id="listCriteria" resultMap="boardResult">
35        <![CDATA[
36            SELECT *
37                FROM (SELECT ROWNUM AS RNUM,
38                            BO./*
39                                FROM (SELECT * FROM BOARD
40                                    ]])>
41        <include refid="search"/>
42        <![CDATA[
43            ORDER BY SEQ DESC) BO)
44            WHERE RNUM BETWEEN #{pageStart} and #{perPageNum}
45        ]]>
46    </select>
47
48@    <sql id="search">
49@        <if test="searchCondition == 'TITLE'">
50            WHERE TITLE LIKE '%'||#{searchKeyword}||'%'
51        </if>
52@        <if test="searchCondition == 'CONTENT'">
53            WHERE CONTENT LIKE '%'||#{searchKeyword}||'%'
54        </if>
55@        <if test="searchCondition == 'WRITER'">
56            WHERE WRITER LIKE '%'||#{searchKeyword}||'%'
57        </if>
58    </sql>
```

- `liarCriteria` : 게시판 리스트의 페이지 처리를 위한 SQL
- `search` : 검색 조건에 따른 WHERE 절 선택

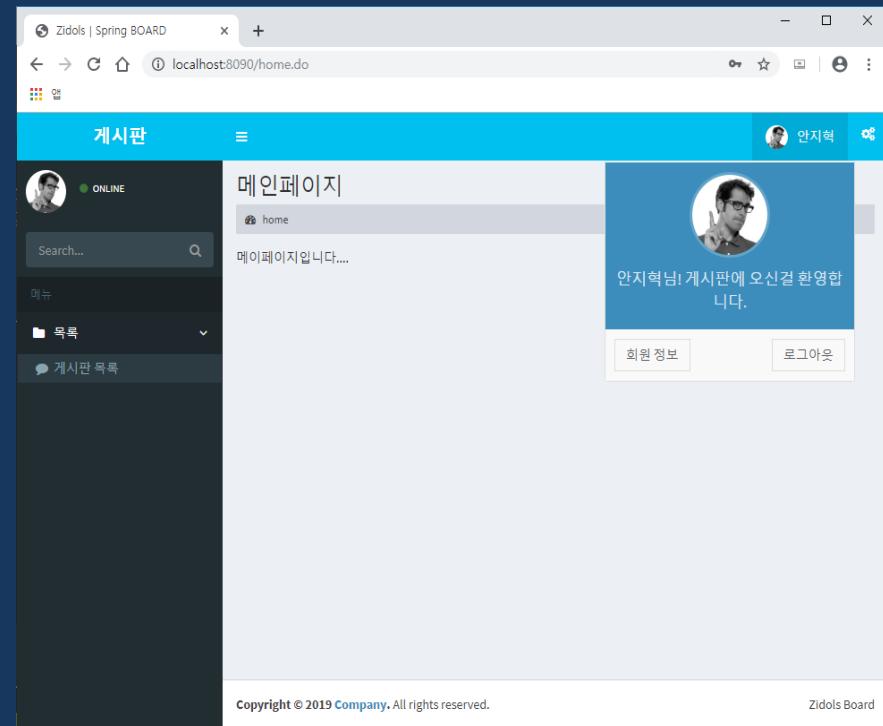
```
159
160      <select id="countArticles" resultType="int">
161          <![CDATA[
162              SELECT
163                  COUNT(seq)
164              FROM board
165              WHERE seq &gt; 0
166          ]]&gt;
167          &lt;include refid="search2"/&gt;
168      &lt;/select&gt;
169
170      &lt;sql id="search2"&gt;
171          &lt;if test="searchCondition == 'TITLE'"&gt;
172              AND TITLE LIKE '%'||#{searchKeyword}||'%'
173          &lt;/if&gt;
174          &lt;if test="searchCondition == 'CONTENT'"&gt;
175              AND CONTENT LIKE '%'||#{searchKeyword}||'%'
176          &lt;/if&gt;
177          &lt;if test="searchCondition == 'WRITER'"&gt;
178              AND WRITER LIKE '%'||#{searchKeyword}||'%'
179          &lt;/if&gt;
180      &lt;/sql&gt;</pre>
```

- countArticles : 페이지 처리를 위한 게시글 수 조회하는 SQL
- search2 : 검색 조건에 따른 WHERE 절 선택

02 본론 | 프로젝트 기술 분석 – JSP 게시판



로그인 전 홈 화면



로그인 후 홈 화면

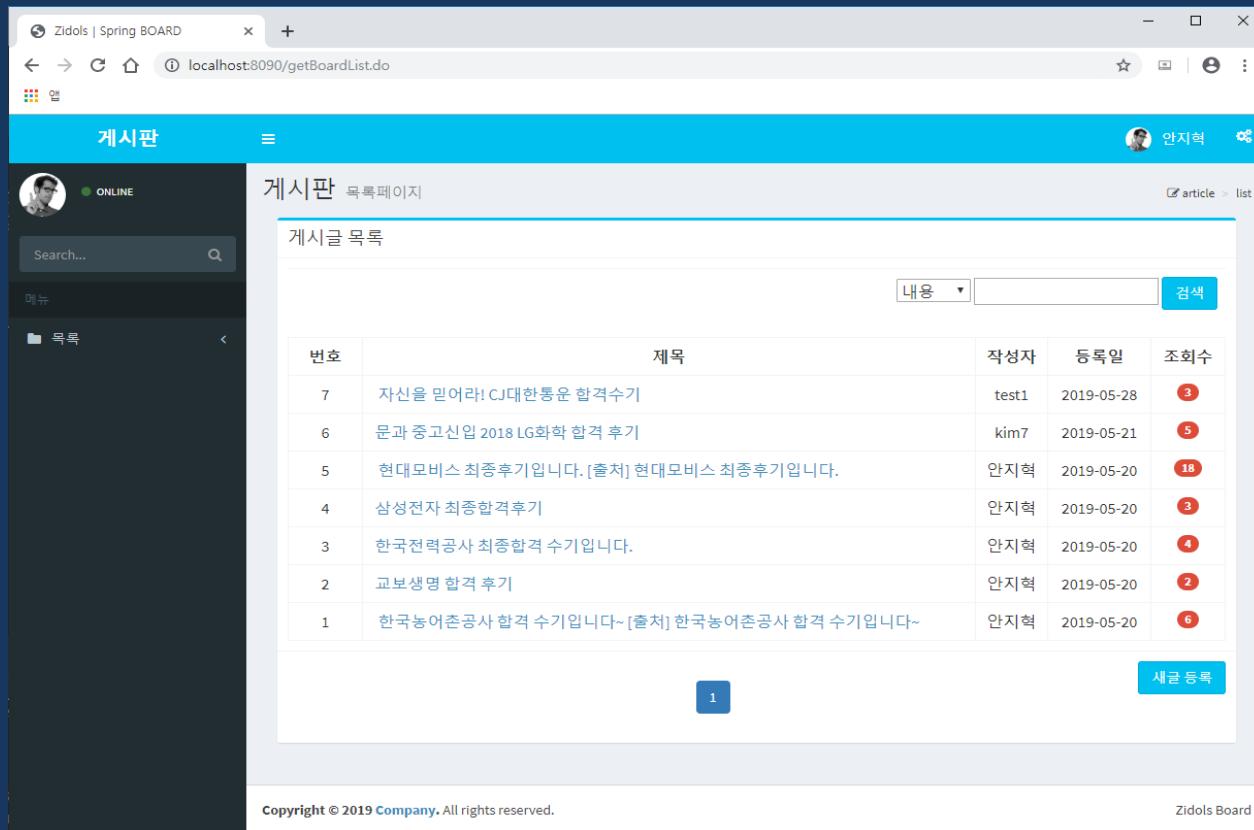
02 본론 | 프로젝트 기술 분석 – JSP 게시판

```
<p>
<c:choose>
    <c:when test="${not empty userId}">
        ${userName}<spring:message code="message.board.list.welcomMsg"/>
    </c:when>
    <c:otherwise>
        <spring:message code="message.board.list.loginPls"/>
    </c:otherwise>
</c:choose>

</p>
</li>
<li class="user-footer">
    <div class="pull-left">
        <c:choose>
            <c:when test="${not empty userId}">
                <a href="getuser.do?userId=${userId}" class="btn btn-default btn-flat"><spring:message code="message.board.list.userInfo"/></a>
            </c:when>
            <c:otherwise>
                <a href="joinPage.do" class="btn btn-default btn-flat"><spring:message code="message.board.list.join"/></a>
            </c:otherwise>
        </c:choose>
        </div>

        <div class="pull-right">
            <c:choose>
                <c:when test="${not empty userId}">
                    <a href="logout.do" class="btn btn-default btn-flat"><spring:message code="message.board.list.logout"/></a>
                </c:when>
                <c:otherwise>
                    <a href="login.do" class="btn btn-default btn-flat"><spring:message code="message.board.list.login"/></a>
                </c:otherwise>
            </c:choose>
        </div>
    </li>
</ul>
</li>
```

- 세션에 저장 한 회원 아이디로 화면 다르게 출력(jstl 사용)



- 로그인 후 게시판 목록 화면

- Page: 각 페이지 번호, perPageNum 페이지당 보여줄 개수
- 글 상세보기, 글 수정, 삭제 시 전달 됨

게시판 목록페이지

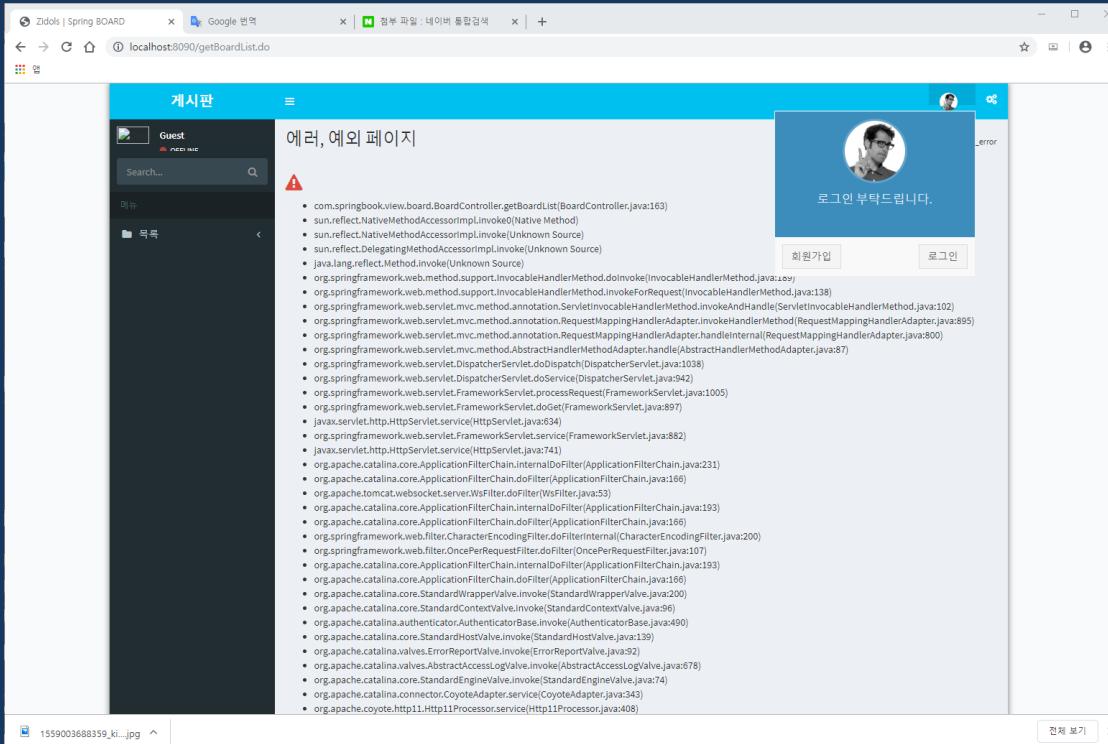
번호	제목	작성자	등록일	조회수
3012	글 등록 테스트1	지도리	2019-05-22	39
3011	글 등록 테스트	지들쓰쁨베	2019-05-21	80
3010	글 등록 테스트ㅋㅋㅋㅋ	안지돌	2019-05-21	23
3009	글 등록 테스트ㅋㅋㅋㅋ	안지돌	2019-05-21	2
3008	asdad	adsadad	2019-05-20	12
3007	1000번째 글 제목입니다...	user00	2019-05-17	8
3006	999번째 글 제목입니다...	user09	2019-05-17	1
3005	998번째 글 제목입니다...	user08	2019-05-17	3
3004	997번째 글 제목입니다...	user07	2019-05-17	3
3003	996번째 글 제목입니다...	user06	2019-05-17	3

내용 검색

1 2 3 4 5 6 7 8 9 10 다음 새글 등록

Copyright © 2019 Company. All rights reserved. Zidols Board

페이지 처리 화면



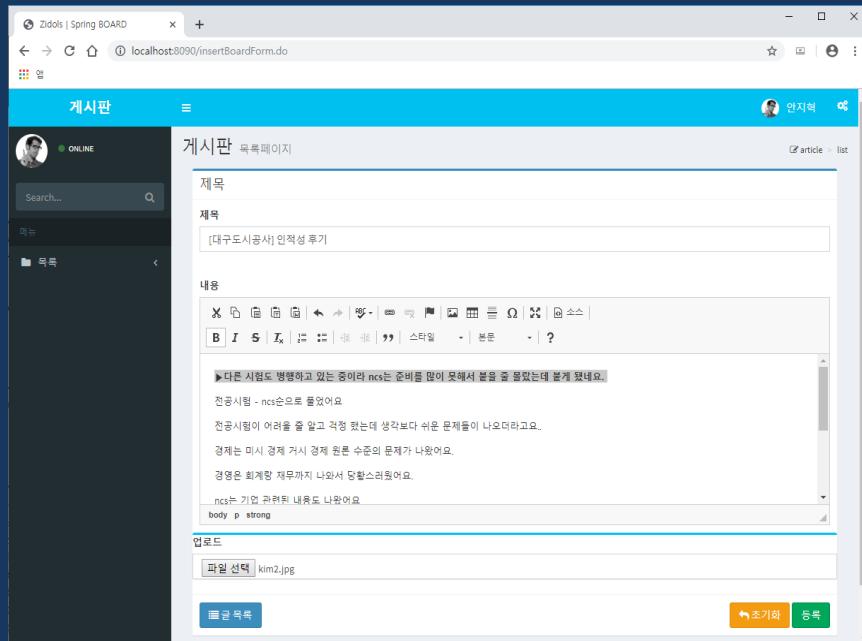
- 비 로그인 게시판 목록 화면

```

</div>
<div class="box-body">
    <form action="getBoardList.do" method="get">
        <table class="table">
            <tr>                                게시판 검색 창
                <td align="right">
                    <select name="searchCondition">
                        <c:forEach var="option" items="${conditionMap}">
                            <option value="${option.value }" ${option.key}>
                        </c:forEach>
                    </select>
                    <input name="searchKeyword" type="text">
                    <input type="submit" value="" class="btn btn-info">
                </td>
            </tr>
        </table>
    </form>
    <table class="table table-bordered">          게시판 목록
        <tr>
            <td align="center" width="8%" style="font-weight: bolder;"><spring:message code="message.board.list.table.head.seq"/></td>
            <td align="center" style="font-weight: bolder;"><spring:message code="message.board.list.table.head.title"/></td>
            <td align="center" style="font-weight: bolder;"><spring:message code="message.board.list.table.head.writer"/></td>
            <td align="center" style="font-weight: bolder;"><spring:message code="message.board.list.table.head.regDate"/></td>
            <td align="center" width="8%" style="font-weight: bolder;"><spring:message code="message.board.list.table.head.cnt"/></td>
        </tr>
        <c:forEach items="${boardList }" var="board">
            <tr>
                <td align="center">${board.seq }</td>
                <td align="left">&ampnbsp<a href="getBoard.do${pageMaker.makeQuery(pageMaker.criteria.page)}&seq=${board.seq }">${board.title }</a></td>
                <td align="center">${board.writer }</td>
                <td align="center"><fmt:formatDate value="${board.regDate }" pattern="yyyy-MM-dd"/></td>
                <td align="center"><span class="badge bg-red">${board.cnt}</span></td>
            </tr>
        </c:forEach>
    </table>

```

- 검색 창 키워드 : jstl로 검색 키워드 key값과 value를 받아와서 화면에 출력
- 게시판 목록 List형식으로 게시판 목록을 받아와 반복문으로 10개씩 화면에 출력



글 등록 화면

번호	제목	작성자	등록일	조회수
8	[대구도시공사] 인적성 투기	test1	2019-05-28	1
7	자신을 믿어라! CJ대한통운 학격수기	test1	2019-05-28	2
6	문과중고신입 2018 LG화학 학격 후기	kim7	2019-05-21	3
5	현대모비스 최종후기입니다. [출처] 현대모비스 최종후기입니다.	안지혁	2019-05-20	18
4	삼성전자 최종합격수기	안지혁	2019-05-20	3
3	한국전력공사 최종합격 수기입니다.	안지혁	2019-05-20	4
2	교보생명 학격 후기	안지혁	2019-05-20	2
1	한국농어촌공사 학격 수기입니다- [출처] 한국농어촌공사 학격 수기입니다-	안지혁	2019-05-20	6

글 등록 후 화면

```
<section class="content container">
    <div class="col-lg-12">
        <form role="form" id="writeForm" method="post" action="insertBoard.do" enctype="multipart/form-data">
            <input class="form-control" type="hidden" id="writer" name="writer" value="${userId}">
            <div class="box_box-primary">
                <div class="box-header with-border">
                    <h3 class="box-title"><spring:message code="message.board.insert.title"/></h3>
                </div>
                <div class="box-body">
                    <div class="form-group">
                        <label for="title"><spring:message code="message.board.insert.title"/></label>
                        <input class="form-control" id="title" name="title" placeholder="제목을 입력해주세요">
                    </div>
                </div>
                <!-- /.box-header -->
                <div class="box-body pad">
                    <label for="content"><spring:message code="message.board.insert.content"/></label>
                    <textarea id="editor1" name="content" rows="10" cols="80" placeholder="내용을 입력해주세요"></textarea>
                </div>
                <div class="box box-info">
                    <div class="form-group">
                        <label for="upload"><spring:message code="message.board.insert.upload"/></label>
                        <input class="form-control" type="file" name="uploadFile">
                    </div>
                </div>
                <div class="box-footer">
                    <button type="button" onclick="location.href='getBoardList.do'" class="btn btn-primary"><i class="fa fa-list"></i>
                    <spring:message code="message.board.insert.boardList"/></button>
                    <div class="pull-right">
                        <button type="reset" class="btn btn-warning"><i class="fa fa-reply"></i> 초기화</button>
                        <input type="submit" class="btn btn-success" value="<spring:message code="message.board.insert.insertBtn"/>">
                    </div>
                </div>
            </div>
        </form>
    </div>
```

```
<script>
    var result = "${msg}";
    if (result == "regSuccess") {
        alert("게시글 등록이 완료되었습니다.");
    } else if (result == "modSuccess") {
        alert("게시글 수정이 완료되었습니다.");
    } else if (result == "delSuccess") {
        alert("게시글 삭제가 완료되었습니다.");
    }
</script>
```

글 등록을 위해 insertBoard.do를 Post방식으로 호출, 파일 업로드 위해 enctype : "multipart/form-data" 추가
<script></script>: 게시글을 등록, 수정, 삭제 성공 후 controller에서 메시지를 전달 해 alert창을 띄움

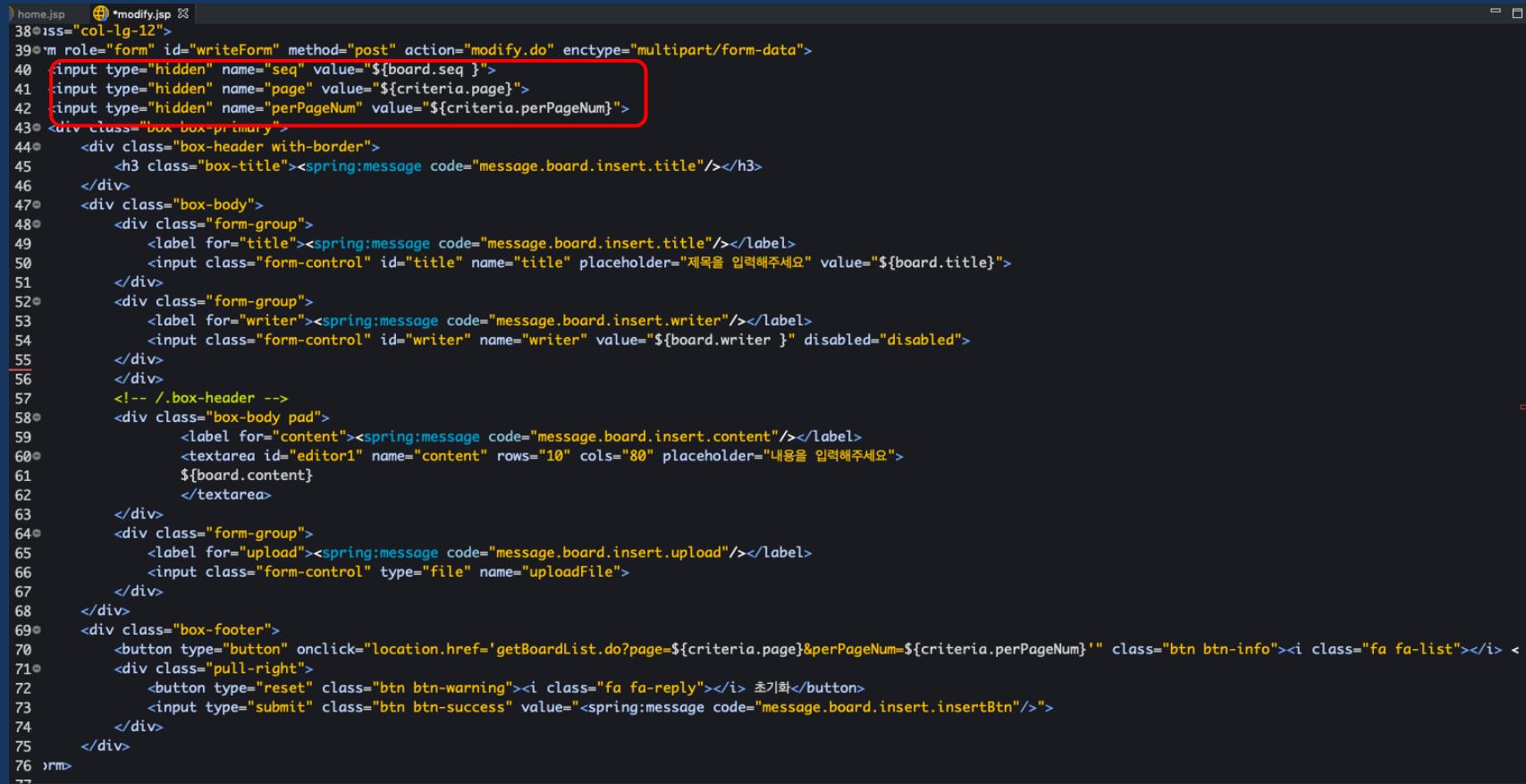
• 선택 한 글의 번호(seq)를 가지고 수정 화면에 내용을 출력하고 수정 함
• 수정 후 기존에 있던 페이지로 돌아가기 위해 page, perPageNum을 가지고 감

게시판 수정 화면

- 수정후 기존의 페이지로 돌아옴
- 3031번글의 제목 수정

번호	제목	작성자	등록일	조회수
3032	업로드 테스트88	test1	2019-05-27	0
3031	수정]글 등록 테스트	test1	2019-05-27	4
3030	업로드 테스트66	test1	2019-05-27	0
3029	업로드 테스트55	test1	2019-05-27	0
3028	업로드 테스트4	test1	2019-05-27	1
3027	업로드 테스트2	test1	2019-05-27	0
3026	파일업로드 테스트	test1	2019-05-27	4
3025	zzzzzz	test1	2019-05-25	1
3024	수정4]글 등록 테스트ㅋㅋㅋㅋ	test1	2019-05-25	9
3023	수정2]글 등록 테스트ㅋㅋㅋㅋ	test1	2019-05-25	4

수정 후 기존에 페이지로 돌아온 화면

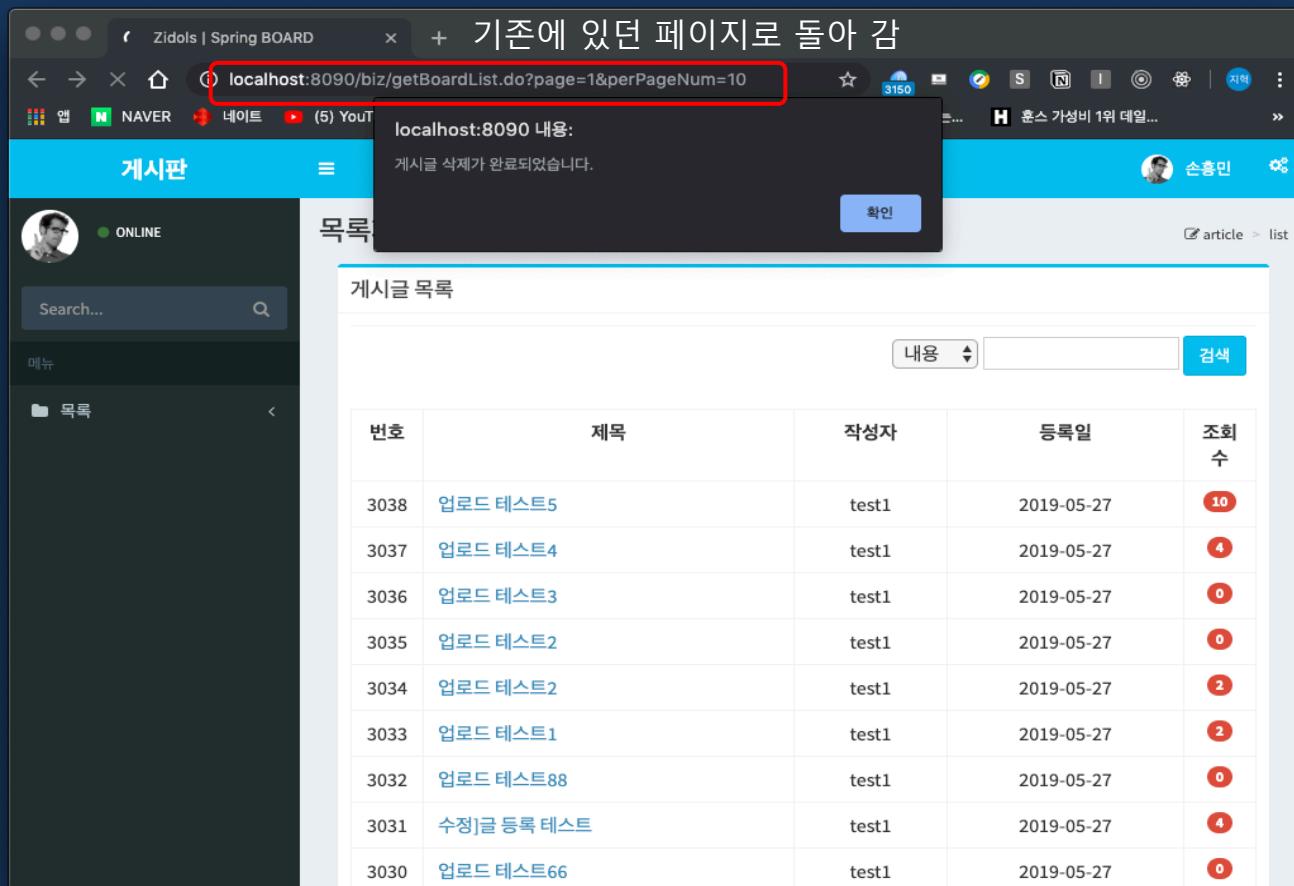


```
home.jsp ④ *modify.jsp ④
38@iss="col-lg-12">
39@  m role="form" id="writeForm" method="post" action="modify.do" enctype="multipart/form-data">
40  <input type="hidden" name="seq" value="${board.seq }">
41  <input type="hidden" name="page" value="${criteria.page}">
42  <input type="hidden" name="perPageNum" value="${criteria.perPageNum}">
43@ <div class="box box-primary">
44@   <div class="box-header with-border">
45    <h3 class="box-title"><spring:message code="message.board.insert.title"/></h3>
46  </div>
47@  <div class="box-body">
48@    <div class="form-group">
49      <label for="title"><spring:message code="message.board.insert.title"/></label>
50      <input class="form-control" id="title" name="title" placeholder="제목을 입력해주세요" value="${board.title}">
51    </div>
52@    <div class="form-group">
53      <label for="writer"><spring:message code="message.board.insert.writer"/></label>
54      <input class="form-control" id="writer" name="writer" value="${board.writer }" disabled="disabled">
55    </div>
56  </div>
57  <!-- /.box-header -->
58@  <div class="box-body pad">
59    <label for="content"><spring:message code="message.board.insert.content"/></label>
60    <textarea id="editor1" name="content" rows="10" cols="80" placeholder="내용을 입력해주세요">
61      ${board.content}
62    </textarea>
63  </div>
64@  <div class="form-group">
65    <label for="upload"><spring:message code="message.board.insert.upload"/></label>
66    <input class="form-control" type="file" name="uploadFile">
67  </div>
68</div>
69@ <div class="box-footer">
70    <button type="button" onclick="location.href='getBoardList.do?page=${criteria.page}&perPageNum=${criteria.perPageNum}'" class="btn btn-info"><i class="fa fa-list"></i> <
71@    <div class="pull-right">
72      <button type="reset" class="btn btn-warning"><i class="fa fa-reply"></i> 초기화</button>
73      <input type="submit" class="btn btn-success" value="<spring:message code="message.board.insert.insertBtn"/>">
74    </div>
75  </div>
76 >rm>
```

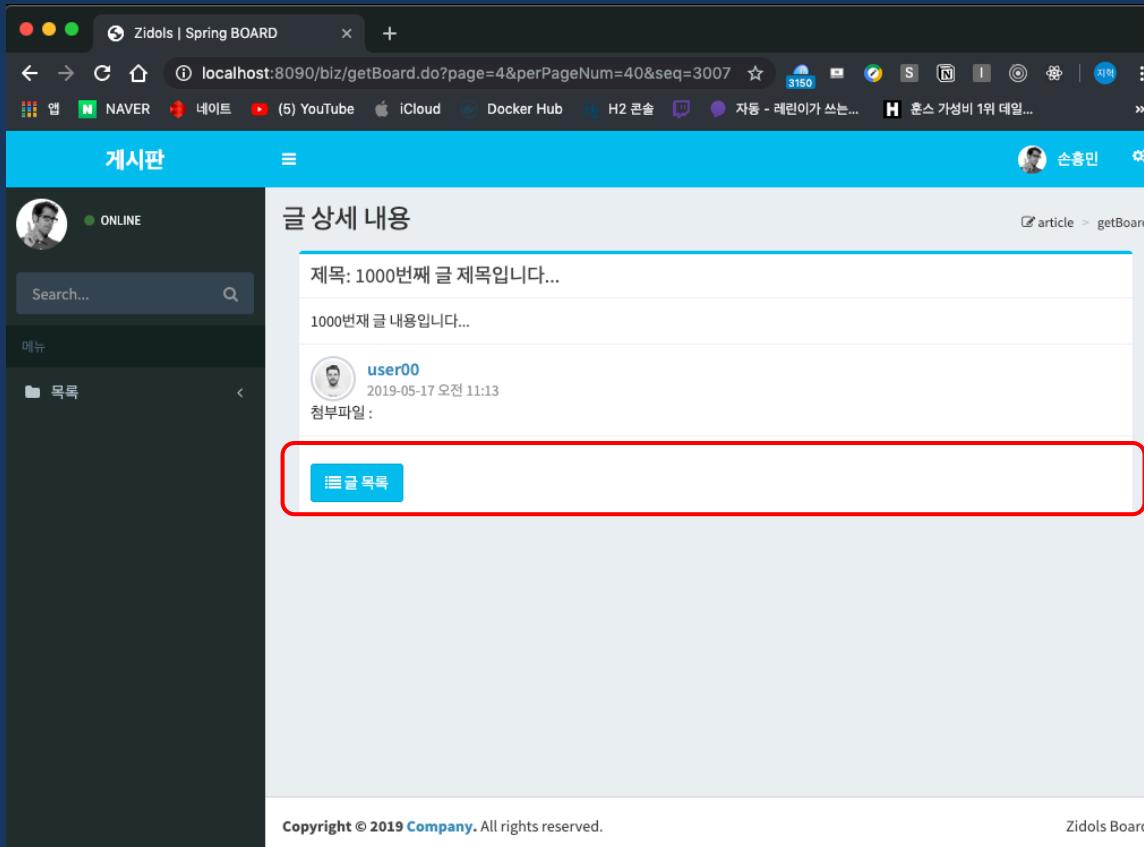
게시 글 수정 후 글번호, 페이지 번호, 페이지당 개수를 hidden 태입으로 보내 줌

번호	제목	작성자	등록일	조회수
3041	업로드 테스트7	test1	2019-05-28	10
3038	업로드 테스트5	test1	2019-05-27	10
3037	업로드 테스트4	test1	2019-05-27	4
3036	업로드 테스트3	test1	2019-05-27	0
3035	업로드 테스트2	test1	2019-05-27	0
3034	업로드 테스트2	test1	2019-05-27	2
3033	업로드 테스트1	test1	2019-05-27	2
3032	업로드 테스트88	test1	2019-05-27	0
3031	수정]글 등록 테스트	test1	2019-05-27	4
3030	업로드 테스트66	test1	2019-05-27	0

게시 글 삭제 전 화면



게시 글 삭제 전 화면



다른 회원 글

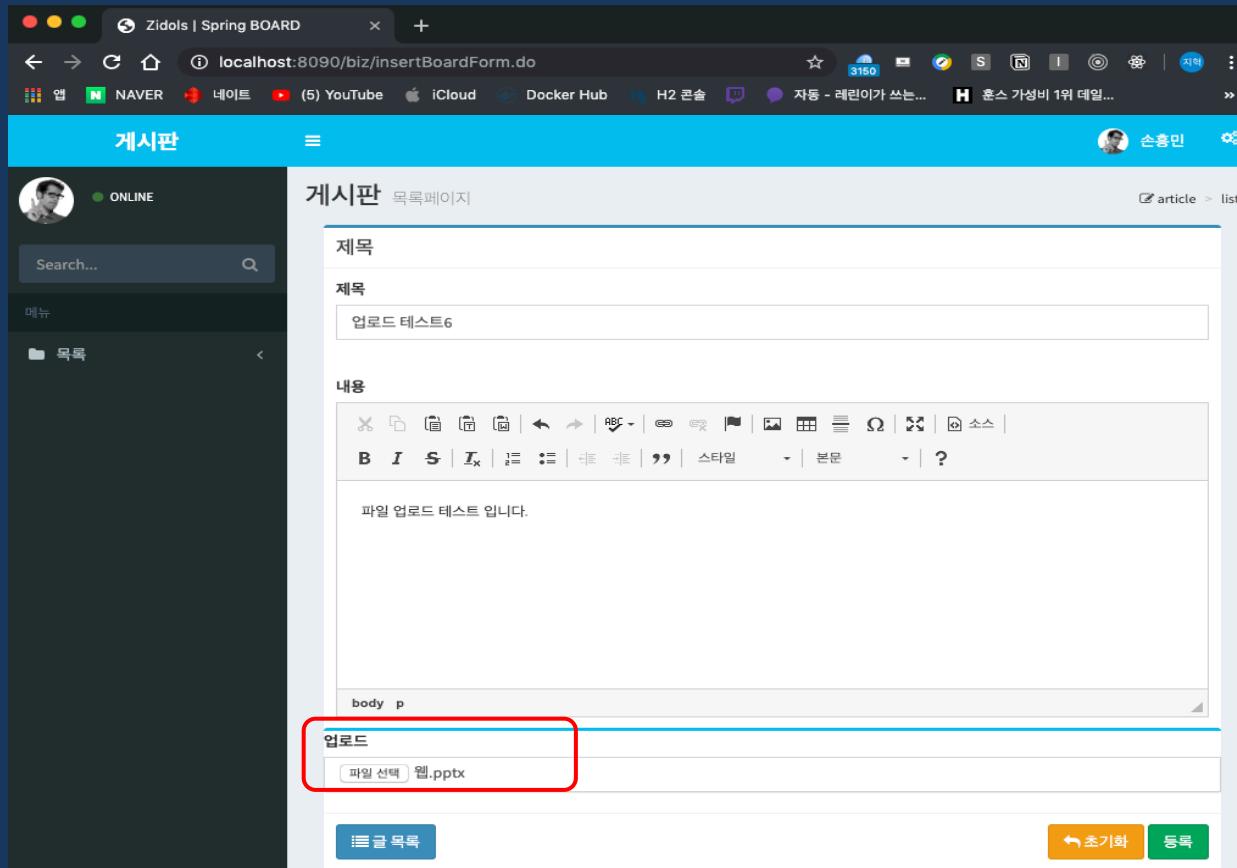
```

35@     <section class="content container-fluid">
36@         <div class="col-lg-12">
37@             <div class="box box-info">
38@                 <div class="box-header with-border">
39@                     <h3 class="box-title"><spring:message code="message.board.detail.title" />: ${board.title}</h3>
40@                 </div>
41@                 <div class="box-body" >
42@                     <div id="content">
43@                         </div>
44@                     </div>
45@                     <div class="box-footer">
46@                         <div class="user-block">
47@                             
48@                             <span class="username">
49@                                 <a href="#">${board.writer }</a>
50@                             </span>
51@                             <span class="description"><fmt:formatDate pattern="yyyy-MM-dd a HH:mm" value="${board.regDate}" /></span>
52@                             <span><spring:message code="message.board.detail.attachments" /> :
53@                                 <a href="download.do?originalFileName=${board.originalFileName }&seq=${board.seq}">${board.originalFileName }</a></span>
54@                         </div>
55@                     </div>
56@                     <div class="box-footer">
57@                         <form role="form" method="post">
58@                             <input type="hidden" name="seq" value="${board.seq }" >
59@                         </form>
60@                         <button type="button" class="btn btn-info listBtn" onclick="location.href='getBoardList.do?page=${criteria.page}&perPageNum=${criteria.perPageNum}'">
61@                             <i class="fa fa-list"></i> <spring:message code="message.board.detail.link.boardList" /></button>
62@                         <div class="pull-right">
63@                             <c:if test="${userId eq board.writer}">
64@                                 <button type="button" onclick="location.href='modifyForm.do?page=${criteria.page}&perPageNum=${criteria.perPageNum}&seq=${board.seq }'">
65@                                     <i class="fa fa-edit"></i> <spring:message code="message.board.detail.updateBtn"/></button>
66@                                 <button type="button" onclick="location.href='deleteBoard.do?page=${criteria.page}&perPageNum=${criteria.perPageNum}&seq=${board.seq }'">
67@                                     <i class="fa fa-trash"></i> <spring:message code="message.board.detail.link.deleteBoard" /></button>
68@                             </c:if>
69@                         </div>
70@                     </div>
71@                 </div>
72@             </div>
73@         </section>
74@         <!-- /.content -->

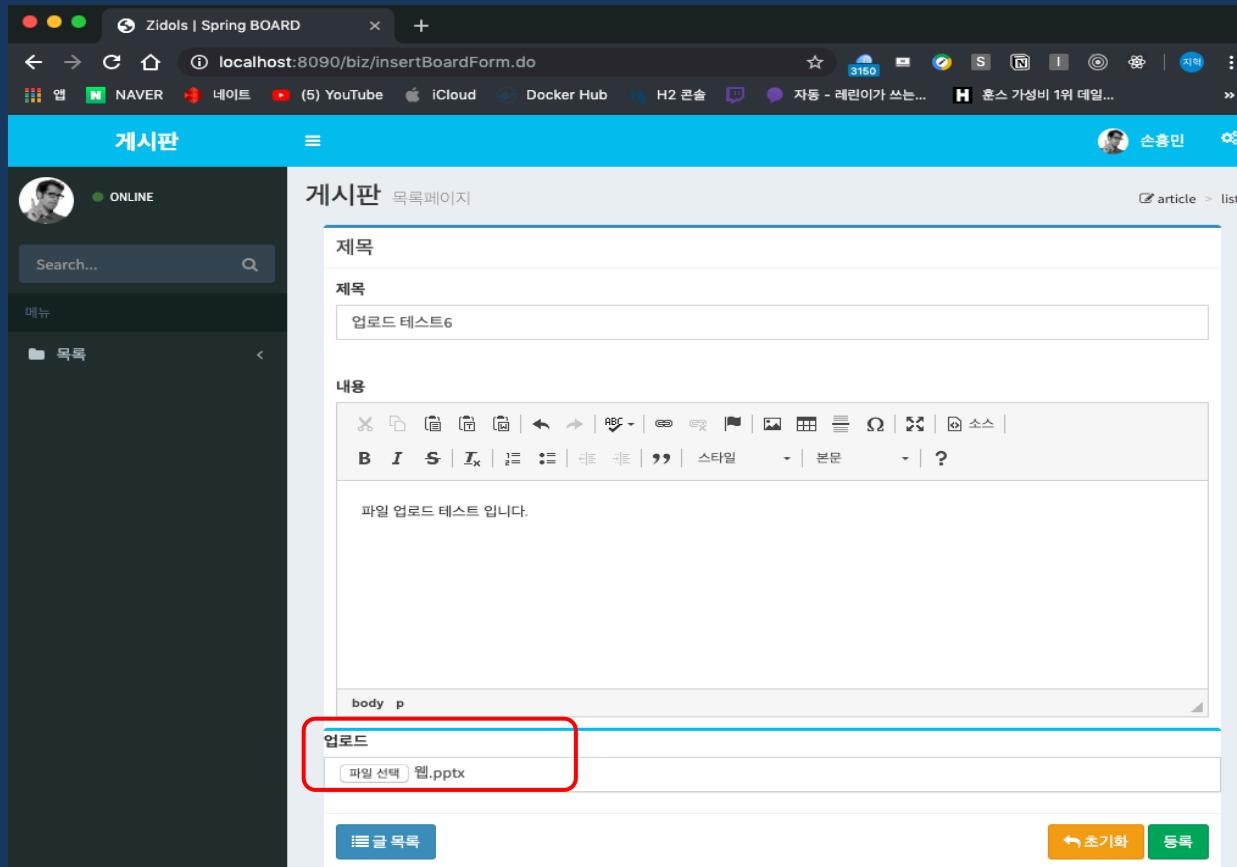
```

세션에 들어있는 로그인 id와 작성자 id를 비교해서 보여줌

게시글 상세화면 jsp



다른 회원 글



다른 회원 글

The screenshot shows a web browser window titled "Zidols | Spring BOARD" displaying a blog post. The post title is "제목: 업로드 테스트7" and the content is "파일 중복 테스트입니다.". Below the content, there is a red box highlighting a comment section. A comment by "test1" from May 29, 2019, at 8:44 AM is shown, with the file attachment "첨부파일 : 헵.jpg".

The screenshot shows a second web browser window titled "Zidols | Spring BOARD" displaying a blog post. The post title is "제목: 업로드 테스트8" and the content is "동일한 파일 업로드 테스트입니다.". Below the content, there is a red box highlighting a comment section. A comment by "test1" from May 29, 2019, at 8:49 AM is shown, with the file attachment "첨부파일 : 헵.jpg".

동일한 이름의 파일을 저장

The screenshot shows a file browser window titled "upload". The left sidebar lists "즐겨찾기", "AirDrop", "최근 사용", "응용 프로그램", "데스크탑", and "드라이브". The main pane displays a hierarchical file structure:

- xml
 - conf
 - logs
 - publish.txt
 - temp
 - webapps
 - work
- BoardWebMyBatis
 - ROOT
- META-INF
- nouse
- resources
- sql
- uml
- upload
- WEB-INF

On the right, there is a preview area showing several image files: 20140720_211933.jpg, 155911279..._211933.jpg, 1559119772982_웹.jpg, 웹.jpg, and 웹.pptx.

Below the file browser is a table showing uploaded files:

SEQ	TITLE	WRITER	CONTENT	REGDATE	CNT	ORIGINALFILENAME	SAVEFILENAME
3041	업로드 테...	test1	<p>동일한 파일 업로드 테스트입니다.</p>	2019-05-29	1	웹.jpg	1559119772982_웹.jpg
3040	업로드 테...	test1	<p>파일 중복 테스트입니다.</p>	2019-05-29	4	웹.jpg	웹.jpg

중복 되는 파일 이름 고유번호 생성해서 중복을 피함

- 게시판 댓글 기능 추가
- AOP 기능의 효율 적인 활용 미약
- ORM을 Mybatis에서 Hibernate로 교체
- Spring Boot를 활용해서 개발
- Restful API로 변경 후 react, vue.js 와 같은 frontend와 적용

- CRUD 기본 개념을 이해
- AOP, DI의 개념 이해
- 프레임 워크의 많은 장점 습득
- Spring의 3-layer 구조를 이해
- Spring에서 제공 하는 클래스들이 매우 유용함
- Mybatis를 이용해서 DB를 연동하는 것의 편리함

감사합니다.

안지혁

Spring과 Mybatis를 활용한 회원게시판

74