

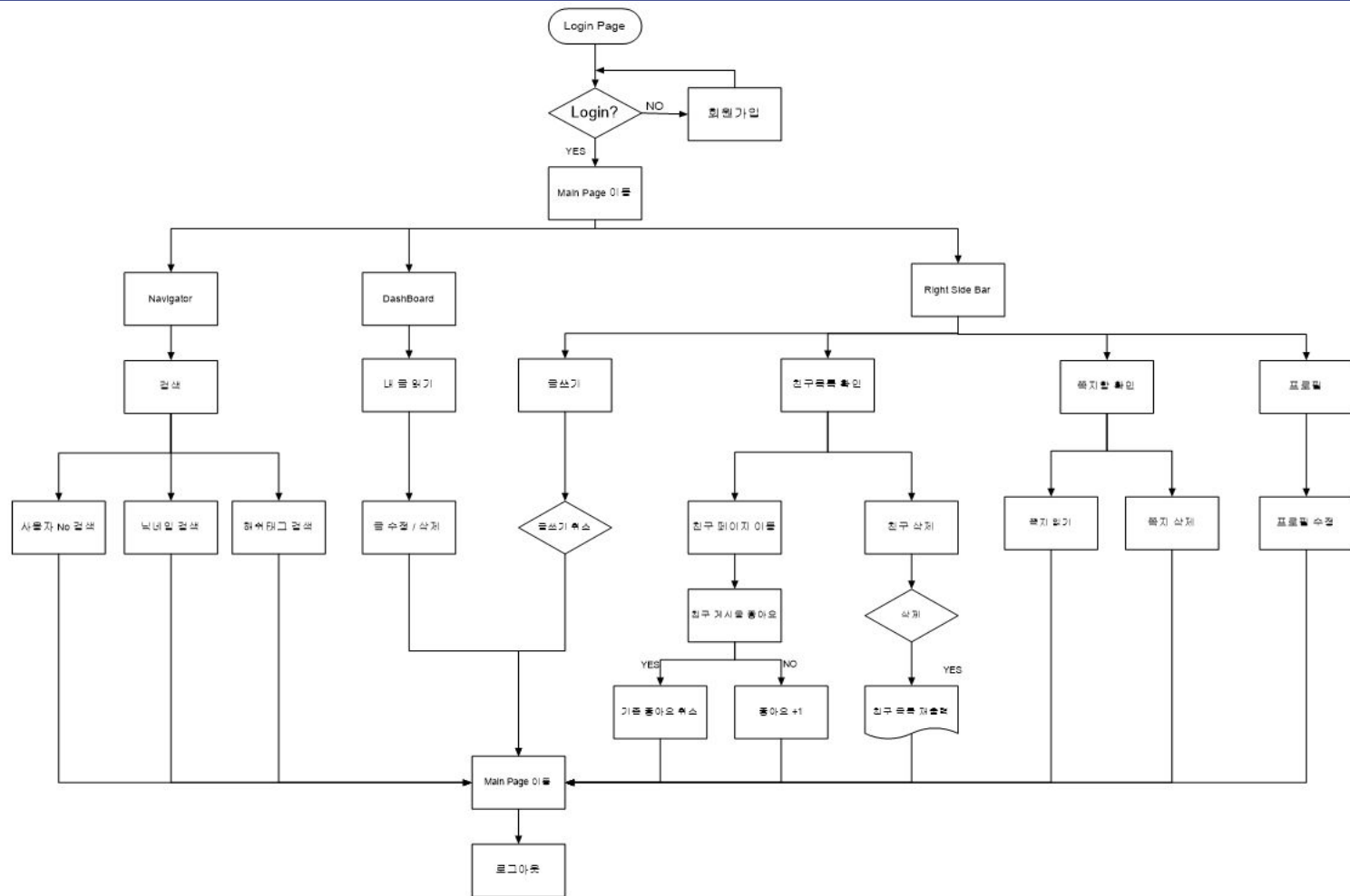
---

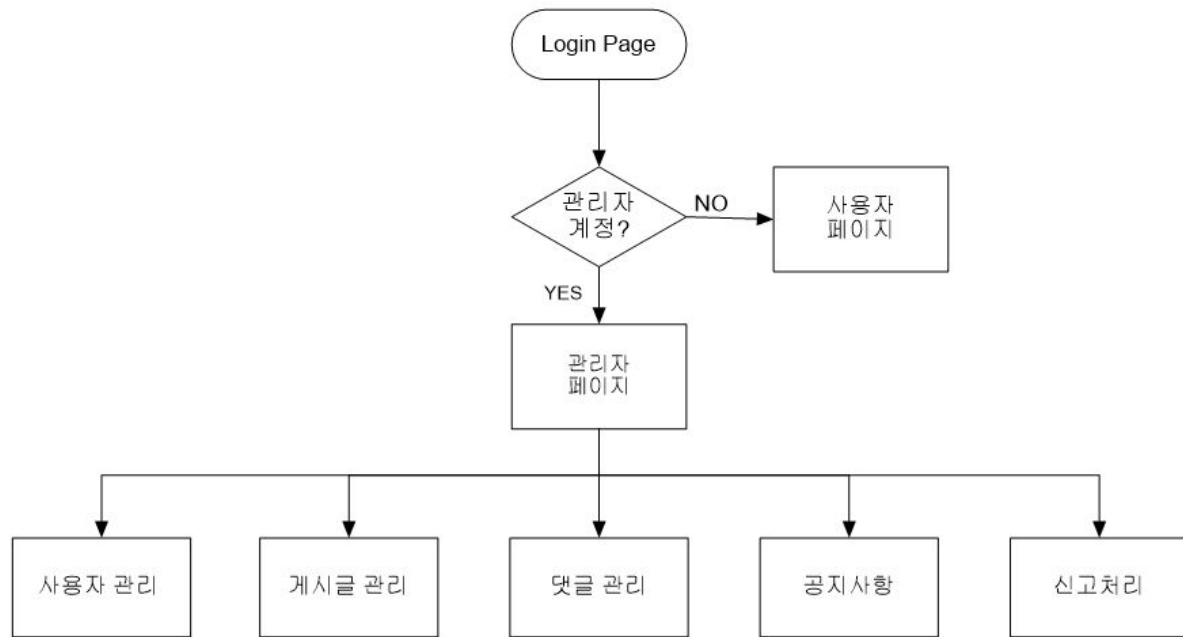
# MVC Pattern을 이용한 SNS 서비스 구현 PROJECT

김솔, 김혜연, 문상혁, 유자영, 이민아, 하지승

---

## 사용자 페이지





## DB 구성도

Table명		컬럼정보						
MEMBER	UserNo(PK)	Userid	UserName	UserPassword	UserProfilePhoto	UserIntro	UserBlock	Manager
	사용자 번호	사용자 아이디	사용자 이름	비밀번호	사용자 프로필 사진	사용자 소개	계정 공개 여부	관리자여부
	Not Null, AI, unique INT	Not Null, unique VARCHAR(45)	Not Null VARCHAR(45)	Not Null VARCHAR(45)	Null VARCHAR(500)	Null VARCHAR(2000)	TRUE Boolean	FALSE boolean
사용자								
BOARD	BoardNo(PK)	UserNo(FK)	BoardContent	BoardPhoto	HashTag	BoardDate		
	게시글 번호	사용자 번호	게시글 내용	게시판 사진	해쉬태그	작성시간		
	Not Null, AI INT	Not Null INT	Not Null VARCHAR(2000)	Null VARCHAR(500)	Null VARCHAR(600)	Not null DATETIME		
	사용자 게시물 데이터							
RELATION	UserNo(FK)	FriendNo(FK)						
	사용자 번호	친구 번호						
	Not Null INT	Not Null INT						
	사용자 친구							
REQUEST	RequestNo(PK)	RequestUserNo(FK)	RequestUserName	ResponseUserNo(FK)	ResponseUserName	Request_Accept	Request_Cancel	
	요청 번호	요청한 사용자 No	요청한 사용자 이름	요청받은 사용자 No	요청받은 사용자 이름	수락 여부	요청 취소 여부	
	Not Null, AI INT	Not Null INT	Not null VARCHAR(45)	Not null INT	Not null VARCHAR(45)	FALSE Boolean	FALSE Boolean	
	친구 요청							
ISGOOD	BoardNo(FK)	UserNo(FK)						
	게시글 번호	사용자 번호						
	Not Null INT	Not Null INT						
	좋아요							

## DB 구성도

MESSAGE	MessageNo(PK)	SendId(FK)	RecvId(FK)	MessageCont	SendDate	CheckMessage
	쪽지 번호	보내는 사용자 ID	받는 사용자ID	메세지 내용	메세지 보낸 시간	메세지 확인여부
	Not Null, AI	Not Null	Not Null	Not Null	Not null	boolean
	INT	VARCHAR(45)	VARCHAR(45)	VARCHAR(2000)	DATETIME	FALSE
쪽지						
COMMENT	CommentNo(PK)	BoardNo(FK)	UserId	CommentCont	CommentDate	
	댓글 번호	게시글 번호	작성자 ID	댓글 내용	댓글 작성 시간	
	Not Null, AI	Not Null	Not Null	Null	Not Null	
	INT	INT	varchar(45)	VARCHAR(2000)	DATETIME	
댓글						
NOTICE	NoticeNo(PK)	NoticeTitle	NoticeWriter	NoticeCont	NoticePhoto	NoticeDate
	공지사항 번호	공지사항 제목	작성자	공지사항 내용	공지사항이미지	공지사항 작성 시간
	Not Null, AI	Not Null	Not Null	Not Null	Null	Null
	INT	VARCHAR(200)	VARCHAR(100)	VARCHAR(2000)	VARCHAR(500)	DATETIME
공지사항						
REPORT(임시)	NoticeNo(FK)	UserNo(FK)				
	공지사항 번호	사용자 번호	<- 임시테이블			
	INT	INT				
	게시글 신고					

## 로그인 페이지

---



The login page features a dark blue background with a central light gray login box. The box contains the B+ BITBOX logo, email and password input fields, a login button, and a link to the sign-up page. The bottom of the page is decorated with a row of colorful circles in blue, red, yellow, and teal.

**B+ BITBOX**

이메일 주소

암호

☒ 아이디 저장하기

**로그인**

[계정이 없으신가요? 회원가입](#)

B+ BITBOX

Search

Q

공지사항

마이페이지

로그아웃

내가 쓴 게시물

#번호: 1, 로그인 userno: 14  
내용: 두찌빠찌포치  
해시태그: #미니언즈 #바나나  
등록일: 2018-11-02  
수정 삭제  
신고하기 신고된갯수 :  
좋아요 : 0 좋아요!  
[댓글달기]  
댓글...댓글을 보자!

#번호: 2, 로그인 userno: 14  
내용: ef  
해시태그:  
등록일: 2018-11-02  
수정 삭제  
신고하기 신고된갯수 :  
좋아요 : 0 좋아요!  
[댓글달기]  
댓글...댓글을 보자!


[1]

글쓰기

친구목록

쪽지함

프로필



ID : mini@gmail.com

이름 : 유자영

내 소개 : 야호

## 회원목록

회원번호	아이디	이름	비밀번호	회원소개
1	manager@gmail.com	김뚜리	1234	ㅇㅇㅇ
4	hyeon21@bit.com	현	123	아오
7	nick@naver.com	닉	1234	하이염
9	bono@gmail.com	하지승	1234	헛소리하지마 임마
11	sol@naver.com	뚜릿	1234	안녕하세요 ^_^~~
12	alreadyna@bit.bit	미나리아꺄	1234	방가워여
13	moon@naver.com	moon	1234	저장 왜 안돼 —
14	mini@gmail.com	유자영	1234	야호



## 업무분담

화면	상세 내용	투입인원	담당자
회원가입/로그인/로그아웃, 프로필	회원가입: id/pw/이름/사진/소개 회원 프로필 수정	1	유자영
게시글 목록(CRUD)	로그인 한 사용자의 글이 최신순으로 표출, 수정 및 삭제 가능	1	이민아
좋아요, 댓글(CRUD)	사용자의 게시글에 대해 좋아요, 댓글 작성 가능	1	김혜연
친구 목록(CRUD) - 관리	로그인 한 사용자의 친구 목록 표출, 친구 프로필 클릭 시 친구 페이지로 이동 친구 추가/삭제 가능	1	문상혁
쪽지 보관함(CRUD)	쪽지목록, 쪽지보내기, 검색 기능, 친구 요청,친구 요청 취소	1	하지승
관리자 페이지	사용자 관리, 글 관리, 댓글 관리, 공지사항 >> 신고기능(3번 신고되면 자동으로 blind 처리)	1	김솔

---

# 구현 목표 - 유저영

## [목표]

수업시간에 배운 스프링 MVC 구조를 이해

MyBatis를 이용한 코드 작성

## [완성]

로그인, 로그아웃, 회원가입,

프로필 정보 수정

## [미완성]

가입시 계정 공개 여부에 따른 서비스 처리

# 로그인, 회원가입 - 유저영

## [구현내용]

로그인 폼 통해 로그인 하기

가입된 계정이 없으면 회원가입 시키기

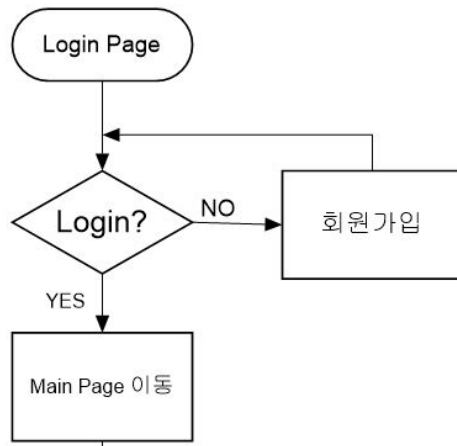
DB : MySQL

TABLE MEMBER 사용

SNS 사이트 특성상

로그인이 돼있어야 서비스를 이용할 수 있기에

첫 화면을 로그인 폼으로 구현



## 코드 리뷰 Member Controller - 유저영

Controller	기능	RequestMapping	RequestMethod
Login	로그인	loginForm	<ul style="list-style-type: none"><li>- <b>Post:</b> userId, userPassword를 이용해 로그인 처리하고 service에서 로그인 정보를 세션에 저장</li></ul>
Reg	회원가입	team/memberReg	<ul style="list-style-type: none"><li>- <b>GET:</b> 회원 가입 폼으로 이동</li><li>- <b>POST:</b> service를 통해 쿼리문을 실행해 회원 정보를 memberInfo 객체와 request 객체에 담아 DB에 저장</li></ul>

## 코드 리뷰 Member Controller - 유저영

Controller	기능	RequestMapping	RequestMethod
profile Edit	사용자 프로필 수정	team/profileEdit	<ul style="list-style-type: none"><li>- <b>Get</b> : userId를 이용해 회원의 정보를 받아서 폼에 뿌려줌</li><li>- <b>Post</b> : 인터페이스 매퍼를 통해 수정된 정보를 DB와 session에 저장, 로그인한 userId를 modelAndView에 담아 게시글 페이지로 보냄</li></ul>

---

# 구현 목표 - 이민아

## [목표]

MVC pattern을 이용하여 게시글 CRUD 구현

## [완성]

로그인한 사용자 게시글 메인에 페이징처리하여  
표출

게시글 작성, 수정, 삭제 완료

## [미완성]

게시글 작성/수정 시 사진 업로드 관련 미구현

> 파일명 저장 시 **boardNo**로 **imgName**을  
만드는데, 게시글 작성 시 **db**에서는 **boardNo**가  
**AI**여서 **boardNo**가 **default** 값(**0**)으로 저장되어  
이미지가 변경되는 이슈가 발생.

> 게시글에 대한 고유값을 가져오는 방법으로  
고민 후 수정할 예정

---

---

# 구현 시 활용한 DB - 이민아

## [Board]

boardno / **userno** / boardcontent / boardphoto /  
hashtag / boarddate

## [Member]

**userno** / userid / ...

## [리뷰]

Member 테이블에서 userno를 가지고

Board 테이블과 inner join 하여 userId로

사용자 게시글에 대한 정보를 가져옴

## 코드 리뷰 Board Controller - 이민아

Controller	기능	RequestMapping	RequestMethod
List	사용자 게시글 리스트	board/boardList	<ul style="list-style-type: none"><li>- <b>GET</b>: page, userid list service를 거쳐 userid에 해당하는 게시글을 보여주며, 사용자가 작성한 글의 전체 개수를 파악하여 <b>paging</b> 처리 한다.</li></ul>
Write	사용자 게시글 작성	board/write	<ul style="list-style-type: none"><li>- <b>GET</b>: 게시글 작성 폼으로 이동</li><li>- <b>POST</b>: userid, board board 객체를 write service를 통해 db에 저장하며, userid를 가지고 list controller를 거쳐 메인으로 이동한다.</li></ul>



## 코드 리뷰 Board Controller - 이민아

Controller	기능	RequestMapping	RequestMethod
Edit	사용자 게시글 수정	board/edit	<ul style="list-style-type: none"><li>- <b>GET:</b> boardno board edit service를 거쳐 boardno에 해당하는 board 객체 데이터를 보내준다.</li><li>- <b>POST:</b> userid 게시글 수정 후 사용자의 메인 페이지로 이동해야 하므로 (list controller) userid를 보내준다.</li></ul>
Delete	사용자 게시글 삭제	board/delete	<ul style="list-style-type: none"><li>- <b>GET:</b> boardno, userid delete service를 거쳐 boardno에 해당하는 게시글을 삭제한다. userid를 전달받아 boardList로 redirect 하여 로그인한 사용자의 게시글 list 표출한다.</li></ul>

# 게시물 좋아요! - 김혜연

## [구현내용]

게시물 좋아요 :

각 게시물 - 사용자별 좋아요 / 취소 기능 구현

DB : MySQL

TABLE ISGOOD

BOARDNO(int 11) / USERNO(int 11)

메인페이지 접속 -> DB에서 좋아요 개수 카운트  
-> 게시글리스트와 함께 표시 -> 좋아요 여부에  
따라 버튼 라벨을 다르게 처리

좋아요 버튼 클릭 -> 클릭시마다 좋아요와  
좋아요 취소가 번갈아가면서 처리됨 -> ajax  
처리 -> 좋아요 개수를 즉시 리스트에 표출

- 게시글 삭제시 해당 게시글번호로 조회된  
좋아요 함께 삭제 처리

## 좋아요 / 좋아요 취소 시연 / 코드리뷰

#번호: 1, 로그인 userno: 14  
내용: 두찌빠찌포치  
해시태그: #미니언즈 #바나나  
등록일: 2018-11-02  
수정 삭제

신고하기 신고된갯수 :

좋아요 : 0 좋아요!

[댓글달기]

댓글...댓글을 보자!

#번호: 1, 로그인 userno: 14  
내용: 두찌빠찌포치  
해시태그: #미니언즈 #바나나  
등록일: 2018-11-02  
수정 삭제

신고하기 신고된갯수 :

좋아요 : 1 안좋아요

[댓글달기]

댓글...댓글을 보자!

## 좋아요 / 좋아요 취소 시연 / 코드리뷰

```
$(document).ready(function() {  
    var userNo = ${loginInfo.userNo};  
    $.ajax({  
        type : "GET",  
        url : "isGood",  
        data : {"userNo" : userNo},  
        success : function(isGood) {  
            if (isGood.length > 0) {  
                for (var i = 0; i < isGood.length; i++) {  
                    var bNo = isGood[i].boardNo;  
                    $('#is_' + bNo + '_' + userNo).empty();  
                    $('#is_' + bNo + '_' + userNo).append("안좋아요");  
                }  
            }  
        }  
    });  
});
```

---

# 게시물 댓글 - 김혜연

## [구현내용]

게시물 댓글 :

각 게시물에 댓글(코멘트)을

등록, 수정, 삭제할 수 있는 기능 구현

DB : MySQL

TABLE COMMENT

commentno / boardno / userid

commentcont / commentdate

댓글달기 -> DB에 댓글 등록 -> 수정, 삭제 기능

댓글보기 -> 해당 게시물번호로 DB조회 ->

게시물 하단으로 댓글 표출( ajax ) -> 댓글보기

다시 클릭시 표출된 댓글 숨김 처리

- 게시물 삭제시 해당 게시물번호로 조회된

댓글 함께 삭제 처리

# 댓글 등록, 수정, 삭제 시연 / 코드리뷰

#번호: 1, 로그인 userno: 14  
내용: 뚜찌빠찌포치  
해시태그: #미니언즈 #바나나  
등록일: 2018-11-02  
수정 삭제

신고하기 신고된갯수 :

좋아요 : 0 좋아요!

[댓글달기]

댓글...댓글을 보자!

#번호: 1, 로그인 userno: 14  
내용: 뚜찌빠찌포치  
해시태그: #미니언즈 #바나나  
등록일: 2018-11-02  
수정 삭제

신고하기 신고된갯수 :

좋아요 : 1 안좋아요

[댓글달기]

댓글...댓글을 보자!

총 댓글: 2

댓글번호 : 28  
[수정하기][삭제하기]  
작성자 : mini@gmail.com  
코멘트 : a

댓글번호 : 31  
[수정하기][삭제하기]  
작성자 : mini@gmail.com  
코멘트 : □□□□

# 댓글 등록, 수정, 삭제 시연 / 코드리뷰

```
status = $('#commHidden_' + bNo).css("display");
if (status == "none") {
    status = $('#commHidden_' + bNo).css("display", "");

    $.ajax({
        type : "GET",
        url : "viewComment",
        data : {"bNo" : bNo},
        dataType : "JSON",
        success : function(data) {
            commList = data;
            if (commList.length == 0) {
                comm += 'NO COMMENT!';
            } else {
                comm += '<br>총 댓글: ' + commList.length+ '<br>';

                for (var i = 0; i < commList.length; i++) {
                    comm += '<br>댓글번호 : ' + commList[i].commentNo+ '<br>';
                    comm += '<c:if test="${userId eq commList[i].userId}">';
                    comm += '<a href="editComment?commentNo='+ commList[i].commentNo+ '">[수정하기]</a>';
                    comm += '<a href="deleteComment?commentNo='+ commList[i].commentNo+ '&userId='+ userId+ '">[삭제하기]</a>';
                    comm += '</c:if>';
                    comm += '작성자 : ' + commList[i].userId+ '<br>';
                    comm += '코멘트 : ' + commList[i].commentCont+ '<br>';
                }
            }
            $('#commHidden_' + bNo).empty();
            $('#commHidden_' + bNo).append(comm);
        }
    });
} else {
    status = $('#commHidden_' + bNo).css("display", "none");
}
```

# 구현 목표 - 문상혁

## [목표]

- \* 우측 **SideBar**에 정의된 버튼을 이용하여 **SideBar** 내에 친구 목록 보여주도록 구현하기
- \* 친구 이름 클릭 시 친구 페이지 이동하여 친구 정보 확인 및 게시물에 좋아요, 댓글 등의 기능 수행하도록 구현하기
- \* 전달받은 친구 요청에 대해 수락/거절 기능, 친구 목록에 등록된 친구에 대하여 친구 해제(삭제) 기능 구현하기

## [완성]

- \* 전달받은 친구 요청에 대해 수락/거절 기능, 친구 목록에 등록된 친구에 대하여 친구 해제(삭제) 기능 구현 완료
- \* 친구 이름 클릭 시 친구 페이지 이동하여 친구 정보 확인 및 게시물에 좋아요, 댓글 등의 기능 수행 연동 완료
- \* 페이지 이동을 통해 친구 목록 보여주기 구현

## [미완성]

- \* 우측 **SideBar** 내에 친구 목록 보여주기  
(**ajax** 처리 필요할 것으로 예상)
- \* 친구 요청 수락 및 거절, 친구 해제(삭제) 시 **ajax** 처리  
(현재는 **location.reload**)



# 친구목록, 친구페이지 - 문상혁

## [구현내용]

- \* 친구 목록 가져오기
  - 친구 해제(삭제)
- \* 전달받은 친구 요청 관리
  - 요청 수락/거절
- \* 친구 페이지 이동 및 게시물 확인

DB : MySQL

relation, request table 사용

친구 목록

친구목록		
미나리야크	친구 해제	쪽지보내기

받은 친구 요청

요청목록		
유지영	수락	거절

친구 페이지

ID : alreadyna@bit.bit

이름 : 미나리야크

[친구신청]

#번호: 131 / UserNo: 12

사진: 

내용: /

해시태그:

등록일:

수정 삭제

신고하기

좋아요: 0

좋아요

[댓글달기]

댓글: 댓글들보러

#번호: 87 / UserNo: 12

사진: 

내용: c c c c

해시태그:

등록일:

수정 삭제

신고하기

좋아요: 1

좋아요

[댓글달기]

댓글: 댓글들보러

## 코드 리뷰 FriendController - 문상혁

Controller	기능	RequestMapping	RequestMethod
List	친구 목록 출력	friend/friendList	<b>- GET</b> 로그인 시 Session 저장된 로그인 정보 중 <b>userNo</b> 를 이용하여 DB에서 친구 목록 정보를 가져와 <b>view</b> 로 출력
Page	친구 목록에서 친구 선택 시 친구 페이지로 이동 및 내용 출력	friend/friendsPage/{userNo}	<b>- GET</b> <b>@PathVariable</b> 을 통하여 친구의 <b>userNo</b> 를 전달받아 DB에서 친구 정보를 가져오고 친구 페이지로 이동하여 친구 정보를 출력한다.

## 코드 리뷰 FriendController - 문상혁

Controller	기능	RequestMapping	RequestMethod
Reg	친구 요청 기능	friendReg	<b>- GET</b> 친구 페이지에서 친구 추가 요청 시 세션에 저장된 로그인 정보의 <b>userNo</b> 와 친구의 <b>userNo</b> 를 친구 요청 DB로 전달하여 친구 요청 대기상태로 만든다. 요청 취소 시 각 <b>userNo</b> 를 이용하여 친구 요청 DB에서 제거
Accept	받은 친구 요청을 수락 (Accept) /거절(Deny)	friendAccept	<b>- GET</b> <b>Accept</b> 시 세션에 저장된 로그인 정보의 <b>userNo</b> 와 친구의 <b>userNo</b> 를 확인 후 친구 요청 DB에 해당 하는 컬럼을 제거, 이 후 친구 관계 DB에 각 <b>userNo</b> 를 넣어 저장한다. <b>Deny</b> 시 친구 요청 DB의 정보만 제거 한다.

## 코드 리뷰 FriendController - 문상혁

Controller	기능	RequestMapping	RequestMethod
Delete	친구 끊기(삭제)	friendRemove	<b>- GET</b> 친구 관계 DB에서 본인과 친구의 userNo를 확인 후 제거

# 코드 리뷰 - 문상혁

## [Controller]

```
@Controller
public class FriendsListController {

    @Autowired
    private FriendsListService listService;

    @Autowired
    private FriendRequestService requestService;

    @RequestMapping("/friend/friendsList")
    public ModelAndView getFriendsList(HttpSession session) {

        // ModelAndView 메서드 사용을 위한 객체 선언
        ModelAndView mav = new ModelAndView();

        // 로그인 시 저장된 로그인 세션 정보를 MemberInfo 타입으로 새로 저장 (이 정보를 가지고 친구목록/받은 친구 요청 목록을 가져온다)
        MemberInfo loginInfo = (MemberInfo)session.getAttribute("loginInfo");

        // 가져온 친구 목록이 있으면 ModelAndView 객체에 friendsList란 이름으로 객체를 저장하여 view 단에 넘겨준다.
        if(!listService.viewFriendsList(loginInfo.getUserNo()).isEmpty()) {
            mav.addObject("friendsList", listService.viewFriendsList(loginInfo.getUserNo()));
        }

        // 가져온 친구요청 목록이 있으면 ModelAndView 객체에 friendsList란 이름으로 객체를 저장하여 view 단에 넘겨준다.
        if(!requestService.viewRequestList(loginInfo.getUserNo()).isEmpty()) {
            mav.addObject("requestList", requestService.viewRequestList(loginInfo.getUserNo()));
        }

        mav.setViewName("friend/friendsList");

        return mav;
    }
}
```

# 코드 리뷰 - 문상혁

## [Controller]

```
@Controller
public class FriendsPageController {

    @Autowired
    private FriendsViewService viewService;

    @Autowired
    private FriendRequestService requestService;

    @Autowired
    private BoardListService service;

    @Autowired
    private GetIsGoodCountAllService getIsGoodCountAllService;

    @RequestMapping(value = "/friend/friendsPage/{userNo}")
    public ModelAndView viewFriendsPage(@PathVariable("userNo") int friendNo, HttpSession session, HttpServletRequest request) {

        // ModelAndView 메서드 사용을 위한 객체 선언
        ModelAndView mav = new ModelAndView();

        // 현재 컨트롤러로부터 이동할 view단의 default 페이지 주소 설정
        mav.setViewName("friend/friendsPage");
        System.out.println("하이");
        System.out.println(viewService.viewFriendPage(friendNo));

        // 로그인 시 저장된 로그인 세션 정보를 MemberInfo 타입으로 새로 저장
        MemberInfo loginInfo = (MemberInfo) session.getAttribute("loginInfo");

        Friends friends = new Friends(loginInfo.getUserNo(), friendNo);

        int chkFriend = viewService.chkFriend(friends);

        FriendRequestInfo requestChk = new FriendRequestInfo(loginInfo.getUserNo(), friendNo);

        // MemberInfo 객체 형식으로 가져온 친구 정보를 view단에 friendInfo라고 넘겨준다.
        MemberInfo friendInfo = viewService.viewFriendPage(friendNo); // 친구 목록에 있는 친구 정보를 클릭하여 전달받은 친구의 userNo 값으로 친구
                                                                    // 정보를 가져옴
        mav.addObject("friendInfo", friendInfo);
    }
}
```

# 코드 리뷰 - 문상혁

## [Controller]

```
@Controller
public class FriendRegController {

    @Autowired
    private FriendRequestService requestService;

    @RequestMapping("/friendReg")
    @ResponseBody
    public String setFriendReg(FriendRequestInfo friendRequestInfo, Model model) {
        System.out.println("setFriendReg 접속");
        System.out.println(friendRequestInfo);
        int result = requestService.friendRequest(friendRequestInfo);
        // model.addAttribute("response", result);
        System.out.println(result + "");
        return result + "";
    }

    @RequestMapping("/friendRegCancel")
    @ResponseBody
    public String friendRegCancel(FriendRequestInfo friendRequestInfo) {

        System.out.println("friendRegCancel 접속");

        int result = requestService.deleteRequest(friendRequestInfo);

        return result + "";
    }
}
```

# 코드 리뷰 - 문상혁

## [Controller]

```
@Controller
public class FriendAcceptController {

    @Autowired
    private FriendAcceptService acceptService;

    @RequestMapping("/friendAccept")
    @ResponseBody
    public String friendAccept(FriendRequestInfo friendRequestInfo, Friends friends) {

        System.out.println("friendAccept 진행");

        int insertCnt = 0;
        int deleteCnt = 0;

        insertCnt = acceptService.friendAccept(friends);
        deleteCnt = acceptService.deleteRequest(friendRequestInfo);

        System.out.println("insertCnt의 값: " + insertCnt);
        System.out.println("accept용 deleteCnt의 값: " + deleteCnt);

        return "friend/friendsList";
    }

    @RequestMapping("/friendDeny")
    @ResponseBody
    public String friendDeny(FriendRequestInfo friendRequestInfo) {

        int deleteCnt = 0;

        deleteCnt = acceptService.deleteRequest(friendRequestInfo);

        System.out.println("deny용 deleteCnt의 값: " + deleteCnt);

        return "friend/friendsList";
    }
}
```



# 코드 리뷰 - 문상혁

## [Controller]

```
@Controller
public class FriendDeleteController {

    @Autowired
    private FriendDeleteService deleteService;

    @RequestMapping("/friendRemove")
    @ResponseBody
    public String deleteFriend(Friends friends) {
        System.out.println(friends);
        int deleteCnt = deleteService.deleteFriend(friends);

        return "friend/friendsList";
    }
}
```

# 쪽지,검색,친구요청/취소 - 하지승

## [구현내용]

쪽지,친구목록,검색 기능(DB CRUD)

친구목록 -> 쪽지 보내기

쪽지 목록 -> 답장

검색 기능

DB : MySQL

TABLE COMMENT

board / message / request

친구 목록 -> 쪽지 보내기-> messageDB에 저장

쪽지함 1. 메세지 확인

2. 쪽지 답장

3. 쪽지 삭제

검색 -> 입력받는 글 유형에 따라 나뉨

- 해시태그('#')포함된 유형
- 사용자 번호로 구분
- 그 외의 나머지 유형은 전부 게시물

친구목록 요청/요청 취소기능.

# 쪽지함 접근

MessageTo.jsp

ReadMessageController.java

MessageService.java

MemberDao.java

MemberMapper.xml

[MessageTo.jsp]

Script부분 Ajax - 처음 화면 로드시 메세지 확인 버튼 안 누른것의 개수가 출력되고 그 메시지당 각각 '읽음/답장/삭제'기능 있습니다.

friendsList.jsp

MessageWriteController.java

MessageService.java

MemberDao.java

MemberMapper.xml

[friendsList.jsp]

쪽지 보내기 버튼 눌렀을때의 작동

## 쪽지함 접근

```
/* 페이지 로드 될때 읽음 버튼 누른애들은 버튼 비활성화 */
$(document).ready(function() {
    $('.checkMessage').filter(function(text) {
        var secondthis = $(this);
        $(this).find('td[id*="unique"]').each(function(index, item) {
            if (item.innerHTML == "true") {
                secondthis.find('.ReadMessage').attr('disabled', true);
                secondthis.find('.ReadMessage').css('color', 'blue');
            }
        });
    });
});
```

# 쪽지함 접근

```
/* 읽음 버튼 누른애들 비활성화+메세지 읽음 처리 */
$('.ReadMessage').click(function(kk) {
    var messageNum = $(this).val();
    var attribute = $(this);
    /* 루트기준 절대경로 */
    var url = '/springsns/ReadMessage';
    var innerUrl = '/springsns/ReloadMessage'
    var checkMessage = '#unique' + messageNum;

    $.ajax({
        url : url,
        data : {
            /* 파라미터 넘기는것 '?num=ttt' */
            "messageno" : messageNum
        },
        success : function(response) {
            if (response == 1) {
                $(checkMessage).text('true');
                attribute.attr('disabled', true);
                attribute.css('color', 'red');
                $.ajax({
                    url:innerUrl,
                    data:{
                        "recvid":$('h3').attr('value')
                    },
                    success:function(result){
                        if($('span').attr('class')==$('h3').attr('value')){
                            $('span[value*="msgcnt"]').text(result.count);
                        }
                    }
                })
            }
        }
    })
})
```

# 검색 기능

layout\_top.jsp

SearchDataController.java

SearchDataService.java

DataInterface.java

DataInterface.xml

check.jsp

[layout\_top.jsp]

검색어 넣고 검색 button 누를시 service까지 접근해서 service에서

// 문자열을 숫자 형식을 바꿔 주기 위해 DecimalFormat사용

```
NumberFormat nF = NumberFormat.getInstance();
```

// parse의 위치가 맨 끝에있는지 확인하기 위해 파서포지션 사용

```
ParsePosition pos = new ParsePosition(0);
```

// Number형식으로 parse한것 String로 변환

```
String ldData = "" + nF.parse(searchData, pos);
```

# 검색 기능

```
// 문자열을 숫자 형식을 바꿔 주기 위해 NumberFormat 사용
NumberFormat nF = NumberFormat.getInstance();
// parse의 위치가 맨 끝에 있는지 확인하기 위해 파서포지션 사용
ParsePosition pos = new ParsePosition(0);
// Number형식으로 parse한것 String로 변환
String IdData = "" + nF.parse(searchData, pos);

boolean bool = searchData.contains("#");

if (bool) {
    paramDataDao.setHashtag(searchData);
    resultDataDao = data.searchHashTagData(paramDataDao);
} else if (!bool) {
    // 숫자인지 확인
    if (searchData.length() == pos.getIndex()) {
        paramDataDao.setUserno(IdData);
        friendsListView.setUserNo(Integer.parseInt(IdData));

        resultDataDao = data.searchIdData(paramDataDao);
        resultfriendsListView = data.searchUserId(friendsListView);
        searchAndPageResultData.setFriendsListView(resultfriendsListView);
    } else {
        // 숫자가 아니면 컨텐츠인걸로 확인
        paramDataDao.setBoardcontent(searchData);
        resultDataDao = data.searchContentData(paramDataDao);
    }
}
```

# 친구 요청 취소

friendsPage.jsp x FriendRegController.java FriendRequestService.java FriendsDaoInterface.java FriendsDaoInterface.xml

[friendsPage.jsp]

친구 신청 버튼 클릭 => ajax 통하여 요청중으로 변경 & 요청 취소 버튼 생성



---

# 구현 목표 - 김솔

## [목표]

- ★ 회원으로 로그인했을때, 공지사항 글쓰기버튼 및 수정삭제 숨겨주기
- ★ 공지사항 **CRUD**, 페이징처리
- ★ 회원/게시글/댓글 리스트 & 수정/삭제기능
- ★ 신고버튼은 하나의 게시물당 한번만 누를 수 있고, 3번이상 신고된 게시물은 블라인드처리해주기

DB>>

notice, member, comment, board, report,  
report\_cnt

## [완성]

공지사항 **CRUD**, 페이징처리

회원/게시글/댓글 리스트 처리 후,  
수정삭제

## [미완성]

신고 기능 **ajax**처리 미완성

---

# 코드 리뷰 - 김솔 NoticeViewController.java

## [코드]

```
Notice viewData = service.getNotice(noticeNo);  
service.updateHits(noticeNo);  
  
ModelAndView modelAndView = new ModelAndView();  
  
modelAndView.setViewName("notice/view");  
modelAndView.addObject("viewData", viewData);  
  
return modelAndView;  
}
```

## [리뷰]

리스트에서 제목을 클릭했을때에 게시글이 나타나게 해주고,클릭함과 동시에 조회수를 올려주는 쿼리를 실행시켜 다시 목록으로 돌아갔을때에 조회수가 올라간 것을 확인할 수 있음.

## 코드 리뷰 - 김솔

## 조회수 증가 쿼리

[코드]

```
<update id="updateHits">  
  update notice  
  set noticehit = noticehit + 1  
  where noticeno = #{noticeNo}  
</update>
```

[리뷰]

제목이 클릭되고 공지사항 게시물이  
조회됐을때,

notice테이블에 noticehit에 +1 처리를 해주고,

업데이트!

# 코드 리뷰 - 김솔 삭제 확인 처리

## [코드]

```
<input type="button" class="btn btn-default"
Onclick="deleteNo(${i.noticeNo})" value="삭제">
```

```
<script>
```

```
function deleteNo(noticeNo) {
if (confirm('삭제하시겠습니까?')) {
location.href="delete?noticeNo=" + noticeNo;}
}
</script>
```

## [리뷰]

삭제버튼을 눌러주면 Onclick이벤트로,

컨펌창이 띄워지고, 확인버튼을 눌렀을 때에만  
삭제처리해주는 컨트롤러로 이동하게 만들어줌!

## 코드 리뷰 - 김솔

## 뷰페이지에서 신고기능

[코드]

```
<c:if test="${boardcontent cont}">
```

```
input type="hidden" id="reportChk"  
value="${board.reportCheck}">
```

```
<button class="btn btn-default rechk" <c:if  
test="${board.reportCheck}"> disabled </c:if>  
value="${board.boardNo}" >신고하기</button>  
</c:if>
```

[리뷰]

신고받은게시물은 신고버튼이 보이지 않게 하고

버튼을 눌렀을때, **ajax**로 클릭 이벤트 처리하는

코드입니다 .....

# 코드 리뷰 - 김솔

## [코드]

```
$('#rechk').click(function() {
```

```
.....
```

```
if (chk) {  
$.ajax({
```

```
.....
```

```
$.ajax({  
    success : function (data) {  
  
        url : url + '/reportcnt',  
        data : {"boardNo" : boardNo},  
        success : function(result) {
```

if

```
(result == 1) {  
    $('#reportChk').val(chch);  
    $(obj).attr('disabled', true);  
    .....
```

## [리뷰]

신고버튼 눌렀을때에 구현되는 **ajax**코드

컨펌버튼 확인을 눌렀을 때에 **ajax**가 두번 실행

∴ 두개의 컨트롤러를 거쳐야함

게시글번호/사용자번호를 넣어주는 테이블과

게시글번호와 신고카운트를 해주는 테이블을

거쳐야하므로 두번의 **ajax**가 실행

하지만..완성되지못했습니다..

---

Thank you !!!

---