

CHALKAG

카페형 카메라 커뮤니티

INFINITYSTONE

목차

- 팀원 소개 및 역할 분담
- 기대 효과
- 프로젝트 목적
- 프로젝트 주요 기능
- 개발 환경
- 프로젝트 설계
- 시연 및 기능 설명
- 추후 개발 방향



팀원 소개 및 역할 분담



팀원1

팀장
Controller
아이디 찾기
비밀번호 찾기
에러페이지
문자 API
SNS 로그인 API



팀원2

팀원
Model
게시글 작성
게시글 수정
CKEditor
이미지 업로드
SNS 로그인 API



팀원3

팀원
VIEW
댓글 CRUD
비동기 처리
모달창



팀원4

팀원
VIEW
회원가입
로그인
로그아웃
문자 API
이미지 업로드



팀원5

팀원
Controller
게시글 전체 출력
게시글 세부 출력
필터 검색



정석진

팀원
Model
마이페이지
페이징처리
필터검색
이미지 업로드

프로젝트 기대 효과

- MVC 2 패턴에 대한 이해 확립
- API에 대한 이해와 활용 능력 향상
- 각 기능 개발 과정에서의 설계 수정을 회의록으로 문서화를 통한 협업 능력 향상
- 설계 수정을 통한 의사소통 능력 향상
- 코드 취합을 통한 코드 리뷰 능력 향상
- 코드 취합을 위한 주석 작성의 습관화
- 오류 탐색 및 오류 발생 시 해결 능력 향상

프로젝트 목적

- 카메라 전문가와 입문자들이 자유롭게 의견을 교류하고 정보를 공유할 수 있는 카페형 커뮤니티를 제공
- MVC 2 패턴에 대한 이해 확립
- API의 이해와 활용

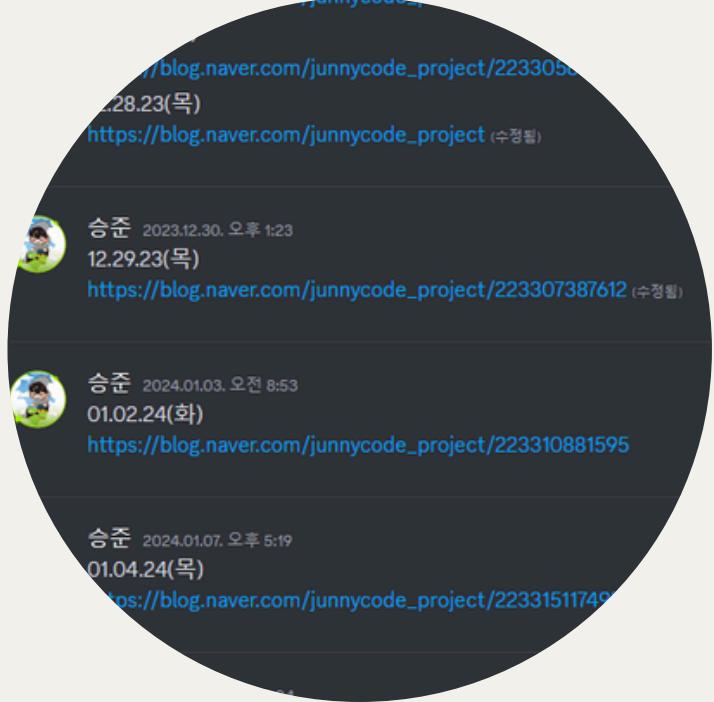
프로젝트 주요 기능

- 회원 가입 및 로그인
- 카테고리 별 글 작성, 수정, 삭제
- 댓글 작성, 수정
- 관리자 메뉴

개발 환경

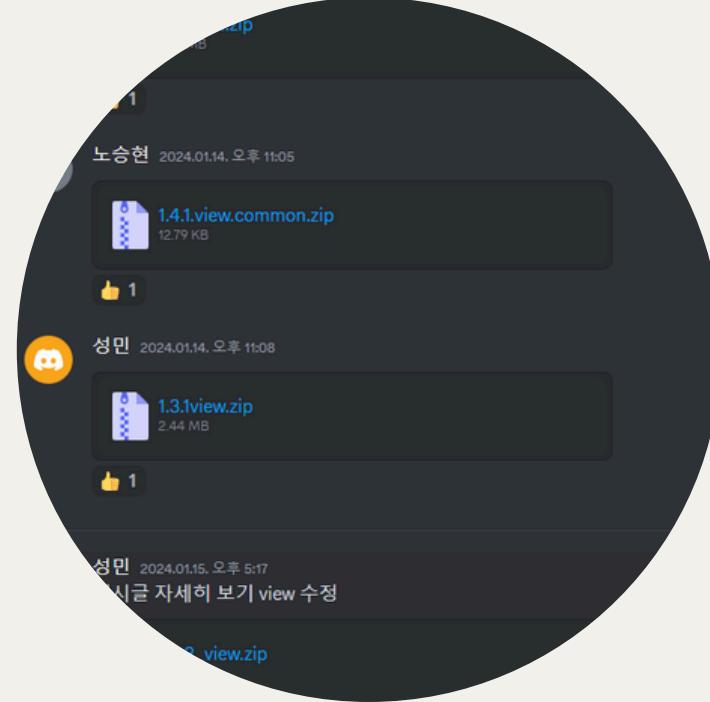


협업 관리



회의록 관리

회의록 작성 후 디스코드로 공유



버전 관리

0.0.0.1 버전으로 시작
각 팀원들에게 고유번호 부여
개인 버전 관리

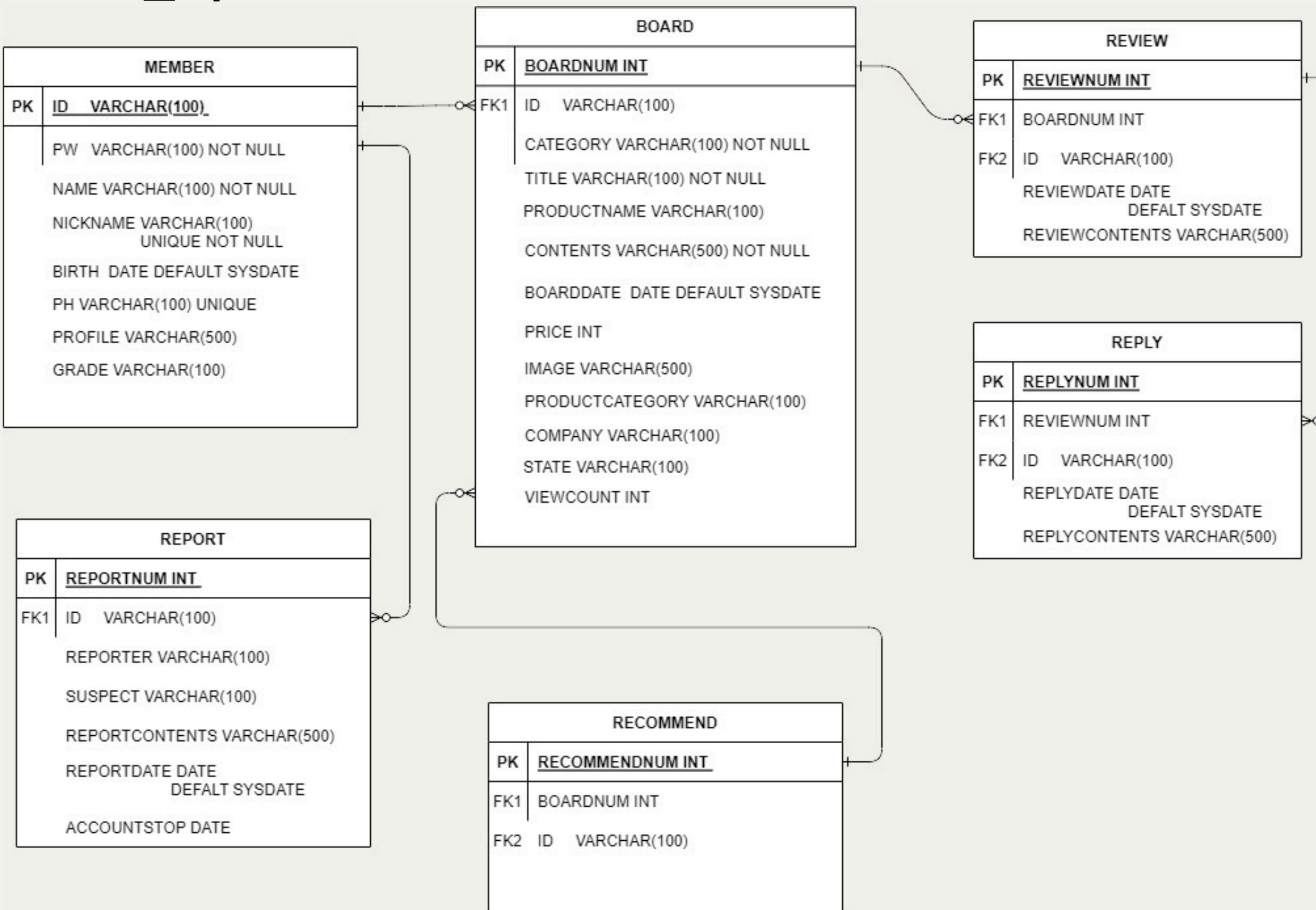


버전 관리

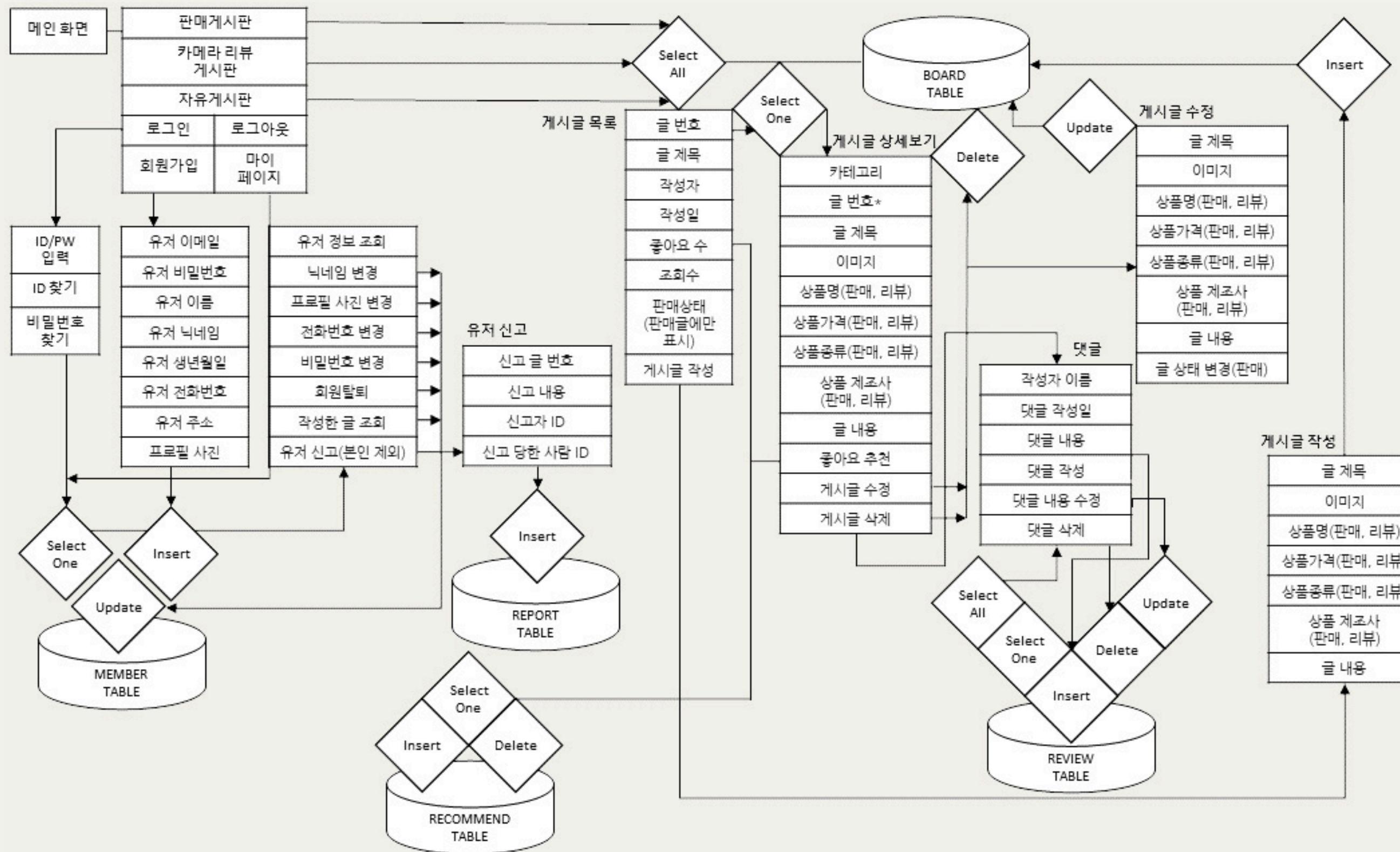
기능 구현 완료 시 취합
이후 버전 증가

프로젝트 개발 일정

TABLE 설계



LOGIC PROCESS 설계



데이터 설계

9.0.1ver 기준		
	jsp	Data
board	cameraReviewSelectAllPage 카메라 리뷰글 목록보기 화면	boardNum, title, nickname, boardDate, recommendCNT, viewCount 보드넘버, 제목, 작성자, 작성일, 좋아요 수, 조회수
	cameraReviewSelectOnePage 카메라 리뷰글 상세보기 화면	title, nickname, boardDate, recommendCNT, viewCount, contents, reviewDate, reviewContents, image 제목, 작성자, 작성일, 좋아요 수, 조회수, 내용, 댓글 작성일, 댓글 내용, 이미지
	cameraReviewUpdatePage 카메라 리뷰글 내용 변경 화면	title, price, productCategory, company, contents, image 제목, 가격, 종류, 제조사, 내용, 이미지
	cameraReviewWritePage 카메라 리뷰글 작성 화면	title, price, productCategory, company, contents, image 제목, 가격, 종류, 제조사, 내용, 이미지
	freeBoardSelectAllPage 자유게시판 목록보기 화면	boardNum, title, nickname, boardDate, recommendCNT, viewCount 보드넘버, 제목, 작성자, 작성일, 좋아요 수, 조회수
	freeBoardSelectOnePage 자유게시판 상세보기 화면	title, nickname, boardDate, recommendCNT, viewCount, contents, reviewDate, reviewContents, image 제목, 작성자, 작성일, 좋아요 수, 조회수, 내용, 댓글 작성일, 댓글 내용, 이미지
	freeBoardUpdatePage 자유게시판 내용변경 화면	title, contents, image 제목, 내용, 이미지
	freeBoardWritePage 자유게시판 글 작성 화면	title, contents, image 제목, 내용, 이미지
	myBoardSelectAllPage 내가 작성한 글 목록보기 화면	boardNum, title, nickname, boardDate, recommendCNT, viewCount 보드넘버, 제목, 작성자, 작성일, 좋아요 수, 조회수
	myBoardSelectOnePage 내가 작성한 글 상세보기 화면	title, nickname, boardDate, recommendCNT, viewCount, contents, reviewDate, reviewContents, image 제목, 작성자, 작성일, 좋아요 수, 조회수, 내용, 댓글 작성일, 댓글 내용, 이미지
	sellBoardSelectAllPage 카메라 판매글 카테고리보기 화면	boardNum, title, nickname, boardDate, recommendCNT, viewCount 보드넘버, 제목, 작성자, 작성일, 좋아요 수, 조회수
	sellBoardSelectOnePage 카메라 판매글 상세보기 화면	title, nickname, boardDate, recommendCNT, viewCount, contents, reviewDate, reviewContents, image 제목, 작성자, 작성일, 좋아요 수, 조회수, 내용, 댓글 작성일, 댓글 내용, 이미지
	sellBoardUpdatePage 카메라 판매글 수정 화면	title, price, productCategory, company, contents, image 제목, 가격, 종류, 제조사, 내용, 이미지
	sellBoardWritePage 카메라 판매글 글 작성 화면	title, price, productCategory, company, contents, image 제목, 가격, 종류, 제조사, 내용, 이미지
common		
	changePwPage 비밀번호 변경 화면	newPw, newPwCheck 변경할 비밀번호, 변경할 비밀번호 확인
	checkPwPage 비밀번호 확인 화면	checkPw, checkPwCheck 현재비밀번호, 현재비밀번호 확인
	findIdPage 아이디 찾기 화면	findName, phCheck 이름, 전화번호
	findIdResultPage 아이디 찾기 결과 화면	id 아이디
	findPwPage 비밀번호 찾기 화면	findName, findId 이름, 아이디
	findPwResultPage 비밀번호 찾기 결과 화면	pw 비밀번호
	joinPage 회원가입 화면	email, pw, pwCheck, memberName, nickName, img, ph 아이디, 비밀번호, 비밀번호확인, 이름, 닉네임, 프로필이미지, 핸드폰번호
	loginPage 로그인 화면	id, pw 아이디, 비밀번호, 비밀번호확인, 이름, 닉네임, 프로필이미지, 핸드폰번호
	mainPage 메인 화면	X
	memberPage 유저 정보 화면	img, myPageName, userPageNickname, myPageID 프로필이미지, 이름, 닉네임, 아이디
	myPage 내 정보 화면	img, myPageID, myPageName, myPageNickname, myPagePh 프로필이미지, 아이디, 이름, 닉네임, 번호

액션 설계

FrontController			
No.	action.equals()	Action Class	기능
1	main.do	MainAction	메인페이지로 이동
2	joinPage.do	JoinPageAction	회원가입 페이지로 이동
3	join.do	JoinAction	회원가입
4	loginPage.do	LoginPageAction	로그인 페이지로 이동
5	logout.do	LogoutAction	로그아웃
6	findIdPage.do	FindIdPageAction	아이디 찾기 페이지로 이동
7	findId.do	FindIdAction	아이디 찾기
8	findIdResultPage.do	FindIdResultAction	아이디 찾기 결과 페이지로 이동
9	findPwPage.do	FindPwPageAction	비밀번호 찾기 페이지로 이동
10	findPw.do	FindPwAction	비밀번호 찾기
11	changePwPage.do	ChangePwPageAction	비밀번호 변경 페이지로 이동
12	changePw.do	ChangePwAction	비밀번호 변경
13	checkPwPage.do	CheckPwPageAction	비밀번호 확인 페이지로 이동
14	checkPw.do	CheckPwAction	비밀번호 확인
15	myPage.do	MyPagePageAction	마이페이지로 이동
16	memberPage.do	MemberPageAction	유저 페이지로 이동
17	changeNickname.do	ChangeNicknameAction	닉네임 변경
18	changePh.do	ChangePhAction	전화번호 변경
19	myBoardSelectAllPage.do	MyboardSelectAllPageAction	내가 작성한 글 목록보기 페이지로 이동
20	myBoardSelectOnePage.do	MyBoardSelectOnePageAction	내가 작성한 글 상세보기 페이지로 이동
21	deleteAccount.do	DeleteAccountAction	회원 탈퇴
22	memberBoardSelectAllPage.do	MemberBoardSelectAllPageAction	유저가 작성한 글 목록보기 페이지로 이동
23	memberBoardSelectOnePage.do	MemberBoardSelectOnePageAction	유저가 작성한 글 상세보기 페이지로 이동
24	sellBoardSelectAllPage.do	SellBoardSelectAllPageAction	카메라 판매글 카테고리 페이지로 이동
25	sellBoardSelectOnePage.do	SellBoardSelectOnePageAction	카메라 판매글 상세보기 페이지로 이동
26	sellBoardWritePage.do	sellBoardWritePageAction	카메라 판매글 작성 페이지로 이동
27	sellBoardWrite.do	sellBoardWriteAction	카메라 판매글 작성
28	sellBoardUpdatePage.do	SellBoardUpdatePageAction	카메라 판매글 수정페이지로 이동
29	sellBoardUpdate.do	SellBoardUpdateAction	카메라 판매글 수정
30	sellBoardDelete.do	SellBoardDeleteAction	카메라 판매글 삭제
31	sellBoardReviewPage.do	SellBoardReviewPageAction	카메라 판매글 댓글 상세보기 페이지로 이동
32	cameraReviewSelectAllPage.do	CameraReviewSelectAllPageAction	카메라 리뷰글 카테고리 페이지로 이동
33	cameraReviewSelectOnePage.do	CameraReviewSelectOnePageAction	카메라 리뷰글 상세보기 페이지로 이동
34	cameraReviewWritePage.do	CameraReviewWritePageAction	카메라 리뷰글 작성 페이지로 이동
35	cameraReivewWrite.do	CameraReviewWriteAction	카메라 리뷰글 작성
36	cameraReivewUpdatePage.do	CameraReviewUpdatePageAction	카메라 리뷰글 수정 페이지로 이동
37	cameraReviewUpdate.do	CameraReviewUpdateAction	카메라 리뷰글 수정
38	cameraReivewDelete.do	CameraReviewReviewPageAction	카메라 리뷰글 삭제
39	freeBoardSelectAllPage.do	FreeBoardSelectAllPageAction	자유게시판 카테고리 페이지로 이동
40	freeBoardSelectOnePage.do	FreeBoardSelectOnePageAction	자유게시판 상세보기 페이지로 이동
41	freeBoardWritePage.do	FreeBoardWritePageAction	자유게시판 작성 페이지로 이동
42	freeBoardWrite.do	freeBoardWriteAction	자유게시판 작성
43	freeBoardUpdatePage.do	FreeBoardUpdatePageAction	자유게시판 수정 페이지 이동
44	freeBoardUpdate.do	FreeBoardUpdateAction	자유게시판 수정
45	freeBoardDelete.do	freeBoardDeleteAction	자유게시판 글 삭제
46	reviewWrite.do	ReviewWriteAction	댓글 입력
47	reviewUpdate.do	ReviewUpdateAction	댓글 수정
48	reviewDelete.do	ReviewDeleteAction	댓글 삭제
49	reportWritePage.do	ReportWritePageAction	신고 작성 페이지로 이동
50	reportWrite.do	ReportWriteAction	신고 작성
51	reportManagePage.do	ReportSelectAllPageAction	신고 관리 페이지로 이동
ASYNC			
1	changeNickname.do	ChangeNicknameAction	닉네임 변경하기
2	checkID.do	CheckIDAction	이메일 중복확인
3	checkNickname.do	CheckNicknameAction	닉네임 중복확인
4	checkPh.do	CheckPhAction	전화번호 중복 확인
5	kakaoJoin.do	KakaoJoinAction	카카오 계정 회원가입
6	kakaoLogin.do	KakaoLoginAction	카카오 로그인
7	loginCheck.do	LoginCheckAction	로그인 (비동기)
8	naverJoin.do	NaverJoinAction	네이버 회원가입
9	naverLogin.do	NaverLoginAction	네이버 로그인
10	profileUpload.do	ProfileUploadAction	프로필 이미지 업로드
11	recommendUpAndDown.do	RecommendUpAndDownAction	좋아요 등록 및 삭제
ERROR			
1	errorPage.do	ErrorPageAction	에러페이지로 이동

프로젝트 시연

회원가입 및 로그인 구현

- 회원가입
- 문자 API
- 로그인
- 에러 해결

회원가입 (1/4)

MemberDAO.java

```
private static final String INSERT_JOIN = "INSERT INTO MEMBER(ID,PW,NAME,NICKNAME,BIRTH,PH,PROFILE,GRADE)"  
+ "VALUES(?,?,?,?,?,TO_DATE(?,'yyyy-MM-dd'),?,COALESCE(?,'default.jpg'),'신입')";
```

가독성을 위해 기능별로 서브쿼리를 작성

```
try {  
    pstmt = conn.prepareStatement(INSERT_JOIN);  
    pstmt.setString(1, memberDTO.getId());  
    pstmt.setString(2, memberDTO.getPw());  
    pstmt.setString(3, memberDTO.getName());  
    pstmt.setString(4, memberDTO.getNickname());  
    pstmt.setString(5, memberDTO.getBirth());  
    pstmt.setString(6, memberDTO.getPh());  
    // 만약 memberDTO.getProfile()이 null 이거나 비어 있다면 기본 값 'default_profile'을 사용하여 프로필  
    // 매개변수 설정  
    pstmt.setString(7,memberDTO.getProfile());  
}
```

```
// .metadata 앞까지 문자열잘라서 이미지가 저장되는 폴더인 memberProfileImages까지의 상경로 부여  
uploadDir = uploadDir.substring(1, uploadDir.indexOf(".metadata")) +  
            "chalKag/src/main/webapp/memberProfileImages"; // 윈도우 경로  
uploadDir = uploadDir.substring(0, uploadDir.indexOf("/WEB-INF")) + "/memberProfileImages"; // 맥북 경로
```

프로필 사진이 저장되는 경로 설정

회원가입 (2/4)

```
<div class="field">
    <label for="birth">birth</label> <select id="year" name="year"></select>
    <label for="year">year</label> <select id="month" name="month"></select>
    <label for="month">month</label> <select id="day" name="day"></select>
    <label for="day">day</label>
</div>
```

JoinPageAction.java

```
// String 타입과 file의 정보를 저장해야 하기 때문에 multipartRequest 사용 .노승현
MultipartRequest multipartRequest = new MultipartRequest(request, uploadDir, 1024 * 1024 * 10, "UTF-8")
memberDTO.setId(multipartRequest.getParameter("id"));
memberDTO.setPw(multipartRequest.getParameter("pw"));
memberDTO.setName(multipartRequest.getParameter("name"));
memberDTO.setNickname(multipartRequest.getParameter("nickname"));

// db 저장 시 생년월일을 하나의 String 타입으로 받기 때문에 하나의 String 타입으로 변환 시켜줌 .노승현
String year = multipartRequest.getParameter("year");
String month = multipartRequest.getParameter("month");
String day = multipartRequest.getParameter("day");
String memberBirth = year + "-" + month + "-" + day; // 형식에 맞게 조합
memberDTO.setBirth(multipartRequest.getParameter(memberBirth));
memberDTO.setPh(multipartRequest.getParameter("ph"));
```

년, 월, 일 3개의 값을 하나의 문자열 타입으로 가공

회원가입 (3/4)

EMAIL

이메일 중복확인

사용가능한 이메일입니다.

checkID.js

```
$ajax({
    type : "POST",
    url : "/chalKag/checkID.do",
    data : { 'id':id},
    dataType: 'text',
    success : function(data){
        if(data=='1'){
            $("#IDErrMsg").text("사용가능한 이메일 입니다.");
            $("#IDErrMsg").css("color","green");
        }
        else{
            $("#IDErrMsg").text("중복된 이메일입니다. 다시 입력해주세요");
            $("#IDErrMsg").css("color","red");
        }
    }
})
```

CheckIDAction.java

```
mDTO.setSearchCondition("ID중복검사");
mDTO.setId(request.getParameter("id"));
mDTO=mDAO.selectOne(mDTO);

//System.out.println("mDTO"+mDTO);
int flag=0;

// 저장된 아이디가 없다면 중복이 아니라면?
if(mDTO==null) {
    flag=1;
}
else {
    // 아이디가 중복이라면
}

PrintWriter out=response.getWriter();
out.print(flag);
System.out.println(flag);
```

MemberDAO.java

```
else if (memberDTO.getSearchCondition().equals("ID중복검사")) {
    try {
        pstmt = conn.prepareStatement(SELECTONE_IDCHECK);
        pstmt.setString(1, memberDTO.getId());
        ResultSet rs = pstmt.executeQuery();

        if (rs.next()) {
            data = new MemberDTO();
            data.setId(rs.getString("ID"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

중복검사는 비동기로 동작

Controller는 서블릿으로 작성

회원가입 (4/4)

JoinAction.java

```
boolean flag = memberDAO.insert(memberDTO);

if (flag) { // 성공시 메인으로 이동

    forward.setPath("error/alertPage.jsp");
    forward.setRedirect(false);
    request.setAttribute("status", "joinSuccess");
    request.setAttribute("title", "회원가입 성공!");
    request.setAttribute("text", "축하드립니다!");

} else { // 실패시 alert 창으로 이동

    forward.setPath("error/alert.jsp");
    forward.setRedirect(false);
    request.setAttribute("status", "joinFail");
    request.setAttribute("title", "회원가입 실패!");
    request.setAttribute("text", "다시 시도해주세요!");

}
```

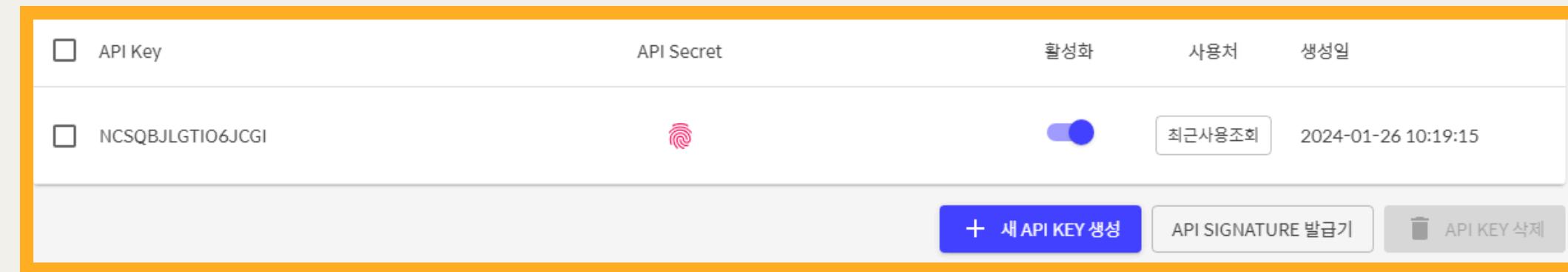
alertPage.jsp

```
} else if("${status}"=="joinSuccess"){
    Swal.fire({
        title: '${title}',
        text: '${text}',
        icon: 'success',
        confirmButtonText: '확인'
    }).then((result) => {
        if (result.isConfirmed) {
            location.href = "/chalKag/main.do";
        }
    });
}

else if("${status}"=="joinFail"){
    Swal.fire({
        title: '${title}',
        text: '${text}',
        icon: 'error',
        confirmButtonText: '확인'
    }).then((result) => {
        if (result.isConfirmed) {
            history.go(-1);
        }
    });
}
```

forward를 사용하여 setAttribute 값을 가지고 페이지 이동

문자 API (1/2)



AuthenticationSMS.java

```
public AuthenticationSMS(String api_key, String api_secret) {
    this.api_key = "REDACTED";
    this.api_secret = "REDACTED";
}
// 수신자 // 메시지 내용(인증번호)
public boolean sendMsg(String recipientPhoneNumber, String verificationCode) {
    Message coolsms = new Message(api_key, api_secret);

    HashMap<String, String> params = new HashMap<>();
    params.put("to", recipientPhoneNumber); // 수신자 전화번호
    params.put("from", "01066221689"); // 발신자 전화번호
    params.put("text", "인증 코드는: " + verificationCode); // 메시지 내용
    params.put("type", "sms"); // 메시지 타입

    try {
        JSONObject JO = (JSONObject) coolsms.send(params);
        // 선택적으로 디버깅을 위해 결과 JSON을 로그로 출력할 수 있습니다.
        System.out.println("SMS 전송 결과: " + JO.toJSONString());
        return true; // 메시지 전송 성공
    } catch (CoolsmsException e) {
        System.out.println("SMS 전송 오류: " + e.getMessage());
        System.out.println("오류 코드: " + e.getCode());
        return false; // 메시지 전송 실패
    }
}
```

phCheck.js

```
<div class="field">
    <!-- 번호를 입력 한.안승준 -->
    <label for="ph">phone number</label> <input type="text" name="ph" id="ph" placeholder="-빼고 입력해주세요 입력해주세요." maxlength="11" oninput="this.value = this.value.replace(/[^0-9.]/g, '').replace(/(\..*)\./g, '$1');" required />

    <!-- 인증번호 보내기 버튼.안승준 -->
    <input style="margin-top: 20px;" type="button" value="인증번호 보내기" id="phCheckBtn" onclick="sendAuthenticationSMS()" />

    <!-- 인증번호 입력 한.안승준 -->
    <input type="text" name="phCheck" id="phCheck" placeholder="인증번호를 입력해주세요" style="margin-top: 20px;" />
```

버튼을 누르면 문자를 전송하는 함수가 동작함

API Key와 발신자 번호가 같은 사람의 Key

문자 API (2/2)

sendAuthentic.js

```
var serverGeneratedCode="";
function sendAuthenticationSMS() {
    alert("인증번호 발송이 완료되었습니다.");
    // 사용자가 입력한 전화번호 가져오기
    var ph = $("#ph").val();

    // AJAX를 사용하여 서버에 전화번호 전송
    $.ajax({
        url: "/chalKag/sendAuthenticationSMS",
        type: "POST",
        dataType:"text",
        data: { ph: ph },
        success: function(data) {
            // 서버 응답에 따른 처리
            if (data !== "fail") {
                console.log("data "+data);
                alert("인증 보내기 성공");
                // 성공적으로 SMS를 보낸 경우 추가 동작을 수행할 수 있습니다.
                serverGeneratedCode=data;
            } else {
                alert("인증 보내기 실패");
                // SMS 전송 실패 시 사용자에게 알림을 표시할 수 있습니다.
            }
        }
    });
}
```

SendAuthentication.java

```
String ph = request.getParameter("ph");
AuthenticationSMS authenticationSMS = new AuthenticationSMS("NCS8QSCV2PUOXICE",
    "0HWUXNZ9BQV5I602DXDST5FY06WAOZRW");

// 인증코드 생성 로직작성
// CreateVerificationCode의 인스턴스 생성
// 인증 코드 생성해주는 클래스
CreateVerificationCode verificationCodeGenerator = new CreateVerificationCode();

// 인증 코드 생성
String verificationCode = verificationCodeGenerator.createVerificationCode();

// 인증 SMS 전송
boolean isSMSsent = authenticationSMS.sendMsg(ph, verificationCode);

if (isSMSsent) {
    response.getWriter().write(verificationCode);
    request.setAttribute("result", verificationCode);
} else {
    response.getWriter().write("fail");
}
```

smsCheck.js

```
// 이 함수는 서버에서 전송한 인증코드와 사용자가 입력한 코드를 비교하고 결과를 처리합니다.
$(document).ready(function() {
    $("#smsCheck").on('click', function() {
        console.log("smsCheck 동작함");
        console.log(serverGeneratedCode);
        if ($("#phCheck").val() ==serverGeneratedCode) {
            $(".successPhCheck").text("인증번호가 일치합니다.");
            $(".successPhCheck").css("color", "green");
        } else {
            $(".successPhCheck").text("인증번호가 일치하지 않습니다. 확인해주시기 바랍니다.");
            $(".successPhCheck").css("color", "red");
            $(this).attr("autofocus", true);
        }
    });
});
```

CreateVerificationCode.java

```
package controller.sms;

public class CreateVerificationCode {
    public String createVerificationCode() {
        char[] charSet = new char[] { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9' };
        StringBuilder verificationCodeBuilder = new StringBuilder();

        for (int i = 0; i < 6; i++) {
            int randomIndex = (int) (charSet.length * Math.random());
            verificationCodeBuilder.append(charSet[randomIndex]);
        }

        String verificationCode = verificationCodeBuilder.toString();
        System.out.println(verificationCode);

        return verificationCode;
    }
}
```

랜덤한 숫자 6자리를
전송하는 코드

ERRORPAGE 무한 탐색 오류(1/2)

에러 발생 시 페이지 경로를 무한으로 탐색

```
localhost:8088/chalKag/common/error/error/error/error/
```

에러페이지 경로 설정

```
ActionForward forward = new ActionForward();  
forward.setPath("error/errorPage.jsp");  
forward.setRedirect(true);
```

ERRORPAGE 무한 탐색 오류(2/2)

Path

/chalKag

```
ActionForward forward = new ActionForward();
forward.setPath("/chalKag/error/errorPage.jsp");
forward.setRedirect(true);

return forward;
```

/chalKag으로 파일 탐색 시작 경로 수정

게시글 작성 및 수정

카카오 API 로그인

- 게시글 작성 및 수정
- 카카오 API

게시글 작성/수정 (1/4)

게시글 작성 화면 VIEW

The screenshot shows a form for creating or editing a post. It includes fields for Title, ProductName, Price, productCategory, Company, an image upload button labeled "UPLOAD", and a CKEditor content area with a toolbar. A red border highlights the entire form area.

Title

ProductName

Price

productCategory

Company

UPLOAD

B I H L G V

Enter your post

POST BOARD

게시글 작성 화면 VIEW

이 입력란을 작성하세요.

이미지 업로드 버튼 :
사용자의 이미지 1장을 업로드

CKEditor 내용 입력칸 :
작성한 글씨에 여러 효과를 적용시켜
내용을 입력

게시글 작성/수정 (2/4)

boardWrite.js & Update.js



Enter your post

```
<script src="https://cdn.ckeditor.com/ckeditor5/34.0.0/classic/ckeditor.js"></script>
<script src="https://cdn.ckeditor.com/ckeditor5/34.0.0/classic/translations/ko.js"></script>

ClassicEditor // CK 에디터 설정
  .create( document.querySelector("#contents"), {      // 에디터 생성
    toolbar: [ 'bold', 'italic', 'bulletedList', 'numberedList', 'blockQuote', 'insertTable' ],
    // 사용할 에디터 버튼: 굵기, 기울기, 점 리스트, 숫자 리스트, 블록 처리, 표 만들기
    language: "ko"    // 언어 설정
  })
  .then(editor => { // 에디터 값 설정
    const form = document.querySelector("form");    // form 태그
```

CKEditor : CDN으로 호출하여 구현

toolbar를 통해 사용자가 적용할 수 있는 기능들을 설정

language로 언어를 설정

게시글 작성/수정 (3/4)

오류 메세지

```
org.springframework.web.multipart.MultipartException: Could not parse multipart servlet request, nested exception is java.lang.IllegalStateException: Unable to process parts as no multipart configuration has been provided
    org.springframework.web.multipart.support.StandardMultipartHttpServletRequest.parseRequest(StandardMultipartHttpServletRequest.java:111)
    org.springframework.web.multipart.support.StandardMultipartHttpServletRequest.<init>(StandardMultipartHttpServletRequest.java:85)
    org.springframework.web.multipart.support.StandardMultipartHttpServletRequest.<init>(StandardMultipartHttpServletRequest.java:72)
    org.springframework.web.multipart.support.MultipartResolutionDelegate.adaptToMultipartHttpServletRequest(MultipartResolutionDelegate.java:80)
    org.springframework.web.multipart.support.MultipartResolutionDelegate.resolveMultipartArgument(MultipartResolutionDelegate.java:102)
    org.springframework.web.method.annotation.RequestParamMethodArgumentResolver.resolveName(RequestParamMethodArgumentResolver.java:162)
    org.springframework.web.method.annotation.AbstractNamedValueMethodArgumentResolver.resolveArgument(AbstractNamedValueMethodArgumentResolver.java:98)
    org.springframework.web.method.support.HandlerMethodArgumentResolverComposite.resolveArgument(HandlerMethodArgumentResolverComposite.java:121)
    org.springframework.web.method.support.InvocableHandlerMethod.getMethodArgumentValues(InvocableHandlerMethod.java:161)
    org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.java:128)
    org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHandle(ServletInvocableHandlerMethod.java:114)
    org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandlerMethod(RequestMappingHandlerAdapter.java:827)
    org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal(RequestMappingHandlerAdapter.java:738)
    org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.handle(AbstractHandlerMethodAdapter.java:85)
    org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:963)
    org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:887)
    org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:970)
    org.springframework.web.servlet.FrameworkServlet.doPost(FrameworkServlet.java:872)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:648)
    org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:846)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:729)
    org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)
    org.springframework.web.filter.CharacterEncodingFilter.doFilterInternal(CharacterEncodingFilter.java:197)
    org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:107)
```

[\[ion: Posted content type isn't multipart/form-data\]](#)

form 태그로 이미지 업로드를 할 경우

속성으로 enctype="multipart/form-data"를 설정하지 않았을 때 발생한 오류

게시글 작성/수정 (4/4)

BoardWrite.java & BoardUpdate.java

```
// 이미지 업로드 객체 선언(값, 절대경로, 사이즈, 인코딩)
MultipartRequest multipartRequest = new MultipartRequest(request, uploadDir, 1024 * 1024 * 10, "UTF-8");
```

cos.jar 내부의 **MultipartRequest** 클래스를 사용
이미지를 포함한 다른 입력 값들을 수령

```
// 이미지 업로드를 처리하는 기능
File uploadedFile = multipartRequest.getFile("file");
if (uploadedFile != null && uploadedFile.exists()) {
    String originalFilename = uploadedFile.getName(); // 파일명 저장하는 변수
    String extension = FilenameUtils.getExtension(originalFilename); // 확장자를 저장하는 변수
    String newFilename = UUID.randomUUID().toString() + "." + extension; // 새로운 파일명과 확장자를 저장하는 변수
    String filePath = uploadDir + File.separator + newFilename; // 위 내용을 전부 통합하여 저장하는 변수
    boardDTO.setImage(newFilename);
    // 파일 객체 선언 후 파일 위치를 객체에 저장한다.
    File newFile = new File(filePath);
    // 파일을 새 위치로 이동시킵니다.
    uploadedFile.renameTo(newFile);
}
```

010f1bda-8cb9-4ddd-8e32-ab9d1f615dab.png

변경된 이미지 이름

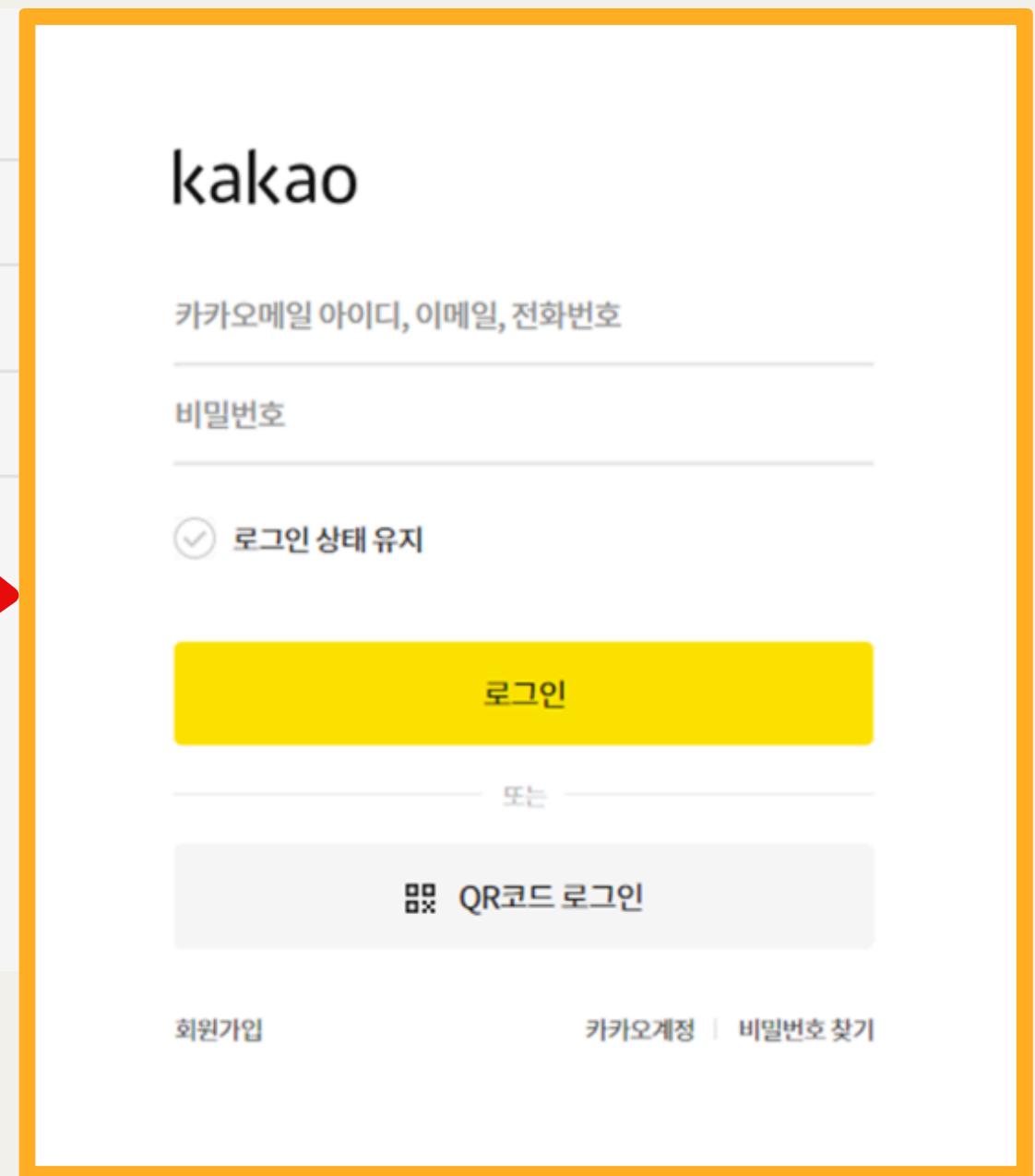
이미지 파일 이름 중복 방지를 위해 **UUID** 클래스를 사용하여 이름을 변경한 뒤 저장

카카오 로그인 API (1/5)

카카오 로그인 버튼



카카오 계정으로 로그인 시
로그인 후 메인 화면으로 이동
비 회원일 경우 자동으로 회원가입



카카오 로그인 API (2/5)

login.jsp

```
<a href="javascript:kakaoLogin();">
    
</a>

<script type="text/javascript" src="https://developers.kakao.com/sdk/js/kakao.js"></script>
```

카카오에서 제공한 로그인 API를 사용할 수 있는 JS코드와 버튼 이미지를 사용

kakaoLogin.js

```
Kakao.init('████████████████'); // 발급받은 키 중 javascript키를 사용해준다.
console.log(Kakao.isInitialized()); // sdk초기화여부판단
// 카카오 로그인
function kakaoLogin() {
    Kakao.Auth.login({           // 카카오 로그인 메서드 호출
        success: function(authResponse) {
            Kakao.API.request({
                url: '/v2/user/me',          // 로그인하는 유저의 카카오 계정 정보를 가져올 수 있는 URL

```

발급받은 키로 카카오 developer에 접속 후 카카오 로그인 메서드를 호출

카카오 로그인 API (3/5)

kakaoLogin.js

```
$.ajax({
    url: '/chalKag/kaKaoLogin.do', // 서블릿 로그인(비동기) 주소
    type: 'post', // post로 전달
    data: {
        memberID: userResponse.kakao_account.email // 카카오 사용자 ID
    },
})
```

kakao

카카오메일 아이디, 이메일, 전화번호

비밀번호

로그인 상태 유지

로그인

— 또는 —

QR코드로그인

회원가입

카카오계정 | 비밀번호 찾기

KakaoLogin.java

```
if (memberDTO != null) { // 로그인 성공시 세션 저장 후 메인으로 이동, 이동 할 정보 없음
    if (!memberDTO.getGrade().equals("탈퇴")) { // 로그인한 유저의 등급이 탈퇴가 아닐 경우에만 로그인할 수 있게 유효성 추가
        HttpSession session = request.getSession(); // 세션 초기값 설정
        session.setAttribute("member", memberDTO.getId()); // 로그인 성공시 세션에 ID를 저장한다.
        System.out.println("[KaKaoLoginAction] 로그인 성공");
        response.getWriter().println("{\"isNewUser\": false}"); // 이미 있는 회원이기에 View에 DB에 있는 회원이라는 것을 알리기 위해 isNewUser의 값을 false 라고 전달한다.
        // isNewUser이 false이면 view에선 로그인 성공이라고 인식

    } else {
        System.out.println("[KaKaoLoginAction] 로그인 실패"); // 탈퇴한 회원일 경우 view에서 로그인 실패 메세지 출력
    }
} else {
    System.out.println("[KaKaoLoginAction] 없는 회원입니다.");
    response.getWriter().println("{\"isNewUser\": true}"); // 없는 회원이기에 View에 DB에 없는 회원이라는 것을 알리기 위해 isNewUser의 값을 true 라고 전달한다.
}
```

Controller에서 if문을 통해 회원정보의 유무를 판단해서
다시 JS로 true, false 값을 전달 변수가 true면 회원가입 ajax코드로 이동

카카오 로그인 API (4/5)

kakaoLogin.js - join ajax

```
if (responseObj.isNewUser) { // isNewUser : 해당 이메일의 정보가 DB에 있는지 유무를 판단하는 변수  
  
    // 카카오 API에서 받아온 출생년도와 생일 정보  
    var birthyear = userResponse.kakao_account.birthyear; // 예: '1990'  
    var birthday = userResponse.kakao_account.birthday; // 예: '0525'  
  
    // 월과 일을 분리  
    var month = birthday.substring(0, 2);  
    var day = birthday.substring(2);  
  
    // 년도, 월, 일을 결합 예 : 1990-05-25  
    var fullBirthday = birthyear + '-' + month + '-' + day;
```

```
$.ajax({  
    url: '/chalKag/kakaoJoin.do', // 서블릿 회원가입(비동기 처리) 주소  
    type: 'post', // post로 전달  
    data: {  
        memberID: userResponse.kakao_account.email, // 카카오 사용자 ID  
        name: userResponse.properties.nickname, // 카카오 사용자 닉네임  
        nickname: userResponse.properties.nickname, // 카카오 사용자 닉네임  
        memberBirth: fullBirthday, // 카카오 사용자 생일  
        ph: userResponse.kakao_account.phone_number // 카카오 사용자 전화번호  
    },
```

회원가입을 위해 카카오 계정에서 필요한 정보들을 받아올 수 있도록 구현

카카오 계정에 저장되어 있는 생일 정보를 변환해 Controller로 전달



카카오 로그인 API (5/5)

kakaoJoin.java

```
// 카카오 계정의 이메일을 저장한다.  
memberDTO.setId(request.getParameter("memberID"));
```

```
// UUID 생성  
UUID uuid = UUID.randomUUID();  
// UUID의 앞 8자리를 임시 비밀번호로 사용  
String temporaryPassword = uuid.toString().substring(0, 8);
```

```
// memberDTO에 임시 비밀번호 설정  
memberDTO.setPw(temporaryPassword);
```

```
// 카카오 계정에서 가져온 이름, 닉네임, 생일을 저장한다.  
memberDTO.setName(request.getParameter("name"));  
memberDTO.setNickname(request.getParameter("nickname"));  
memberDTO.setBirth(request.getParameter("memberBirth"));
```

```
// 카카오 계정에서 가져온 번호 : +82 10-xxxx-xxxx  
String ph = request.getParameter("ph");  
// +82 를 replace 메소드로 제거  
ph = ph.replace("+82 ", "0");  
// -를 replace 메소드로 제거  
ph = ph.replace("-", "");  
// 010xxxxxxxx 형식으로 변경된 번호를 저장한다.  
memberDTO.setPh(ph);
```

```
// DTO에 저장된 값을 DAO로 전달하여 회원가입  
boolean flag = memberDAO.insert(memberDTO);
```

카카오 API 회원가입 Controller

임시 비밀 번호를 생성해 DB에 저장

PW
9364d9e9

카카오 계정에 저장된 전화번호를 변경하는 로직
+82 , - 가 붙어있는 번호는 수정

+ 82 10-0000-0000 =>
01000000000

네이버 로그인 API

아이디/비밀번호 찾기

오류 발견 및 해결 방안

- 네이버 로그인 API
- 아이디/비밀번호 찾기
- 오류 발견 및 해결 방안

네이버 로그인 API (1/5)

이메일을 입력해주세요

비밀번호를 입력해주세요

N 로그인

아이디 찾기 / 비밀번호 찾기 / 회원가입

로그인

CHALKAG

전체 동의하기

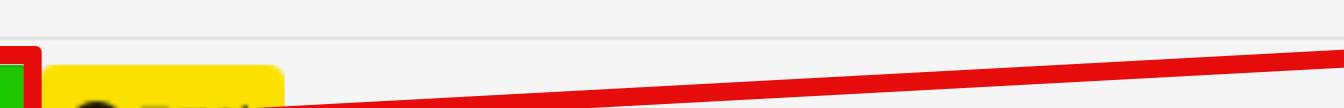
CHALKAG에서 plzjun0731회원님의 개인정보에 접근합니다.
제공된 개인정보(이용자 식별자 기본제공, 그 외 제공 항목이 있을 경우 아래 별도 기재)는 이용자 식별, 통계, 계정 연동 및 CS 등을 위해 서비스 이용기간 동안 활용/보관 됩니다. 본 제공 동의를 거부할 권리가 있으나, 동의를 거부하실 경우 서비스 이용이 제한될 수 있습니다.

필수 제공 항목 (필수)

✓ 출생연도 ✓ 휴대전화번호 ✓ 이름
✓ 이메일 주소 ✓ 별명 ✓ 생일

동의 후에는, 해당 서비스의 이용약관 및 개인정보처리방침에 따라 정보가 관리됩니다.

취소 동의하기



네이버 로그인 API (2/5)

```
<a id="naverLoginLink" href="#">
```

```
= "2jGYcIGvm6EzQohaHOcs"; // 애플리케이션 클라이언트 아이디값.안승준
I = encodeURIComponent("http://localhost:8088/chalKag/naverLogin.do"); // 네이버 로그인 후 리디렉션될
th.random().toString(36).substring(2, 15) + Math.random().toString(36).substring(2, 15); // URI.안승준
https://nid.naver.com/oauth2.0/authorize?response_type=code" + "&client_id=" + clientId
ct_uri=" + redirectURI + "&state=" + state; // CSRF 공격 방지를 위한 URL 생성.안승준
(apiURL); // 네이버 OAuth 인증 요청을 위한 URL 생성.안승준
.setItem("state", state); // 상태 토큰을 세션 스토리지에 저장.안승준
```

```
크를 클릭했을 때 실행될 함수 등록.안승준
lementById("naverLoginLink").addEventListener("click", function() {
ocation.href = apiURL; // 네이버 OAuth 인증 페이지로 리디렉션.안승준
```

a 태그 이용 버튼 생성

클라이언트 아이디 값이 포함 된 URL 설정
네이버 인증 요청에 필요한 state값을 저장

네이버 로그인 API (3/5)

NaverLoginAction.java

```
// 네이버 어플리케이션 클라이언트 아이디.안승준
String clientId = "2jGYcIGym6EzQohgH0cs";
// 네이버 어플리케이션 클라이언트 시크릿.안승준
String clientSecret = "O8RsLA0BDo";

// 네이버 인증 요청에서 받은 code와 state를 가져옴.안승준
String code = request.getParameter("code");
String state = request.getParameter("state");
// 네이버 인증 요청에서 받은 code와 state 확인 로그.안승준
System.out.println("[NaverLoginAction]code 로그 = ["+code+"]");
System.out.println("[NaverLoginAction]state 로그 = ["+state+"]");

// 리다이렉트 URI 인코딩.안승준
String redirectURI = URLEncoder.encode("http://localhost:8080/test/naverLogin");
String apiURL;

// 네이버 인증 API URI 생성.안승준
apiURL = "https://nid.naver.com/oauth2.0/token?grant_type=client_credentials";
apiURL += "&client_id=" + clientId;
apiURL += "&client_secret=" + clientSecret;
apiURL += "&redirect_uri=" + redirectURI;
apiURL += "&code=" + code;
apiURL += "&state=" + state;

String access_token = "";
String refresh_token = "";

// API URI 확인 로그.안승준
System.out.println("[NaverLoginAction]apiURL 로그 =" + apiURL);

try {
    // 네이버 인증 요청.안승준
    URL url = new URL(apiURL);
    HttpURLConnection con = (HttpURLConnection) url.openConnection();
    con.setRequestMethod("GET");
    int responseCode = con.getResponseCode();
    BufferedReader br;

    // responseCode 확인 로그.안승준
    System.out.print("[NaverLoginAction]responseCode 로그= ["+responseCode+"]");

    if (responseCode == 200) { // 정상 호출시 수행.안승준
        br = new BufferedReader(new InputStreamReader(con.getInputStream()));
    } else { // 에러 발생시 수행.안승준
        br = new BufferedReader(new InputStreamReader(con.getErrorStream()));
    }

    String inputLine;
    StringBuffer res = new StringBuffer();

    while ((inputLine = br.readLine()) != null) {
        res.append(inputLine);
    }

    // br = new BufferedReader(new InputStreamReader(con.getInputStream()));.안승준
    // HTTP 연결을 통해 들어오는 데이터를 읽기 위해 사용.안승준
    // con.getInputStream()은 이진 형태로 데이터 제공.안승준
    // BufferedReader를 통해 이진을 텍스트 형태로 변환.안승준
}
```

네이버 인증 요청 API URL 생성

네이버 API 인증 요청

네이버 로그인 API (4/5)

```
if (responseCode == 200) {  
  
    JSONParser parsing = new JSONParser();  
    Object obj = parsing.parse(res.toString());  
    JSONObject jsonObj = (JSONObject) obj;  
  
    // 네이버 인증 응답에서 access_token과 refresh_token을 가져옴. 안승준  
    access_token = (String) jsonObj.get("access_token");  
  
    // 네이버 인증 access_token 확인 로그. 안승준  
    System.out.println("[NaverLoginAction]access_token 로그 = [" + access_token + "]");  
  
    refresh_token = (String) jsonObj.get("refresh_token");  
  
    // 네이버 인증 refresh_token 확인 로그. 안승준  
    System.out.println("[NaverLoginAction]refresh_token 로그 = [" + refresh_token + "]");  
  
    // 사용자 프로필 API 호출  
    String header = "Bearer " + access_token;  
    String apiUrl = "https://openapi.naver.com/v1/nid/me";  
    URL profileUrl = new URL(apiUrl);  
    HttpURLConnection profileCon = (HttpURLConnection) profileUrl.openConnection();  
    profileCon.setRequestMethod("GET");  
    profileCon.setRequestProperty("Authorization", header);  
    int profileResponseCode = profileCon.getResponseCode();  
    BufferedReader profileBr;  
  
    if (profileResponseCode == 200) {  
        profileBr = new BufferedReader(new InputStreamReader(profileCon.getInputStream()));  
    } else {  
        profileBr = new BufferedReader(new InputStreamReader(profileCon.getErrorStream()));  
    }  
  
    String profileInputLine;  
    StringBuffer profileRes = new StringBuffer();  
    while ((profileInputLine = profileBr.readLine()) != null) {  
        profileRes.append(profileInputLine);  
    }  
  
    profileBr.close();
```

JSONParser를 통해 인증 정보를 파싱
파싱된 토큰을 JSONObject로 변환
access_token, refresh_token 추출

사용자 프로필 토큰 생성 API 호출
토큰을 사용해 회원 정보 추출

네이버 로그인 API (5/5)

```
if (profileResponseCode == 200) {  
    System.out.println(profileRes.toString());  
    JSONObject profileObj = (JSONObject) parsing.parse(profileRes.toString());  
    JSONObject responseObj = (JSONObject) profileObj.get("response");  
    String email = (String) responseObj.get("email");  
    String name = (String) responseObj.get("name");  
    String nickname = (String) responseObj.get("nickname");  
    String birthyear = (String) responseObj.get("birthyear");  
    String birthday = (String) responseObj.get("birthday");  
    String mobile = (String) responseObj.get("mobile");  
  
    // 출력 데이터 확인 로그.안승준  
    System.out.println(email);  
    System.out.println(name);  
    System.out.println(nickname);  
    System.out.println(birthyear);  
    System.out.println(birthday);  
    System.out.println(mobile);
```

회원DTO와 회원DAO를 이용해 로그인 수행

미가입 회원의 회원가입을 자동으로 실행

회원 정보를 JSON 형식으로 파싱
profileObj에 회원 토큰을 저장

저장된 JSON 형식의 자료형에서
원하는 회원 정보 추출

```
// 로그인 기능 수행.안승준  
MemberDAO memberDAO = new MemberDAO();  
MemberDTO memberDTO = new MemberDTO();  
memberDTO.setId(email);  
memberDTO.setSearchCondition("로그인");  
memberDTO.setSnsLoginCondition("SNS로그인");  
memberDTO = memberDAO.selectOne(memberDTO);  
  
System.out.println("[NaverLoginAction] = " + memberDTO);  
  
if (memberDTO != null) {  
    if (!memberDTO.getGrade().equals("탈퇴")) {  
        HttpSession session = request.getSession();  
        session.setAttribute("member", memberDTO.getId());  
        System.out.println("[NaverLoginAction]로그인 성공");  
  
        response.sendRedirect("/chalKag/main.do");  
    } else {  
        System.out.println("[NaverLoginAction] 로그인 실패");  
    }  
} else {  
    System.out.println("[NaverLoginAction] 없는 회원입니다.");  
  
    request.setAttribute("naverMember", responseObj);  
    RequestDispatcher dispatcher = request.getRequestDispatcher("/naverJoin.do");  
    dispatcher.forward(request, response);  
}
```

아이디 찾기/비밀번호 찾기(1/3)

**이름과 전화번호를 사용자가 입력
전화번호 확인을 위해 문자 API를 사용**

아이디 찾기/비밀번호 찾기(2/3)

```
ActionForward forward = new ActionForward();
forward.setPath("/chalKag/common/findIdResultPage.jsp");
forward.setRedirect(false);

MemberDAO memberDAO = new MemberDAO();
MemberDTO memberDTO = new MemberDTO();

// 사용자의 이름.안승준
memberDTO.setName(request.getParameter("name"));

// 사용자의 전화번호 입력.안승준
memberDTO.setPh(request.getParameter("ph"));

// searchCondition 설정.안승준
memberDTO.setSearchCondition("아이디찾기");

// 아이디 찾기 로직 실행.안승준
memberDTO = memberDAO.selectOne(memberDTO);

if (memberDTO != null) {
    request.setAttribute("findIdResult", memberDTO.getId());
} else {
    request.setAttribute("findIdResult", "가입 된 아이디가 없습니다!");
}

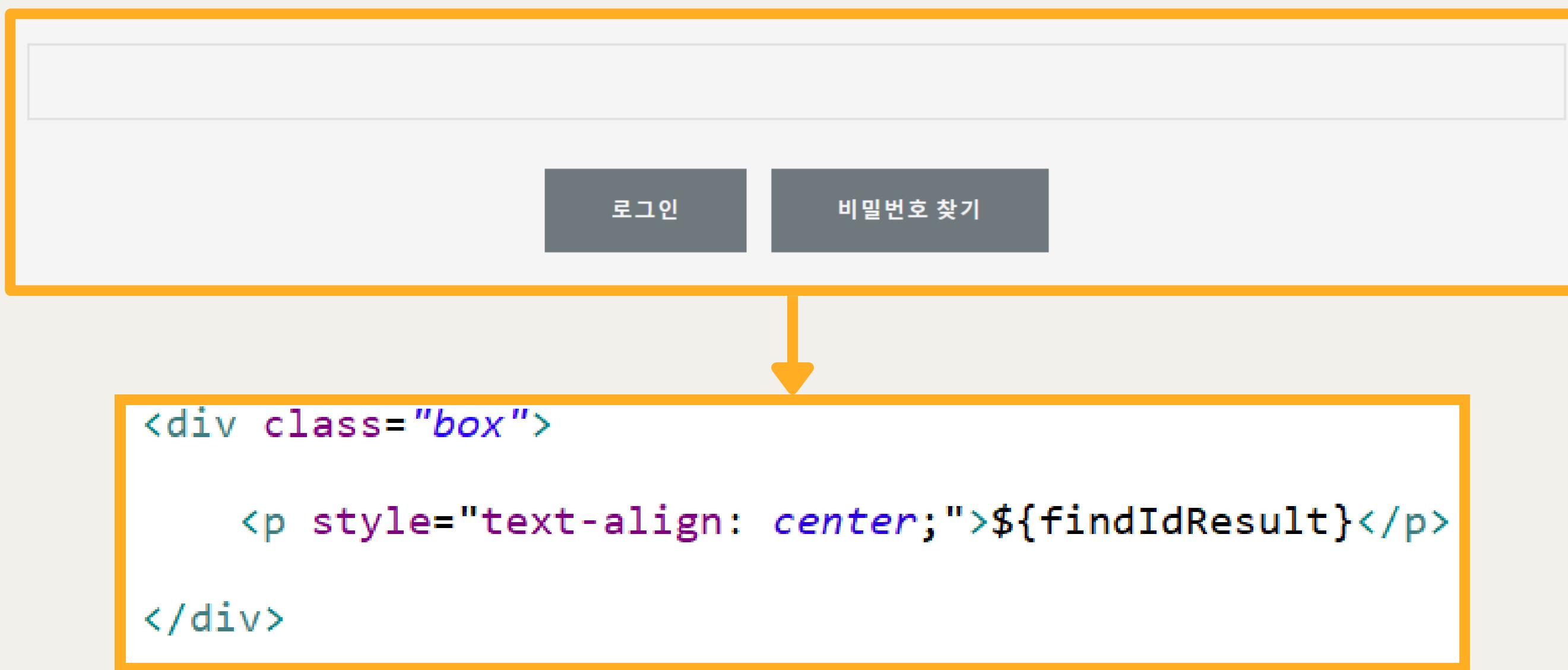
return forward;
```

회원 DTO를 생성
사용자가 입력한 이름과 전화번호를 설정

회원DAO의 아이디 찾기 함수를 통해
아이디 값을 VIEW로 전송

```
// 아이디 찾기 ► 이름, 전화번호 받아서 아이디 찾기 .정석진
private static final String SELECTONE_FINDID = "SELECT ID FROM MEMBER WHERE NAME=? AND PH=?";
```

아이디 찾기/비밀번호 찾기(3/3)



Controller에서 수신한 아이디 값을 EL을 이용해 결과 화면에 출력

비밀번호 찾기 동일 로직 수행

네이버 로그인 모듈화 오류(1/4)

```
<%  
    String clientId = "2jGYcIGym6EzQohgHOcs"; //애플리케이션 클라이언트 아이디값;  
    String redirectURI = URLEncoder.encode("http://localhost:8088/chalKag/naverLogin.do", "UTF-8");  
    SecureRandom random = new SecureRandom();  
    String state = new BigInteger(130, random).toString();  
    String apiURL = "https://nid.naver.com/oauth2.0/authorize?response_type=code" + "&client_id=" + clientId  
        + "&redirect_uri=" + redirectURI + "&state=" + state;  
    session.setAttribute("state", state);  
%>  
  
<a href="<%=apiURL%>"></a>
```

.jsp 파일 내의 java 코드를 제거 하기 위해
로그인 페이지로 이동해주는 .java 파일로 해당 코드 이동

```
ActionForward forward = new ActionForward();  
  
String clientId = "2jGYcIGym6EzQohgHOcs"; //애플리케이션 클라이언트 아이디값;  
String redirectURI = URLEncoder.encode("http://localhost:8088/chalKag/naverLogin.do", "UTF-8");  
SecureRandom random = new SecureRandom();  
String state = new BigInteger(130, random).toString();  
String apiURL = "https://nid.naver.com/oauth2.0/authorize?response_type=code" + "&client_id=" + clientId  
    + "&redirect_uri=" + redirectURI + "&state=" + state;  
  
HttpSession session=request.getSession();  
session.setAttribute("state", state);  
  
request.setAttribute("apiURL", apiURL);  
  
forward.setPath("/common/loginPage.jsp");  
forward.setRedirect(false);  
  
return forward;
```

네이버 로그인 모듈화 오류(2/4)

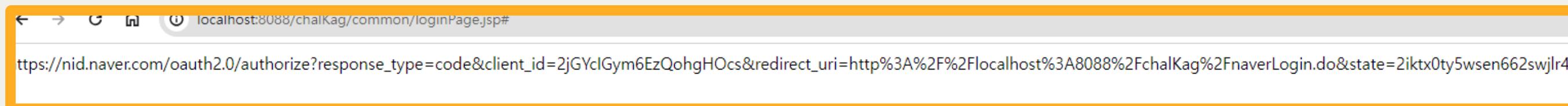
네이버 로그인 기능을 사용하지 않아도
API URL 정보를 갖고 다니는 것은 비효율적이라고 판단
스크립트로 코드 수정
메서드가 실행 될 때만 URL을 생성

```
// 네이버 로그인 링크를 클릭했을 때 실행될 함수. 안승준
document.getElementById("naverLoginLink").addEventListener("click", function() {

    var clientId = "2jGYcIGym6EzQohgHOcs"; // 애플리케이션 클라이언트 아이디값. 안승준
    var redirectURI = encodeURIComponent("http://localhost:8088/chalKag/naverLogin.do"); // 네이버 로그인 후 리디렉션될 URI. 안승준
    var state = Math.random().toString(36).substring(2, 15) + Math.random().toString(36).substring(2, 15); // CSRF 공격 방지를 위한 상태 토큰 생성. 안승준
    var apiURL = "https://nid.naver.com/oauth2.0/authorize?response_type=code" + "&client_id=" + clientId
        + "&redirect_uri=" + redirectURI + "&state=" + state; // 네이버 OAuth 인증 요청을 위한 URL 생성. 안승준
    document.write(apiURL); // 생성된 URL을 출력하여 확인 (실제 사용시에는 제거). 안승준
    session.setItem("state", state); // 상태 토큰을 세션 스토리지에 저장. 안승준

    window.location.href = apiURL; // 네이버 OAuth 인증 페이지로 리디렉션. 안승준
});
```

URL이 정상적으로 생성 되었지만 로그인 페이지로 넘어가지 않는 오류 확인



네이버 로그인 모듈화 오류(3/4)

Controller에 로그 작성을 통해
네이버로부터 code와 state를 송신하지 못하는 것을 발견

```
String code = request.getParameter("code");
String state = request.getParameter("state");

System.out.println("NaverLoginAction.java = ["+code+"]");
System.out.println("NaverLoginAction.java = ["+state+"]");
```

구글링을 통해 JavaScript에서는 session을
사용 할 수 없다는 사실을 알게됨

```
+ "&redirect_uri=" + redirectURI + "&state=" + state;
document.write(apiURL); // 생성된 URL을 출력하여 확인 (실제 사용시에는
session.setItem("state", state); // 상태 토큰을 세션에 저장. 안승준
```

네이버 로그인 모듈화 오류(4/4)

```
document.write(apiURL); // 생성된 URL을 출력하여 확인(실제 사용시에는 제거).안승준  
session.setItem("state", state); // 상태 토큰을 세션에 저장.안승준
```

```
window.location.href = apiURL; // 네이버 OAuth 인증 페이지로 리디렉션.안승준  
});
```

what kind of error can occur in this code?

i wanna get state value from naver and send to controller but it dosent work

**chatGPT를 사용
네이버로 부터 state값을
송신받아 Controller에게
전달하고 싶지만 작동하지 않는 이유**

1. **Incorrect `session` Method:** The code uses `session.setItem("state", state);`, which appears to be an incorrect method for storing the state in the session storage. The correct method is `sessionStorage.setItem("state", state);`. This mistake would prevent the state from being stored properly, affecting your ability to retrieve it later.

```
document.write(apiURL); // 생성된 URL을 출력  
sessionStorage.setItem("state", state);
```

sessionStorage를 사용한 코드로 변경

게시글 출력 및 삭제

- 게시판 전체 출력
- 게시글 상세 보기
- 게시글 삭제

게시글 전체 출력 (1/3)

BoardDAO.java

```
// 게시글 목록 전체 출력. 전미지          // 최종 쿼리 : 게시글들의 정보를 받아오며 작성자의 닉네임과 아이디를 받아옴
private static final String SELECTALL = "SELECT FINAL_DATA.* , MEMBER.NICKNAME, MEMBER.ID "
+ "FROM ( " // 4번째 서브쿼리 : 일련번호를 부여한 게시글의 정보를 받아오는데 좋아요 수의 값이 'null'일 경우 0으로 초기화
+ "      SELECT BOARD_DATA.* , COALESCE(RECOMMEND_DATA.RECOMMENDCNT, 0) AS RECOMMENDCNT "
+ "      FROM ( " // 2번째 서브쿼리 : 받아온 게시글들의 일련번호를 새로 부여
+ "          SELECT ROWNUM, ROWNUM_DATA.* "
+ "          FROM ( " // 1번째 서브쿼리 : 요청받은 게시판 종류과 동일한 게시판의 게시글 정보들을 받아옴
+ "              SELECT BOARDNUM, ID, CATEGORY, TITLE, CONTENTS, TO_CHAR(BOARDDATE, 'YYYY-MM-DD') AS BOARDDATE, "
+ "                  PRICE, PRODUCTCATEGORY, COMPANY, STATE, VIEWCOUNT "
+ "              FROM BOARD "
+ "              WHERE CATEGORY = ?"
+ "              ORDER BY BOARDNUM ASC "
+ "          ) ROWNUM_DATA "
+ "      ) BOARD_DATA "
+ "      LEFT JOIN ( " // 3번째 서브쿼리 : 게시글의 좋아요 수를 합산
+ "          SELECT BOARDNUM, COUNT(BOARDNUM) AS RECOMMENDCNT "
+ "          FROM RECOMMEND "
+ "          GROUP BY BOARDNUM "
+ "      ) RECOMMEND_DATA ON BOARD_DATA.BOARDNUM = RECOMMEND_DATA.BOARDNUM "
+ "      ORDER BY BOARD_DATA.BOARDNUM DESC "
+ "  ) FINAL_DATA "
+ "  JOIN MEMBER ON MEMBER.ID = FINAL_DATA.ID"; // 회원 테이블과 게시글 테이블인 FINAL_DATA를 JOIN
// 조인한 게시판 테이블 : 회원 테이블, 좋아요 테이블
// 사용한 컬럼(보여줄 목록) : 게시글 넘버, 글 제목, 작성자 아이디(회원 테이블), 작성자 닉네임(회원 테이블),
// 작성일, 좋아요 넘버(좋아요 테이블), 조회수, 판매상태, 카운트 함수 사용(좋아요수-좋아요 테이블 / 좋아요 값이 있을 때만 보여짐)
// 검색 조건 : 카테고리 검색 (선택한 카테고리의 게시글만 전체 출력 / CATEGORY에 값이 없으면 오류 발생 주의)
```

COUNT 함수를 사용해
게시글의 좋아요 수를 합산한 뒤
LEFT JOIN으로 가져옴

게시글 전체 출력 (2/3)

게시글이 있을 경우

VIEW에서 스크립트를 통해 바로 페이지 처리 할 수 있도록
데이터들을 JSON 형식으로 변환 후 VIEW로 전달

BoardSeleteAllAction.java

```
// selectAll 메서드를 통해 데이터 베이스에서 모든 게시글 정보를 가져와서 boardDatas에 저장
boardDatas = boardDAO.selectAll(boardDTO);
System.out.println("[로그] CameraReviewSelectAllPageAction.java 2 boardDatas 출력 : " + boardDatas);

if (boardDatas != null) { // 게시글 정보(boardDatas)가 있다면
System.out.println("[로그] CameraReviewSelectAllPageAction.java 3 boardDatas 출력 : " + boardDatas);

    Gson gson = new Gson(); // Gson 객체 생성
    String jsonBoardDatas = gson.toJson(boardDatas); // boardDatas를 JSON 형식으로 변환

    System.out.println("[로그] CameraReviewSelectAllPageAction.java 3 " + jsonBoardDatas);
    request.setAttribute("jsonBoardDatas", jsonBoardDatas); // JSON 형식의 boardDatas를 request 객체에 저장
    request.setAttribute("category", boardDTO.getCategory()); // 게시판 카테고리 정보를 request 객체에 저장

    // 데이터를 보내줄 페이지와 데이터 전송 방식
    forward.setPath("board/cameraReviewSelectAllPage.jsp"); // 포워드 객체 생성 후 JSP 페이지 경로 설정
    // 데이터를 cameraReviewSelectAllPage.jsp로 보냄
    forward.setRedirect(false);
    // 데이터를 보낼 때 리다이렉트(==데이터 없음)가 아니라면 (결과적으로 데이터가 있다면) 포워드 방식(==데이터 있음)으로 보냄
} else { // 게시글 정보(boardDatas)가 없다면
```

게시글 전체 출력 (3/3)

boardSelectAll.jsp

The screenshot shows a search interface with various filters and a table of results.

Search Filters (Top Row):

- 최저 금액 : 0원 (Lowest Price: 0 won)
- 최고 금액 : 0원 (Highest Price: 0 won)

Search Filters (Second Row):

- 제조사: 캐논, 소니, 니콘
- 카메라 기종: DSLR, 미러리스, 컴팩트

Search Input:

- 제목 dropdown
- 검색어를 입력해 주세요. (Input field)
- 검색 button

Buttons:

- WRITE
- MAINPAGE

Table of Posts (Listed in a red box):

BOARDNUM	TITLE	WRITER	BOARDDATE	LIKE	PRICE	VIEWS
13	카메라팝니다.	아이언맨짱	2024-02-16	0	500,000원	0
12	카메라팝니다.	아이언맨짱	2024-02-15	0	500,000원	0
11	카메라팝니다.	아이언맨짱	2024-02-15	0	500,000원	0
10	카메라팝니다.	아이언맨짱	2024-02-15	0	500,000원	0
9	카메라팝니다.	아이언맨짱	2024-02-15	0	500,000원	1

Pagination:

- 1
- 2
- 3

Controller로부터 전달받은
게시글 데이터들을
스크립트를 통해 VIEW에 출력

게시글 상세 보기 (1/2)

BoardDAO.java

```
// 게시글 상세보기 - 판매글, 리뷰 게시판. 전미지 // 최종 쿼리 : 게시글의 정보를 받아오며 작성자의 닉네임과 아이디를 받아옴
private static final String SELECTONE = "SELECT FINAL_DATA.*, MEMBER.NICKNAME, MEMBER.ID "
+ "FROM ( " // 3번째 서브쿼리 : 게시글의 정보를 받아오는데 좋아요 수의 값이 'null'일 경우 0으로 초기화
+ "    SELECT BOARD_DATA.* COALESCE(RECOMMEND_DATA.RECOMMENDCNT, 0) AS RECOMMENDCNT "
+ "    FROM ( " // 1번째 서브쿼리 : 요청하는 게시글 번호에 해당하는 게시글의 정보들을 받아옴
+ "        SELECT BOARDNUM, ID, CATEGORY, TITLE, CONTENTS, TO_CHAR(BOARDDATE, 'YYYY-MM-DD') AS BOARDDATE,"
+ "        IMAGE, PRICE, PRODUCTCATEGORY, PRODUCTNAME, COMPANY, STATE, VIEWCOUNT "
+ "        FROM BOARD "
+ "        WHERE BOARDNUM = ?"
+ "    ) BOARD_DATA "
+ "    LEFT JOIN ( " // 2번째 서브쿼리 : 게시글의 좋아요 수를 합산
+ "        SELECT BOARDNUM, COUNT(BOARDNUM) AS RECOMMENDCNT "
+ "        FROM RECOMMEND "
+ "        GROUP BY BOARDNUM "
+ "    ) RECOMMEND_DATA "
+ "    ON BOARD_DATA.BOARDNUM = RECOMMEND_DATA.BOARDNUM "
+ ") FINAL_DATA "
+ "JOIN MEMBER ON MEMBER.ID = FINAL_DATA.ID"; // 회원 테이블과 게시글 테이블인 FINAL_DATA를 JOIN
// BOARD 테이블과 MEMBER 테이블의 회원 ID를 기준으로 INNER JOIN해 게시글의 작성자 정보를 가져온다.
// 조인한 게시판 테이블 : 회원 테이블, 좋아요 테이블
// 사용한 컬럼(보여줄 목록) : 게시글 번호, 글제목, 작성자 아이디(회원 테이블), 작성자 닉네임(회원 테이블),
// 판매상태, 조회수, 좋아요 넘버(좋아요 테이블),
// 글내용, 이미지, 가격, 상품 종류, 상품명, 상품 제조사,
// 작성일, 카운트 함수 사용(좋아요수-좋아요 테이블 / 좋아요 값이 있을 때만 보여짐)
// 검색 조건 : 게시글 번호 검색 (선택한 게시글 번호의 게시글만 출력 / BOARDNUM 값이 없으면 오류 발생 주의)
```

게시글의 좋아요 수가 null일 경우
COALESE 함수를 사용해
값을 0으로 변경 후 출력

게시글 상세 보기 (2/2)

boardSeleteOne.jsp

```
<!-- 리뷰할 카메라 상품 정보 -->
<div id="cameraReviewData" style="font-weight: bold;"> <!-- 카메라 상품 정보 div -->
    <!-- 상품 가격 -->
    <pre> Price : ${boardData.price}원</pre>
    <!-- 상품 종류 -->
    <pre> ProductCategory : ${boardData.productCategory} </pre>
    <!-- 상품 이름 -->
    <pre> ProductName : ${boardData.productName} </pre>
    <!-- 상품 제조사 -->
    <pre> Company : ${boardData.company} </pre>
</div> <!-- 카메라 상품 정보 div -->

<!-- 게시글 내용 -->
<div> <!-- 게시글 내용 div -->
    <pre style="font-weight: bold;"> Content :</pre>
    <!-- 첨부 이미지 -->
    <div id="imageContainer">
         <br>
    </div>
    <pre class="boardContents"> ${boardData.contents} </pre>
</div> <!-- 게시글 내용 div -->
```

Controller로부터 전달받은 게시글 데이터들을
EL을 사용해 출력

CAMERA REVIEW BOARD

Title 카메라팝니다.

Views 1 | Recommend 0

Date 2024-02-15 | Writer 아이언맨짱

Price | 500000원

ProductCategory | DSLR

ProductName | 캐논

Company | 캐논

Content



너무 좋아요

LIKE

UPDATE DELETE REPORT

게시글 삭제

REVIEW.sql & RECOMMEND.sql

```
-- 댓글 테이블
CREATE TABLE REVIEW(
    REVIEWNUM INT PRIMARY KEY,
    BOARDNUM INT,
    ID VARCHAR(100),
    REVIEWDATE DATE DEFAULT SYSDATE,
    REVIEWCONTENTS VARCHAR(500) NOT NULL,
    FOREIGN KEY (BOARDNUM)
        REFERENCES BOARD(BOARDNUM) ON DELETE CASCADE
);

-- 추천 테이블
CREATE TABLE RECOMMEND(
    RECOMMENDNUM INT PRIMARY KEY,
    BOARDNUM INT,
    ID VARCHAR(100),
    FOREIGN KEY (BOARDNUM)
        REFERENCES BOARD(BOARDNUM) ON DELETE CASCADE
);
```

BoardDAO.java

```
// 게시글 삭제. 전미지
// delete 메서드의 반환 타입은 boolean 타입으로 게시글 삭제 여부를 나타냄
public boolean delete(BoardDTO boardDTO) {
    conn = JDBCUtil.connect(); // JDBCUtil을 사용하여 데이터베이스 연결
    try {
        pstmt = conn.prepareStatement(DELETE); // DELETE 쿼리를 실행하기 위한 PreparedStatement 생성
        pstmt.setInt(1, boardDTO.getBoardNum()); // PreparedStatement에 삭제할 게시글의 번호를 설정
        int rs = pstmt.executeUpdate(); // 게시글이 삭제에 성공하면 1을, 그렇지 않으면 0을 반환
        if (rs <= 0) { // 게시글 삭제에 실패할 경우
            return false; // 'false'를 반환하여 메서드를 종료
        }
    } catch (SQLException e) { // SQLException 예외 발생 시
        e.printStackTrace(); // 콘솔에 예외 내용을 출력하고 'false'로 반환
        return false;
    } finally {
        JDBCUtil.disconnect(pstmt, conn); // 데이터베이스 연결 및 PreparedStatement를 닫음
    }
    return true; // 게시글 삭제를 성공하면 'true'로 반환
}
```



게시글 삭제 시 댓글과 좋아요 테이블도 같이 삭제되게 하기 위해
댓글과 좋아요 테이블 생성 시 ON DELETE CASCADE를 사용

VIEW 모듈화 (1/2)

카피라이트와 같이 중복되는 내용은 커스텀 태그화

```
<!-- 저작권 및 회사 정보를 담은 푸터 섹션 -->
<stone:copyright /> <!-- 카피라이트 태그 -->
```

회사소개 | 광고안내 | 이용약관 | 개인정보처리방침
ADDRESS : 146, TEHERAN-RO, GANGNAM-GU, SEOUL, REPUBLIC OF KOREA | TEL : 010-3975-7635 | MAIL : PLZJUN0731@NAVER.COM
COPYRIGHT © 2023 - 2024 INFINITY STONE . ALL RIGHTS RESERVED.

Controller로부터 전달받은 게시글 데이터들을 EL을 사용해 출력

```
<!-- 리뷰할 카메라 상품 정보 -->
<div id="cameraReviewData" style="font-weight: bold;"> <!-- 카메라 상품 정보 div -->
    <!-- 상품 가격 -->
    <pre> Price : ${boardData.price}원</pre>
    <!-- 상품 종류 -->
    <pre> ProductCategory : ${boardData.productCategory} </pre>
    <!-- 상품 이름 -->
    <pre> ProductName : ${boardData.productName} </pre>
    <!-- 상품 제조사 -->
    <pre> Company : ${boardData.company} </pre>
</div> <!-- 카메라 상품 정보 div -->

<!-- 게시글 내용 -->
<div> <!-- 게시글 내용 div -->
    <pre style="font-weight: bold;"> Content :</pre>
    <!-- 첨부 이미지 -->
<div id="imageContainer">
     <br>
</div>
    <pre class="boardContents"> ${boardData.contents} </pre>
</div> <!-- 게시글 내용 div -->
```

Price | 500000원

ProductCategory | DSLR

ProductName | 캐논

Company | 캐논

Content



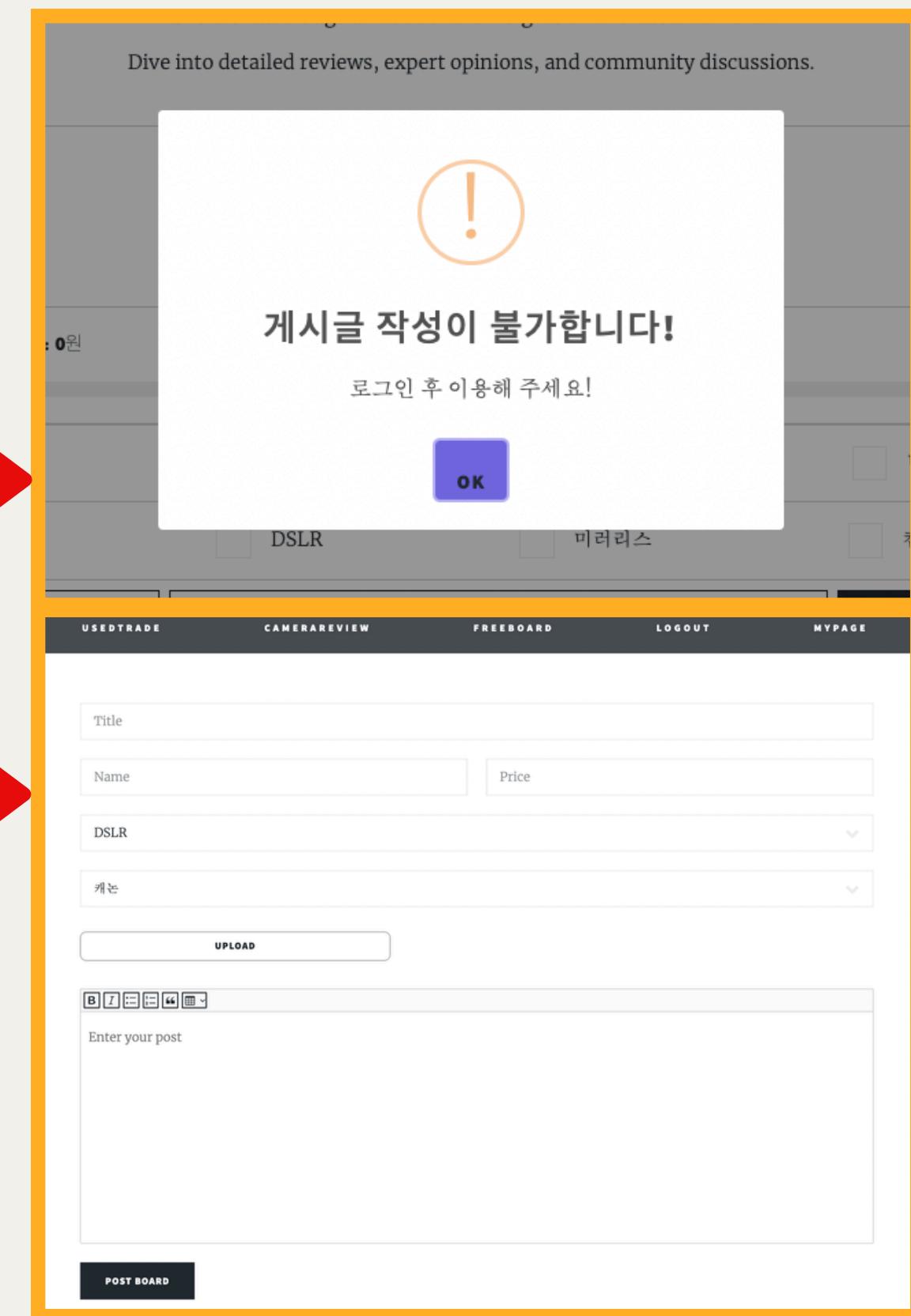
너무 좋아요

VIEW 모듈화 (2/2)

boardDeleteAll.jsp

```
<script> // writeBtn()이 클릭될 때 실행되는 앤수 ()
function writeBtn() {
    var member = "${member}"; // 사용자 정보를 가져오기 위해 JSP에서 전달된 ${member} 값을 가져옴
    if (member == "") { // 만약 회원 정보가 비어있다면 (로그인하지 않은 상태)
        Swal.fire({ // SweetAlert2를 사용하여 경고창을 표시
            title: '게시글 작성이 불가합니다!', // Alert 제목
            text: '로그인 후 이용해 주세요!', // Alert 내용
            icon: 'warning' // Alert 타입
        }).then((result) => { // 경고창이 닫힐 때 확인 버튼이 눌리지 확인
            if (result.isConfirmed) { // 확인 버튼이 눌렸다면 로그인 페이지로 이동!
                location.href = '/chalKag/loginPage.do';
            }
        });
    } else { // 회원 정보가 있다면(로그인 되어있다면) 글 작성 페이지로 이동
        location.href = '/chalKag/cameraReviewWritePage.do';
    }
}
</script>
```

버튼은 모달로 스크립트 처리하여 사용자의 동작에 따라
VIEW에서 유효성 검사 및 페이지 이동 처리



마이페이지, 페이징 처리 및 필터 검색

- 마이페이지
- 페이징 처리
- 필터 검색

마이페이지 JS

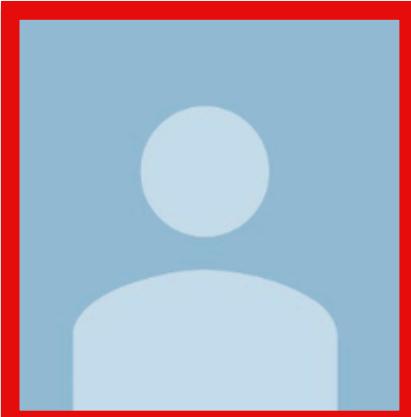
profileImage.js

사용자가 선택한 프로필 이미지 사이즈 조정

이미지의 가로 세로를 비교해
짧은 길이를 기준으로
1:1 비율로 조정

```
var width = img.width; // 이미지의 너비를 가져옴
var height = img.height; // 이미지의 높이를 가져옴

var size = Math.min(width, height); // 너비와 높이 중 작은 값을 size 변수에 저장
var xOffset = (width - size) / 2; // 이미지를 중앙에 놓을 때 필요한 x축 오프셋을 계산
var yOffset = (height - size) / 2; // 이미지를 중앙에 놓을 때 필요한 y축 오프셋을 계산
```



MYPAGE

ID : rornfl@naver.com

NAME : 정석진

NICKNAME : JSJ1689

PH : 010-****-1689

```
ctx.drawImage(img, xOffset, yOffset, size, size, 0, 0, size, size); // 이미지를 캔버스에 그림

var resizedCanvas = document.createElement('canvas'); // 새로운 캔버스 요소를 만들어 변수에 저장
var resizedCtx = resizedCanvas.getContext('2d'); // 새로운 캔버스의 2D 렌더링 컨텍스트를 가져 옴
resizedCanvas.width = maxWidth; // 새로운 캔버스의 너비를 최대 너비로 설정
resizedCanvas.height = maxHeight; // 새로운 캔버스의 높이를 최대 높이로 설정

resizedCtx.drawImage(canvas, 0, 0, size, size, 0, 0, maxWidth, maxHeight); // 원본 캔버스에서 이미지를 그림
```

비율을 유지하며 마이페이지에 보여질 사이즈로 조정

마이페이지 서블릿

ProfileUploadAction.java 프로필 이미지 변경 비동기

```
// 프로필 이미지 업로드 AJAX
$.ajax({
    type: "POST",
    // 폼 데이터가 서버로 제출될 때 해당 데이터를 인코딩하는 방법을 명시하는 속성
    // 파일 전송 같은 이진 데이터를 포함하는 경우에 이 방식을 사용
    enctype: 'multipart/form-data',
    url: "profileUpload.do",
    data: formData,
```

cos.jar 라이브러리를 사용한 이미지 파일 처리

```
// 이 클래스는 cos.jar 라이브러리에 포함되어 있으며,
// 클라이언트로부터 전송받은 데이터가 여러 부분으로 구성되어 있을 때
// 이를 처리하는 데 사용
// 이렇게 MultipartRequest 객체를 생성하면,
// 사용자가 업로드한 파일을 서버에 저장하고, 이 파일에 대한 정보를 처리 가능
MultipartRequest multipartRequest = new MultipartRequest(
    new CustomFileRenamePolicy(id));
```

DB 저장시 파일명 유저 ID로 저장

```
//rename Override를 위한 클래스
class CustomFileRenamePolicy extends DefaultFileRenamePolicy {

    private String newFileName;

    public CustomFileRenamePolicy(String newFileName) {
        this.newFileName = newFileName;
    }

    // 저장하는 파일명 재정의 하는 메서드
    @Override
    public File rename(File file) {

        // 업로드한파일 확장자를 extension에 대입
        String extension = (extractExtension(file.getName()));

        // newName에 newFileName(memberID값) + ".확장자" 대입
        String newName = newFileName + extension;

        // 새로운 파일객체 생성 후 리턴
        File newFile = new File(file.getParent(), newName);

        return newFile;
    }
}
```

페이지 처리 JS (1/3)

pagination.js

```

    currentPage = $(this).data("page"); // 클릭된 페이지 번호를 가져와 currentPage에 저장
    pagination(currentPage); // 선택한 페이지의 게시글 출력

    pagination = function(currentPage) {
        var displayDatas = jsonBoardDatas; // 게시판 이동시 사용되는 게시글 데이터들
        if (isFiltered) {
            displayDatas = jsonFilteredBoardDatas; // 필터링된 데이터들이 있으면 필터링된 데이터들 사용
        }

        var pageDataSize = 5; // 페이지당 게시글 개수
        var totalsize = displayDatas.length; // 전체 게시글 개수
        totalPages = Math.ceil(totalsize / pageDataSize); // 전체 페이지 수 계산

        var startIndex = (currentPage - 1) * pageDataSize; // 시작 인덱스 계산
        var endIndex = Math.min(startIndex + pageDataSize, totalsize); // 끝 인덱스 계산
        var currentPageDatas = displayDatas.slice(startIndex, endIndex); // 현재 페이지에 나타낼 게시글들

        isplayBoardData(currentPageDatas); // 게시글 출력 함수 호출
        isplayPagination(currentPage); // 페이지네이션 표시 함수 호출
    }
}

```

BOARDNUM	TITLE
1	캐논의 보급형 DSLR? 프로용 DS
2	캐논 RF24-105mm F2.8 L IS USM
3	[첫인상] 니콘 Z fc
4	소니 SonyE16mmF2.8Review
5	니콘D70vs캐논EOS300D2편

서버로부터 받아온
게시글 데이터들을
페이지당 게시글 수에 맞게 출력

전체 게시글의 수를 고려하여
페이지 버튼을 생성

페이지 처리 JS (2/3)

pagination.js

```
function displayBoardData(currentPageDatas) {
    // console.log("[로그]" + currentPageDatas);
    var tbody = document.querySelector('.alt tbody');
    var thead = document.querySelector('.alt thead');
    var thCnt = thead.getElementsByTagName('th');
    var maxColspan = thCnt.length; // 최대 컬럼 수
    tbody.innerHTML = ''; // tbody 초기화
    if (currentPageDatas.length === 0) {
        tbody.innerHTML = '<tr><td colspan=' + maxColspan + '" align="center">등록된 게시글이 없습니다.</td></tr>';
    } else {
        currentPageDatas.forEach(function(boardData) {
            var row = document.createElement('tr'); // 새로운 행 요소 생성
            // 데이터에 따라 행 채우기
            if (id === boardData.id) {
                // console.log('[로그] myBoard & memberBoard');
                // 내가 작성한 게시글 및 유저가 작성한 게시글인 경우
                row.innerHTML =
                    '<td>' + boardData.rownum + '</td>' +
                    '<td>' + boardData.title + '</td>' +
                    '<td>' + boardData.nickname + '</td>' +
                    '<td>' + boardData.boardDate + '</td>' +
                    '<td>' + boardData.recommendCNT + '</td>' +
                    '<td>' + boardData.viewCount + '</td>';
            }
            tbody.appendChild(row);
        });
    }
}
```

기존에 테이블에 있던
게시글 데이터들 클리어

받아온 데이터들을 순회하며
게시글을 화면에 동적으로 출력

<tbody>
 <!-- 게시글이 등적으로 채워질 테이블 공간 -->
</tbody>

페이지 처리 JS (3/3)

pagination.js

```

function displayPagination(currentPage) {
    var paginationContainer = $("#paginationContainer"); // 페이지네이션 컨테이너 요소
    paginationContainer.empty(); // 컨테이너 초기화

    var pageSize = 10; // 한 페이지 그룹에 표시할 페이지 수
    var currentGroup = Math.floor((currentPage - 1) / pageSize); // 현재 페이지 그룹
    var startPage = currentGroup * pageSize + 1; // 시작 페이지
    var endPage = Math.min((currentGroup + 1) * pageSize, totalPages); // 끝 페이지

    // 이전 페이지 그룹으로 이동하는 버튼 추가
    if (startPage > 1) {
        var prevGroupPage = startPage - 1;
        var prevGroupLink = "<a href='#' class='page' data-page='" + prevGroupPage + "'";
        paginationContainer.append(prevGroupLink);
    }

    // 페이지 버튼 추가
    for (var i = startPage; i <= endPage; i++) {
        var pageLinkClass = (i === currentPage) ? "page active" : "page";
        var pageLink = "<a href='#' class='" + pageLinkClass + "' data-page='" + i + "'";
        paginationContainer.append(pageLink);
    }
}

```



한 페이지 그룹에
표시할 페이지 수를 설정
시작 페이지와 끝 페이지를 계산

게시글을 나눠서 출력하기 위해
필요한 페이지 버튼이 동적으로 생성

```

<div id="paginationContainer" class="pagination"
      <!-- 페이지 버튼이 동적으로 생성될 공간 -->
</div>

```

필터 검색 화면

최저 금액 : 0원 최고 금액 : 250000원

제조사 캐논 소니 니콘

카메라 기종 DSLR 미러리스 컴팩트

판매 상태 판매중 판매완료

제목 미개봉 검색

제목
내용
작성자
제목 + 내용

WRITE MAINPAGE

BOARDNUM	TITLE	PRICE ↓	STATE	WRITER	BOARDDATE	VIEWS
1	[미개봉]소니E16mmF2.8 판매합니...	237,600원	판매중	티모는오소리	2024-02-20	352
2	캐논 RF15-30판매(미개봉)	200,000원	판매중	월레스와그로밋	2024-02-20	76
3	캐논 50mm 1.2미개봉 팔아요	120,000원	판매중	별사탕후루	2024-02-20	242

1

필터 검색 JS (1/2)

filterSearch.js

```
// 검색 버튼 클릭 이벤트
$("#searchButton").on("click", function() {
    updateVariables();
    filterSearch();
});

// 가격 범위 검색 입력값 변경 이벤트
$("#minPrice, #maxPrice").on("mouseup", function() {
    updateVariables();
    filterSearch();
});

// 체크박스 변경 이벤트
$('input[type=checkbox]').change(function() {
    updateVariables();
    filterSearch();
});
```

사용자가 필터 조건을 선택 시
필터 조건 전체의 상태를 확인해서 업데이트

```
// 블러핑 및 캠핑에 사용되는 변수를 업데이트
function updateVariables() {
    minPrice = $("#minPrice").val(); // 최소 가격
    maxPrice = $("#maxPrice").val(); // 최대 가격
    searchField = $("#searchField").val(); // 검색 기준
    searchInput = $("#searchInput").val(); // 검색 입력값

    // 선택된 체크박스들의 값을 배열로 저장
    // 제조사
    selectedCompanies = $('input[type=checkbox][name=comp]
        return this.value;
    }).get();
```

필터 검색 JS (2/2)

filterSearch.js

```
// 검색 버튼 클릭 이벤트
$("#searchButton").on("click", function() {
    updateVariables();
    filterSearch();
});

// 가격 범위 검색 입력값 변경 이벤트
$("#minPrice, #maxPrice").on("mouseup", function() {
    updateVariables();
    filterSearch(); // 이 줄은 빨간색 박스에 포함되어 있다.
});

// 체크박스 변경 이벤트
$('input[type=checkbox]').change(function() {
    updateVariables();
    filterSearch();
});
```

사용자가 선택한 전체 필터 조건을
서버에게 비동기방식으로 요청

```
.ajax({
    type: "GET",
    url: "filterSearch.do",
    data: { // 필터링 가능한 항목들
        'minPrice': minPrice, // 최대 가격
        'maxPrice': maxPrice, // 최소 가격
        'company': selectedCompanies, // 제조사
        'productcategory': selectedProductCategories, // 카메라 기종
        'state': selectedstates, // 판매 상태
        'category': category, // 게시판에 맞는 게시글 출력을 위한 변수
        'searchField': searchField, // 검색 기준
        'searchInput': searchInput, // 검색 값
        'id': id, // 내가 작성한 게시글 출력을 위한 변수
        // 정렬 기준 및 정렬 방향
        'jsonOrderColumnDirection': JSON.stringify(selectedOrderDirection)
    },
    traditional: true,
    dataType: 'json',
    success: function(jsonFilterBoardDatas) {
        if (jsonFilterBoardDatas != null) { // filterDatas가 존재하는 경우
            window.jsonFilteredBoardDatas = jsonFilterBoardDatas; // 서버에서
            console.log(jsonFilteredBoardDatas); // 데이터 확인
            isFiltered = true; // 데이터가 존재하므로 isFiltered를 true로 설정
            pagination(1);
        }
    }
});
```

필터 검색 서블릿

FilterSearchAction.java

```

if (request.getParameter("minPrice") != null && request.getParameter("maxPrice") != null) {
    int minPrice = Integer.parseInt(request.getParameter("minPrice"));
    int maxPrice = Integer.parseInt(request.getParameter("maxPrice"));
    // System.out.println("[로그] 받아온 가격 파라미터 값 :" + minPrice + ", " + maxPrice);
    searchDTO.setMinPrice(minPrice);
    searchDTO.setMaxPrice(maxPrice);
}

if (company != null) {
    for (int i = 0; i < company.length; i++) {
        cList.add(company[i]);
        System.out.println("[로그] 받아온 회사 파라미터 값 : " + company[i]);
    }
    searchDTO.setCompanyList(cList);
}

if (productcategory != null) {
    for (int i = 0; i < productcategory.length; i++) {
        pList.add(productcategory[i]);
        System.out.println("[로그] 받아온 제품 카테고리 파라미터 값 : " + productcategory[i]);
    }
    searchDTO.setProductcategoryList(pList);
}

```

클라이언트로부터 받은 검색조건을 확인해서
DTO에 세팅해주기 위해 알맞는 데이터 타입으로 변환 후
DAO의 `selectAll` 메서드를 사용해 쿼리를 이용한 필터검색 진행

```

// 검색 DAO를 통해 필터링된 게시글을 가져옴
ArrayList<BoardDTO> filteredBoardDatas = searchDAO.selectAll(searchDTO);
// System.out.println("[로그] 필터된 게시글 : " + filteredBoardDatas);

// 검색 결과를 JSON 형식으로 응답
if (filteredBoardDatas != null) {
    Gson gson = new Gson();
    String jsonFilterBoardDatasStr = gson.toJson(filteredBoardDatas);
    response.setContentType("application/json");
    response.setCharacterEncoding("UTF-8");
    response.getWriter().write(jsonFilterBoardDatasStr);
}

```

필터 검색 Model

FilterSearchDAO.java

```
// 검색 및 필터링을 위한 SQL 구문을 저장하는 변수들
String PRTCESOI = "";
String COMPANYSQL = "";
String PRODUCTCATEGORYSQL = "";
String STATESQL = "";
String USERSEARCHSQL = "";
String ORDERSQL = "";
String CATEGORYSQL = "";
String SQL_SELECTALL = "";
```

SQL_SELECTALL

```
"SELECT SORT_DATA.* , MEMBER.NICKNAME, MEMBER.ID " +
"FROM ( " +
"    SELECT FILTER_DATA.* , COALESCE(RECOMMEND_COUNT.RECOMMENDCNT, 0) AS RECOMMENDCNT " +
"    FROM ( " +
"        SELECT ROWNUM, ROWNUM_DATA.* " +
"        FROM ( " +
"            SELECT BOARDNUM, ID, CATEGORY, TITLE, CONTENTS, " +
"                   TO_CHAR(BOARDDATE, 'YYYY-MM-DD') AS BOARDDATE, " +
"                   PRICE, PRODUCTCATEGORY, COMPANY, STATE, VIEWCOUNT " +
"            FROM BOARD " +
"            WHERE CATEGORY = '" + CATEGORYSQL + "' " +
"                  " + PRTCESOI + " " +
"                  " + COMPANYSQL + " " +
"                  " + PRODUCTCATEGORYSQL + " " +
"                  " + STATESQL + " " +
"        ) ROWNUM_DATA " +
"        ORDER BY ROWNUM " +
"    ) FILTER_DATA " +
"    ORDER BY RECOMMENDCNT DESC, ROWNUM " +
" ) SORT_DATA " +
"    ORDER BY ROWNUM "
```

```
StringBuilder bCompanySb = new StringBuilder();
ArrayList<String> companyDatas = searchDTO.getCompanyList();
for (int i = 0; i < searchDTO.getCompanyList().size(); i++) {
    // 회사 목록의 각 요소에 (')를 붙여 StringBuilder에 추가
    bCompanySb.append("'" + companyDatas.get(i) + "'");
    // 마지막 리스트가 아닐 경우 각 StringBuilder에 (,) 추가
    if (i + 1 < searchDTO.getCompanyList().size()) {
        bCompanySb.append(",");
    }
}
// 특정 회사가 포함된 게시글 검색하는데 사용
COMPANYSQL = "AND COMPANY IN (" + bCompanySb.toString() + ")";
```

입력받은 필터조건에 맞게
알맞는 쿼리문을 생성 후

SQL_SELECTALL에 대입하여
원하는 게시글 검색

댓글 가능

- 댓글 VIEW 모듈화
- 댓글 작성
- 댓글 삭제
- 댓글 수정

댓글 VIEW 모듈화

boardSelectOne.java

```
%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"
import="model.board.*,model.review.*,java.util.ArrayList"%>
%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix=
%@ taglib tagdir="/WEB-INF/tags" prefix="stone"%>

<!-- 댓글 -->
<stone:review />
</section>
</div>
<stone:copyright />
```

카테고리마다 게시글 자세히 보기 페이지를 분류

댓글은 게시글 자세히 보기 페이지에 포함

모듈화를 함으로써 각 카테고리의
게시글 자세히 보기 페이지에 코드를
반복 작성할 필요가 없기 때문에 유지보수에 용이

review.tag

```
<form id="reviewWriteForm" method="post" action="/chalKag/reviewWrite.do">
  <div class="fields">
    <div class="field">
      <label for="message">Write Contents</label>
      <c:if test="${member==null}">
        <textarea disabled name="reviewContents" id="reviewContents" rows="3">로그인 후 댓글을 작성해주세요</textarea>
      </c:if>
      <c:if test="${member!=null}">
        <textarea name="reviewContents" id="reviewContents" rows="3"></textarea>
      </c:if>
    </div>
  </div>
  <input type="hidden" name="boardNum" id="boardNum" value="${boardData.boardNum}">
  <input type="hidden" name="category" id="category" value="${boardData.category}">
  <ul class="actions">
    <c:if test="${member!=null}">
      <li><input type="submit" value="Leaving a comment" /></li>
    </c:if>
  </ul>
  <br>
</form>
</section>

<hr />

<c:if test="${fn:length(reviewDatas) <= 0}">
  <h4>댓글이 없습니다. 가장 먼저 댓글을 남겨보세요!</h4>
</c:if>
```

댓글 작성 (1/2)

review.tag

```
<c:if test="${member==null}">  
<textarea disabled name="reviewContents" id="reviewContents" rows="3">로그인 후 댓글을 작성해주세요!</textarea>  
</c:if>  
<c:if test="${member!=null}">  
<textarea name="reviewContents" id="reviewContents" rows="3"></text  
</c:if>
```

```
<!-- 댓글 작성 시에 필요한 게시글 번호와 카테고리를 전달-->  
<input type="hidden" name="boardNum" id="boardNum"  
value="${boardData.boardNum}">  
<input type="hidden" name="category" id="category"  
value="${boardData.category}">  
<ul class="actions">  
    <c:if test="${member!=null}">  
        <li><input type="submit" value="Leaving a comment" />
```

WRITE CONTENTS

로그인 후 댓글을 작성해주세요!

WRITE CONTENTS

게시글 번호와 댓글 내용
그리고 해당 게시글의 카테고리를
댓글 작성 Controller로 전송

LEAVING A COMMENT

댓글 작성 (2/2)

ReviewDAO.java

```
private static final String INSERT = "INSERT INTO REVIEW (REVIEWNUM, BOARDNUM, ID, REVIEWCONTENTS  
+ "VALUES ((SELECT NVL(MAX(REVIEWNUM), 0) + 1 FROM REVIEW), ?, ?, ?);"  
  
// 댓글 작성  
public boolean insert(ReviewDTO reviewDTO) {  
  
    conn = JDBCUtil.connect();  
    try {  
        pstmt = conn.prepareStatement(INSERT);  
        System.out.println("[로그] insert 접근");  
        pstmt.setInt(1, reviewDTO.getBoardNum());  
        pstmt.setString(2, reviewDTO.getId());  
        pstmt.setString(3, reviewDTO.getReviewContents());  
        int rs = pstmt.executeUpdate();  
        if (rs <= 0) {  
            return false;  
        }  
    } catch (SQLException e) {  
        e.printStackTrace();  
        return false;  
    } finally {  
        JDBCUtil.disconnect(pstmt, conn);  
    }  
    return true;  
}
```

ReviewWriteAction.java

```
reviewDTO.setBoardNum(Integer.parseInt(request.getParameter("boardNum")));  
reviewDTO.setId((String) session.getAttribute("member"));  
reviewDTO.setReviewContents(request.getParameter("reviewContents"));  
  
boolean flag = reviewDAO.insert(reviewDTO);  
  
// 댓글 작성 실행 후 결과를 boolean 타입으로 받아서 로그 확인  
if (flag) {  
    System.out.println("[로그] 댓글 작성 성공");  
} else {  
    System.out.println("[로그] 댓글 작성 실패");  
}  
  
boardDTO.setBoardNum(Integer.parseInt(request.getParameter("boardNum")));  
boardDTO.setCategory(request.getParameter("category"));  
boardDTO.setUpdatePage("");  
  
boardDAO.selectOne(boardDTO);  
  
System.out.println("[로그] boardNum : " + boardDTO.getBoardNum());  
  
String category = boardDTO.getCategory();  
  
if(category.equals("자유게시판")) {  
    forward.setPath("freeBoardSelectOnePage.do");  
}  
else if(category.equals("판매게시판")) {  
    forward.setPath("sellBoardSelectOnePage.do");  
}  
else if(category.equals("리뷰게시판")) {  
    forward.setPath("cameraReviewBoardSelectOnePage.do");  
}
```

**게시글 모델 사용하여
도착 페이지 지정**

댓글 삭제 (1/2)

review.tag

```
<c:if test="${reviewData.id == member}">
```

JSTL를 사용해서
해당 댓글 작성자의 아이디와 현재 로그인한 사용자의 아이디를 비교

```
<a href="/chalKag/reviewDelete.do?reviewNum=${reviewData.reviewNum}&boardNum=${boardData.boardNum}&category=${boardData.category}">삭제</a><br>${reviewData.reviewDate}</div>
```

두 값이 일치한다면 수정 및 삭제 하이퍼링크를 출력

REVIEW COUNT : 1

김성민

좋은 내용입니다!

수정 삭제

2024-02-19 11:29

REVIEW COUNT : 1

김성민

좋은 내용입니다!

2024-02-19 11:29

댓글 삭제 (2/2)

review.tag

```
<c:if test="${reviewData.id == member}">  
  
(a href="/chalkag/reviewDelete.do?reviewNum=${reviewData.reviewNum}  
&boardNum=${boardData.boardNum}&category=${boardData.category}">삭제<  
(div>${reviewData.reviewDate}</div>
```

ReviewDAO.java

```
// 댓글 삭제  
private static final String DELETE = "DELETE FROM REVIEW WHERE REVIEWNUM = ?";  
  
public boolean delete(ReviewDTO reviewDTO) {  
    conn = JDBCUtil.connect();  
    try {  
        pstmt = conn.prepareStatement(DELETE);  
        pstmt.setInt(1, reviewDTO.getReviewNum());  
        int rs = pstmt.executeUpdate();  
        if (rs <= 0) {  
            return false;  
        }  
    } catch (SQLException e) {  
        e.printStackTrace();  
    } finally {  
        JDBCUtil.disconnect(conn);  
    }  
}
```

ReviewDeleteAction.java

```
reviewDTO.setReviewNum(Integer.parseInt(request.getParameter("reviewNum")));  
boolean flag = reviewDAO.delete(reviewDTO);  
// 로그 작성자 | 김성민  
if(flag) {  
    System.out.println("[로그] 댓글 삭제 성공");  
} else {  
    System.out.println("[로그] 댓글 삭제 실패");  
    forward.setPath("errorPage.do");  
}  
  
boardDTO.setBoardNum(Integer.parseInt(request.getParameter("boardNum")));  
boardDTO.setCategory(request.getParameter("category"));  
boardDTO.setUpdatePage("");  
boardDAO.selectOne(boardDTO);  
  
String category = boardDTO.getCategory();  
  
if(category.equals("자유게시판")) {  
    forward.setPath("freeBoardSelectOnePage.do");  
}  
else if(category.equals("판매게시판")) {  
    forward.setPath("sellBoardSelectOnePage.do");  
}  
else if(category.equals("리뷰게시판")) {  
    forward.setPath("cameraReviewBoardSelectOnePage.do");  
}
```

게시글 모델 사용하여
도착 페이지 지정

댓글 수정 (1/3)

review.tag

```
<a class="updateContents" href="javascript:void(0);"  
    onclick="toggleEdit(${reviewData.reviewNum}, '${reviewData.reviewContents}', event)">수정</a>
```

review.tag 비동기 처리 jQuery 코드

```
// 수정을 클릭한 경우  
if (clickedElement.hasClass('updateContents')) {  
    // '수정'인 경우  
    if (clickedElement.text() === '수정') {  
        // '수정'을 '저장'으로 변경  
        clickedElement.text('저장');  
  
        // 해당 댓글Div에 있는 내용을 divContent에 저장  
        var divContent = $('#reviewContents_' + reviewNum).text();  
  
        // 텍스트 영역 엘리먼트 생성  
        var textarea = $('').val(divContent.trim()).attr('rows', 5);  
  
        // div 엘리먼트의 내용을 텍스트 영역으로 완전히 대체  
        var divElement = $('#reviewContents_' + reviewNum);  
  
        if (divElement.length) {  
            // 선택된 div 요소를 비우고, 그 안에 새로운 textarea 요소를 추가  
            divElement.empty().append(textarea);  
        }  
    } // '수정'이 아닌 경우  
    else {  
        // 텍스트 영역에서 업데이트된 내용 가져오기  
        var updatedContents = $('#reviewContents_' + reviewNum + ' textarea').val();  
  
        // 유효성 검사: 공백이나 내용이 없는 경우 저장하지 않음  
        if (updatedContents.trim() === '') {  
            alert('내용을 입력하세요.');// 또는 다른 사용자 피드백 방식으로 변경  
            return; // 저장 중단  
        }  
    }  
}
```

review.tag 비동기 처리 ajax 코드

```
// AJAX 요청 수행  
$.ajax({  
    url: "reviewUpdateAction.do", // 실제 서버 엔드포인트로 변경  
    type: "POST", // 요청 방식 (GET, POST 등)  
    data: {  
        // update를 하기 위한 요소들을 보냄  
        'reviewNum': reviewNum,  
        'updatedContents': updatedContents  
    },  
    dataType: 'text',  
    success: function(response) {  
        // response 확인  
        // console.log('서버 응답:' + response);  
  
        // 성공적으로 업데이트된 경우  
        // div 엘리먼트의 내용을 업데이트된 내용으로 교체  
        var reviewContentsDiv = $('#reviewContents_' + reviewNum);  
        if (reviewContentsDiv.length) {  
            reviewContentsDiv.text(response);  
            // 현재 db에 저장된 데이터  
            // 서버에서 응답을 줄 때 인자인 Response에 태워서 보내줘야함  
        }  
        // '저장'을 '수정'으로 변경  
        clickedElement.text('수정');  
    },  
    error: function(error) {  
        console.log('AJAX 요청 실패:', error);  
    }  
});
```

댓글 번호와 수정된 내용을
요소로 담아서 보냄

댓글 수정 (2/3)

ReviewUpdateAction.java

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) {
    ReviewDTO reviewDTO = new ReviewDTO();
    ReviewDAO reviewDAO = new ReviewDAO();

    reviewDTO.setReviewNum(Integer.parseInt(request.getParameter("reviewNum")));
    reviewDTO.setReviewContents(request.getParameter("updatedContents"));

    System.out.println("[로그] reviewNum : " + reviewDTO.getReviewNum());
    System.out.println("[로그] reviewContents : " + reviewDTO.getReviewContents());

    boolean flag = reviewDAO.update(reviewDTO);

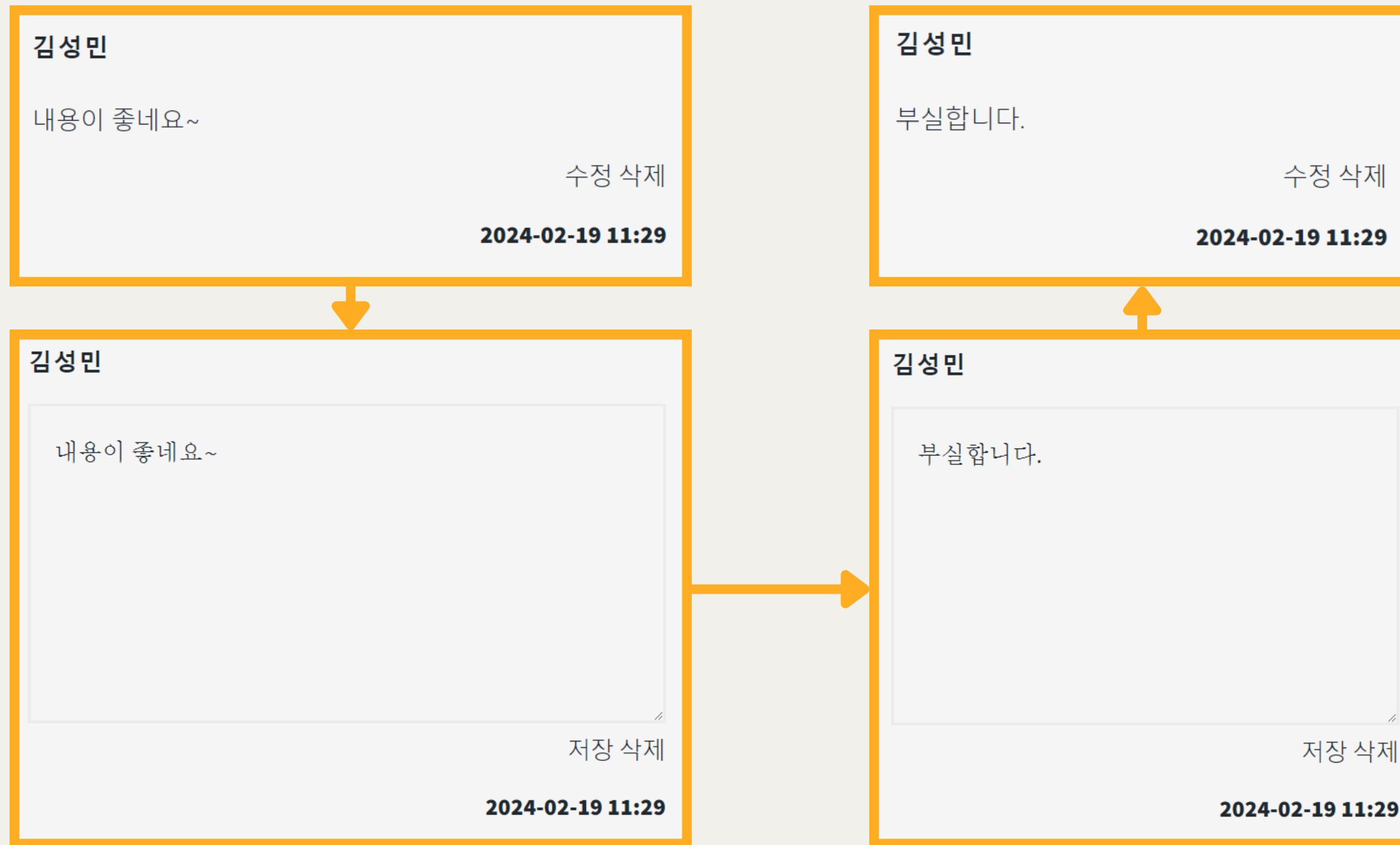
    if(flag) {
        System.out.println("업데이트 완료");
        response.setCharacterEncoding("UTF-8");
        PrintWriter out= response.getWriter();
        out.print(reviewDTO.getReviewContents());
    }
    else {
        System.out.println("업데이트 실패");
    }
}
```

ReviewDAO.java

```
// 댓글 수정
private static final String UPDATE = "UPDATE REVIEW SET REVIEWCONTENTS = ? WHERE REVIEWNUM = ?";

// 댓글 수정
public boolean update(ReviewDTO reviewDTO) {
    conn = JDBCUtil.connect();
    try {
        pstmt = conn.prepareStatement(UPDATE);
        pstmt.setString(1, reviewDTO.getReviewContents());
        pstmt.setInt(2, reviewDTO.getReviewNum());
        int rs = pstmt.executeUpdate();
        if (rs <= 0) {
            return false;
        }
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    } finally {
        JDBCUtil.disconnect(pstmt, conn);
    }
    return true;
}
```

댓글 수정 (3/3)



기능 추가 및 추후 개발 방향

- 대댓글 기능 추가
대댓글 테이블 추가 생성
댓글에 대한 대댓글을 드롭다운으로 출력
- 관리자 모드 추가
회원가입 시 회원 등급에 관리자 추가
등급이 관리자 일 때만 접근 가능한 페이지를 추가할 예정

THANK YOU

감사합니다



정석진

Tel. 010-6622-1689
Email. ddoljin@gmail.com
Blog. <https://varietyofit.tistory.com>
Github. <https://github.com/FullStackJinnnn>