

Spring Framework 기반 LOL Community (+ Web Crawling)



Date: 2022.04

Name: 최성광

Phone: 010-9992-6962

Email: sungkwang0908@naver.com

Blog: <https://blog.naver.com/sungkwang0908>

GitHub: <https://github.com/CHOISUNGKWANG/My-Web-Project.git>

목차

어플리케이션 개요

개발 일정

ERD

페이지 기능

핵심 기능 및 로직

소감 및 추후 개선사항

어플리케이션 개요

구분	내용	
기획배경	지난 프로젝트에서 다뤄보지 못한 게시판을 다루어 보기 위해 주제를 선정하였고 네이버 카페의 자유게시판을 벤치마킹하여 구현해보기로 결정	
기획목적	게시판의 핵심 기능이라 생각되는 게시글+댓글+대댓글을 구현하고 게시글 목록에서 다양한 검색 필터링과 페이징 처리를 함께 할 수 있도록 구현하는 것이 목적	
기대효과	개인 프로젝트이다 보니 MVC 패턴의 흐름을 보다 상세히 이해할 수 있고 스프링 프레임워크를 사용하여 xml 설정과 어노테이션을 통해 컨테이너가 객체를 생성 및 관리하는 유지보수에 용이한 코드를 구현할 수 있음	
기능요약	• 게시글+댓글+대댓글	• 좋아요, 아이디+닉네임 중복검사 (ajax)
	• 검색 필터링+페이징 처리	• 이메일 인증 (메일 api)
	• 이미지 업로드	• 크롤링
개발환경	Java, Oracle, Apache Tomcat, Web Standard, JSP, Spring Framework	

[illegible][illegible]

ERD(Table)

member	변수	타입
아이디 (PK)	mid	varchar(50)
비밀번호	mpw	varchar(50)
닉네임	mnick	varchar(50)
이메일	memail	varchar(50)

board	변수	타입
게시글 (PK)	bid	int
아이디 (FK)	mid	varchar(50)
닉네임	mnick	varchar(50)
제목	bttitle	varchar(100)
이미지	bimg	varchar(50)
내용	bcontent	varchar(500)
조회수	bhits	int
좋아요수	blike	int
날짜	bdate	date

reply	변수	타입
댓글 (PK)	rid	int
게시글 (FK)	bid	int
아이디 (FK)	mid	varchar(50)
닉네임	mnick	varchar(50)
내용	rcontent	varchar(500)
날짜	rdate	date

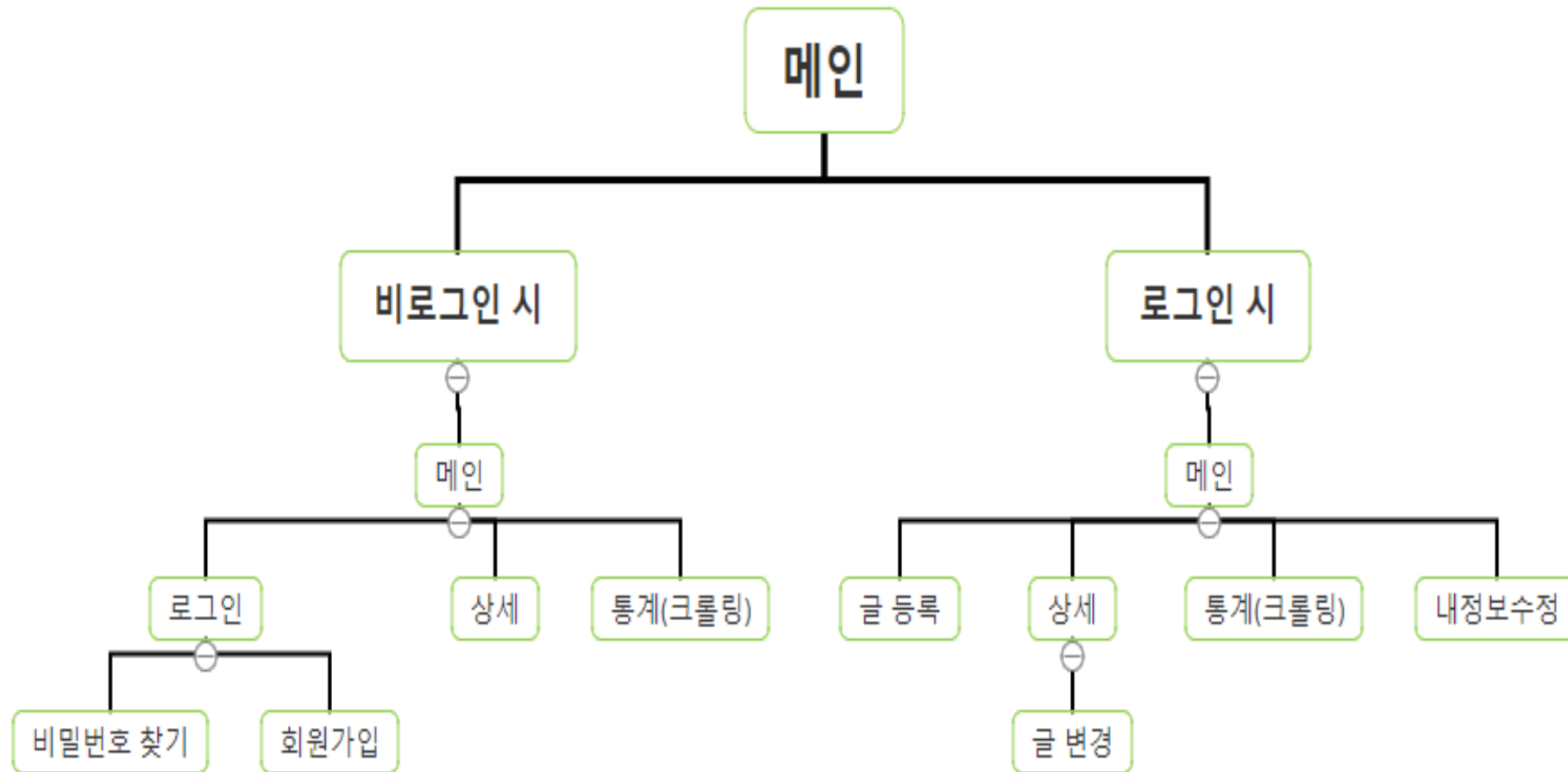
reply2	변수	타입
대댓글 (PK)	rrid	int
댓글 (FK)	rid	int
아이디 (FK)	mid	varchar(50)
닉네임	mnick	varchar(50)
내용	rrcontent	varchar(500)
날짜	rrdate	date

blike	변수	타입
게시글 (FK)	bid	int
아이디 (FK)	mid	varchar(50)

delete on cascade를 사용한
FK(외래키) 설정을 화살표로 표시

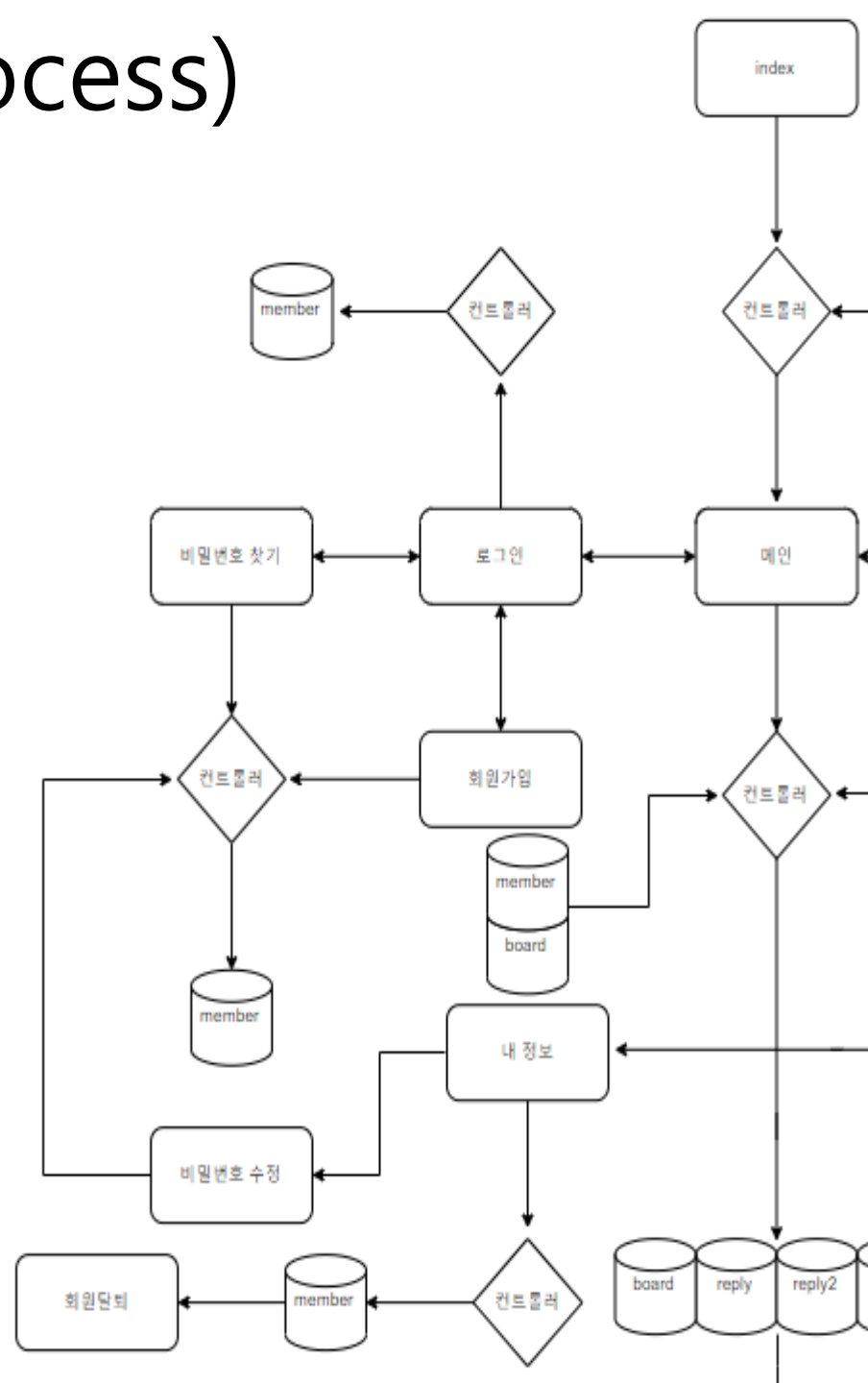
stats	변수	타입
포지션	sposition	varchar(50)
랭크	srank	varchar(50)
챔프 소스	schampsrc	varchar(100)
챔피언	schamp	varchar(50)
티어 소스	stiersrc	varchar(50)
승률	swinrate	varchar(50)
픽률	spickrate	varchar(50)

ERD(User Flow)



사용자의 이용 흐름을
로그인 / 비로그인 시로 나누어
페이지를 기준으로 나타냄

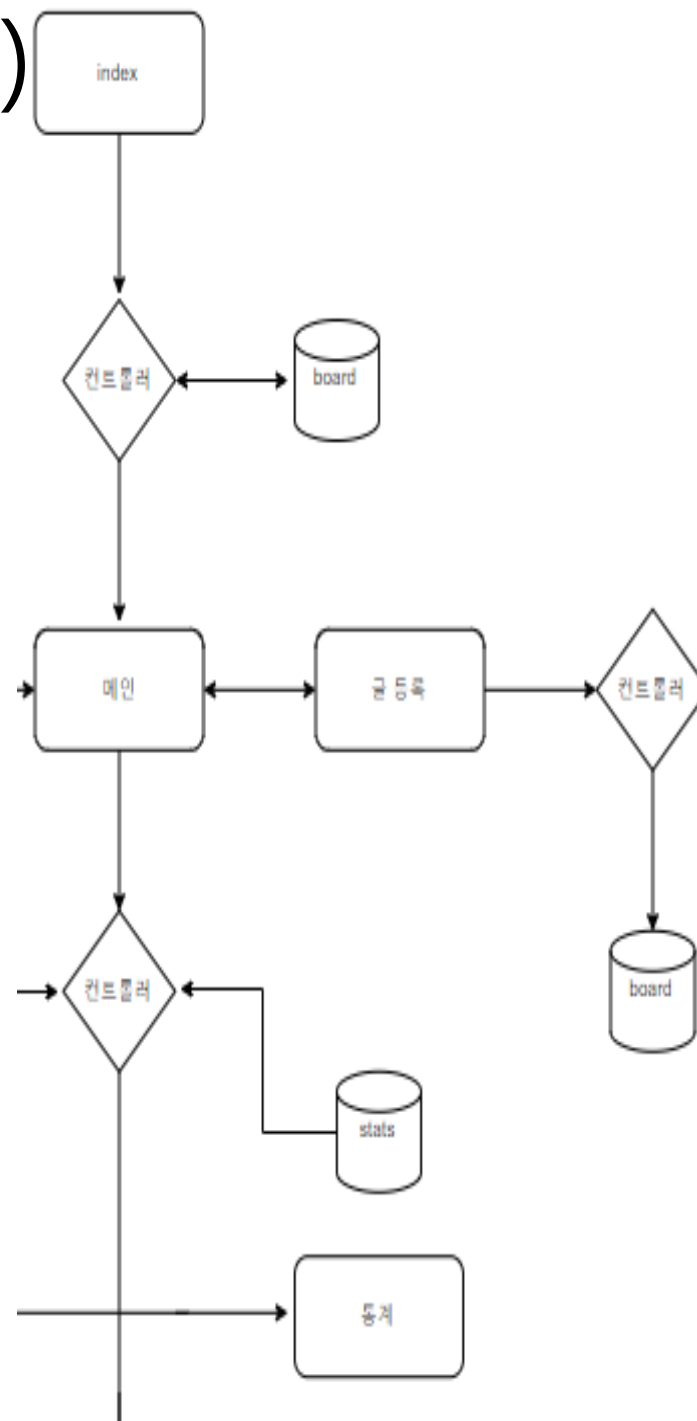
ERD(Logic Process)



member DB를 거치는 영역

- 로그인
- 회원가입
- 비밀번호 찾기
- 내 정보
- 비밀번호 수정
- 회원탈퇴

ERD(Logic Process)



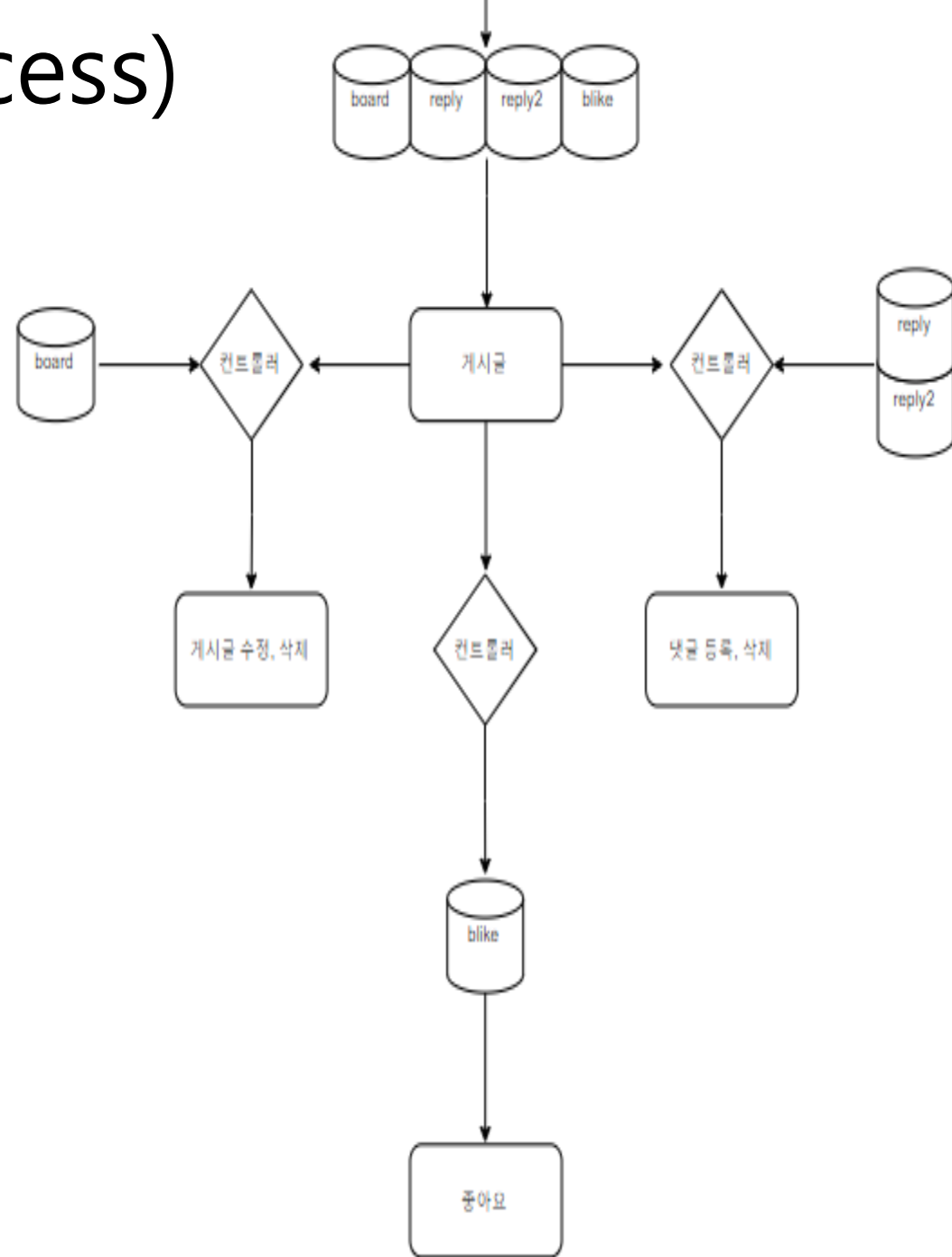
board DB를 거치는 영역

- 게시물 등록

stats DB를 거치는 영역

- 통계 페이지 (크롤링)

ERD(Logic Process)



board DB를 거치는 영역

- 게시글 수정, 삭제

reply DB를 거치는 영역

- 댓글 등록, 삭제

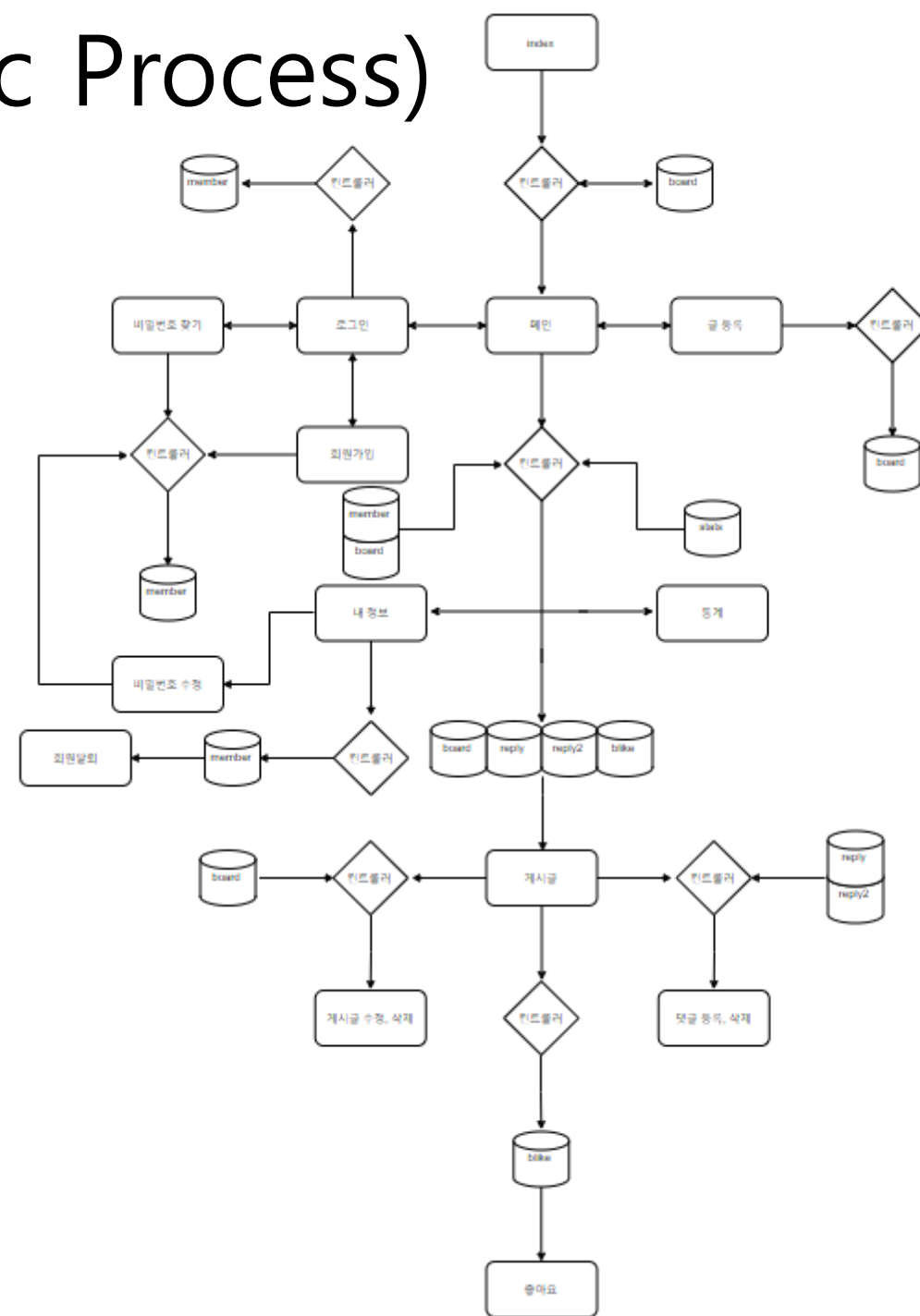
reply2 DB를 거치는 영역

- 대댓글 등록, 삭제

blike DB를 거치는 영역


- 게시글 좋아요

전체 Logic Process의 모습



페이지 기능

메인 페이지



최성광님 환영합니다.

Logout

PAGES

Home

Statistics

Authentications

4

Search... 제목 최신순

자유 게시판☆

5 글 작성하기

6

#	제목	작성자	작성일	조회	좋아요
135	실시간 구마유시 기인 언급 채팅 ㅋㅋㅋ.jpg	최성광	2022-04-19 17:39:38.0	16	0
134	페이커는 이럴땐 입꼭닫하고 소신발언 절대안하더라 ㅋㅋ	최성광	2022-04-19 17:39:37.0	19	0
133	현재 기인한테 가장 필요한거...	최성광	2022-04-19 17:39:36.0	17	0
132	국대 구성이 다 말없는 애들만 있네	최성광	2022-04-19 17:39:35.0	8	1
131	브랜드 서פות하면 좋은점	최성광	2022-04-19 17:39:34.0	36	3
130	카밀 착취 평타도 고정딜 전환됨??	최성광	2022-04-19 17:39:33.0	18	1
129	우제야 탑은 탑이 챙겨야지	최성광	2022-04-19 17:39:32.0	27	0
128	레오나서פות이 왜 2티어나	최성광	2022-04-19 17:39:31.0	56	0
127	케리아랑 조비 불화설 있던데	최성광	2022-04-19 17:39:30.0	36	0
126	난 풀캠정글 박는 브실골풀다 보면 이해가 안됨	최성광	2022-04-19 17:39:29.0	24	2

7 12345678910>

© 2022 , made with ❤ by ThemeSelection

License More Themes Documentation Support

- 로그인 시 생성되는 로그아웃 버튼
- 통계 페이지로 이동
- 로그인 시 내 정보 페이지로 이동
비로그인 시 로그인, 회원가입, 비밀번호 찾기 기능 제공
- 검색 필터(정렬) 제목, 작성자 최신순, 조회순, 좋아요순
- 로그인 시 글 작성 페이지로 이동 가능
- 글 상세 페이지로 이동
- 페이징 처리

로그인 페이지

The image shows a login page for the LOLC community. It features a white card with a blue header containing the LOLC logo. Below the header is a welcome message and a yellow bell icon. There are two input fields: one for the ID (labeled '아이디') and one for the password (labeled '비밀번호'). A blue '로그인' (Login) button is positioned below the password field. At the bottom of the card, there are two links: '비밀번호 찾기' (Forgot Password) and '회원가입' (Sign Up). Numbered callouts are placed around the UI: 1 points to the LOLC logo, 2 points to the '로그인' button, 3 points to the '비밀번호 찾기' link, and 4 points to the '회원가입' link.

① LOLC

Welcome to LOL Community! 🔔

ID

아이디

PASSWORD

비밀번호

② 로그인

③ 비밀번호 찾기 | ④ 회원가입

1. 로고 클릭 시
메인 페이지로 이동
2. 아이디, 비밀번호 정보 일치 시
메인 페이지로 이동
정보 불일치 시
로그인 페이지로 이동
3. 비밀번호 찾기 페이지로 이동
4. 회원가입 페이지로 이동

회원가입 페이지

① LOLC

회원가입 🚀

② ID

sungkwang

사용 가능한 아이디입니다.

NICKNAME

성광

사용 가능한 닉네임입니다.

PASSWORD

비밀번호

③ EMAIL

이메일

인증코드 전송

CODE

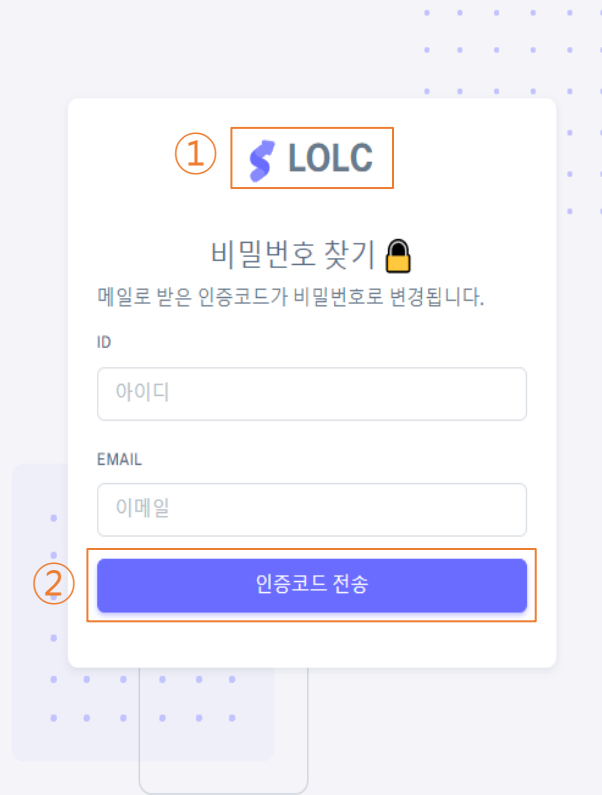
인증코드

④ [이용약관](#)에 모두 동의합니다.


⑤ 가입하기

1. 로고 클릭 시
메인 페이지로 이동
2. Ajax를 이용하여
아이디, 닉네임 중복 검사
(포커스 아웃 이벤트 처리)
3. 메일 API를 이용한 인증 처리
4. 이용약관 클릭 시
약관 페이지로 이동
5. 회원가입 완료 시
메인 페이지로 이동

비밀번호 찾기 페이지



① LOLC

비밀번호 찾기 

메일로 받은 인증코드가 비밀번호로 변경됩니다.

ID

아이디


EMAIL

이메일

② 인증코드 전송

1. 로고 클릭 시
메인 페이지로 이동
2. 인증코드 전송 버튼 클릭 시
입력한 이메일로
인증코드가 전송되고
전송된 인증코드로
회원 비밀번호가 변경됨

내 정보 페이지



최성광님 환영합니다.

Logout

PAGES

Home

Statistics

Authentications

Q Search...

제목 ▾ 최신순 ▾

LOL Community ☆

1

내 정보

2 회원탈퇴

ID

tjdrhkd1234a

NICKNAME

최성광

PASSWORD

.....

3 수정

EMAIL

sungkwang0908@naver.com

4

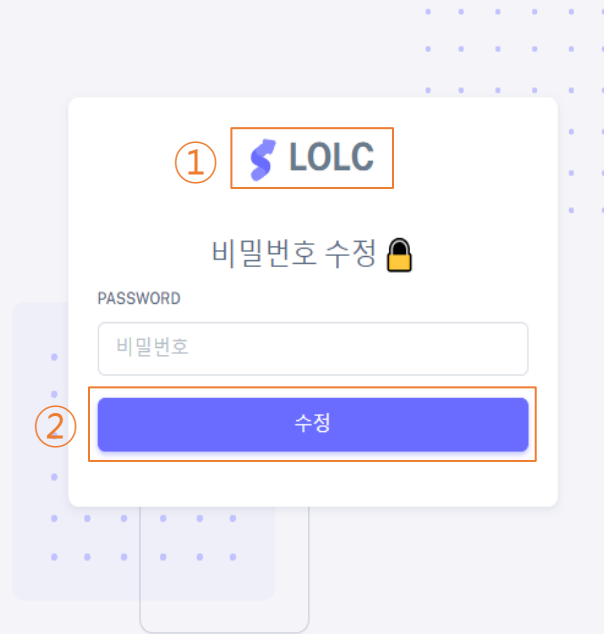
내 게시글

#	제목	작성자	작성일	조회	좋아요
46	데미글	최성광	2022-04-21 19:30:04.0	0	0
45	데미글	최성광	2022-04-21 19:30:03.0	0	0
44	데미글	최성광	2022-04-21 19:30:02.0	0	0
43	데미글	최성광	2022-04-21 19:30:01.0	0	0
42	데미글	최성광	2022-04-21 19:30:00.0	0	0
41	데미글	최성광	2022-04-21 19:29:59.0	0	0
40	데미글	최성광	2022-04-21 19:29:58.0	0	0
39	데미글	최성광	2022-04-21 19:29:57.0	0	0
38	데미글	최성광	2022-04-21 19:29:56.0	0	0
37	데미글	최성광	2022-04-21 19:29:55.0	0	0

12345

- 로그인 시 해당 회원의 정보를 출력
- 회원탈퇴 클릭 시 주의 알림창이 뜨고 계속 진행할 시 해당 회원의 정보, 게시글, 댓글 등의 데이터가 모두 삭제됨 (외래키 처리)
- 수정 버튼을 클릭 시 비밀번호 수정 페이지로 이동
- 로그인한 회원의 게시글을 모두 불러와 출력

비밀번호 수정 페이지



① LOLC

비밀번호 수정 🔒

PASSWORD

비밀번호

② 수정

1. 로고 클릭 시
메인 페이지로 이동
2. 수정 버튼 클릭 시
기존의 비밀번호가
입력한 비밀번호로 변경됨

게시글 작성 페이지



최성광님 환영합니다.

Logout

PAGES

Home

Statistics

Authentications

Search...

제목

최신순

LOL Community ☆

글 작성하기

작성자

최성광

제목

LCK에서 비주열 가장 뛰어난 팀이 어디임?

이미지 업로드

파일 선택

T1.jpg



내용

T1?

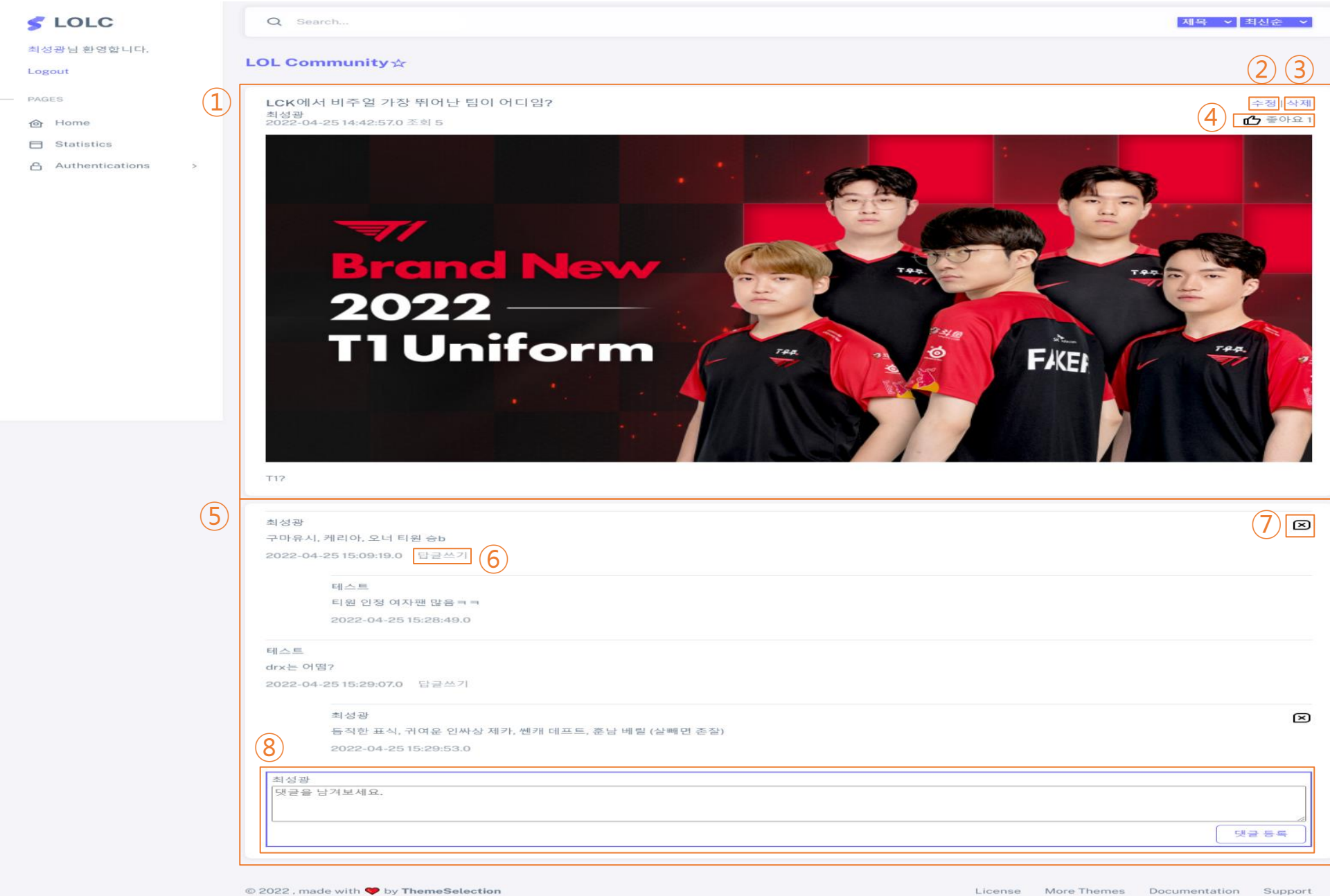
4

등록

1. 글 제목을 입력하는 영역
2. 업로드할 이미지를 선택하면 선택한 이미지를 미리 출력함
3. 글 내용을 입력하는 영역
4. 등록하기 버튼을 통해 입력한 정보들을 DB에 저장함

이후 작성한 글을 볼 수 있는 상세 페이지로 이동

게시글 상세 페이지



1. 작성한 정보 출력
(제목, 작성자, 등록일자, 조회, 내용)
2. 수정, 삭제 버튼은
로그인한 회원의 정보와
게시글 작성자의 정보가
서로 일치할 때 제공됨
3. 삭제 클릭 시 주의 알림창이 뜨고
계속 진행할 시
DB에서 게시글 정보가 삭제됨

외래키로 연결된 댓글, 대댓글 등의
정보 역시 모두 삭제됨
4. 좋아요 버튼을 클릭할 때
요청에 의한 페이지 이동 제거를 위해
Ajax를 사용해 기능을 구현함
5. 댓글, 대댓글 정보 출력
(작성자, 내용, 등록일자)
6. 댓글의 경우만 답글쓰기 버튼을 추가

클릭 시 8번과 같은 형태의
댓글 등록칸이 바로 아래에 생성됨
7. 댓글, 대댓글 삭제 버튼의 경우
2번, 3번과 동일한 프로세스로 진행됨
8. 댓글 내용을 입력하고
댓글 등록 버튼을 클릭하면
데이터가 DB에 저장되고
생성된 댓글을 확인할 수 있음

게시글 수정 페이지

LOLC

최성광님 환영합니다.

Logout

PAGES

Home

Statistics

Authentications

Search...

제목

최신순

LOL Community ☆

글 수정하기

1

작성자

최성광

제목

LCK에서 비주얼 가장 뛰어난 팀이 어디일?

이미지 업로드

파일 선택

선택된 파일 없음



내용

T1?

2

수정

1. 게시글에 대한 정보들을 출력
2. 수정 버튼 클릭 시
수정한 또는 수정하지 않은
데이터가 DB에 업데이트

이미지의 경우
수정을 하지 않으면
컨트롤러에서 기존의 정보로
set 해주고 업데이트를 진행

통계 페이지



PAGES

Home

Statistics

Authentications >

Search...

재목 ▼ 최신순 ▼

LOL Community☆

챔피언 티어 TOP

버전 12.7

랭크	챔피언	티어	승률	픽률
1		강플랭크	57.5%	7.1%
2		피오라	58.8%	3.9%
3		그레이브즈	54.4%	4.1%
4		아트록스	51.6%	4.0%
5		잭스	57.8%	2.9%
6		이렐리아	50.9%	3.6%
7		말파이트	57.0%	2.6%
8		리븐	53.2%	3.1%

LOLC

LOL Community☆

챔피언 티어 TOP

버전 12.7

랭크	챔피언	티어	승률	픽률
1		강플랭크	57.5%	7.1%
2		피오라	58.8%	3.9%
3		그레이브즈	54.4%	4.1%
4		아트록스	51.6%	4.0%
5		잭스	57.8%	2.9%
6		이렐리아	50.9%	3.6%
7		말파이트	57.0%	2.6%
8		리븐	53.2%	3.1%

챔피언 티어 JUNGLE

버전 12.7

랭크	챔피언	티어	승률	픽률
1		그레이브즈	55.4%	5.0%
2		리신	55.7%	3.8%
3		우디르	48.8%	0.3%
4		이블린	49.3%	0.7%
5		가렌	48.4%	0.4%
6		야소오	48.8%	0.7%
7		야소오	51.8%	0.2%
8		야소오	50.8%	0.1%
9		야소오	42.2%	0.7%
10		야소오	57.9%	0.3%

챔피언 티어 MID

버전 12.7

랭크	챔피언	티어	승률	픽률
1		야소오	55.4%	10.4%
2		야소오	54.7%	5.0%
3		야소오	54.3%	4.7%
4		야소오	52.7%	4.2%
5		야소오	52.4%	4.7%
6		야소오	50.2%	0.4%
7		야소오	57.9%	0.3%
8		야소오	50.4%	0.3%
9		야소오	49.4%	0.3%
10		야소오	51.4%	0.3%

챔피언 티어 AD CARRY

버전 12.7

랭크	챔피언	티어	승률	픽률
1		야소오	55.4%	10.4%
2		야소오	52.2%	10.2%
3		야소오	50.2%	7.8%
4		야소오	53.4%	10.7%
5		야소오	50.4%	5.4%
6		야소오	51.4%	5.4%
7		야소오	54.4%	0.3%
8		야소오	50.2%	0.7%
9		야소오	54.4%	0.3%
10		야소오	54.4%	0.7%

챔피언 티어 SUPPORT

버전 12.7

랭크	챔피언	티어	승률	픽률
1		야소오	54.4%	11.5%
2		야소오	54.4%	10.2%
3		야소오	53.4%	5.4%
4		야소오	53.4%	4.3%
5		야소오	51.4%	0.1%
6		야소오	48.2%	2.4%
7		야소오	49.2%	0.4%
8		야소오	50.4%	0.4%
9		야소오	51.2%	0.4%
10		야소오	50.2%	0.4%

© 2022. made with ♥ by ThemeRelatives

Home About Us ThemeRelatives

poro.gg에서 크롤링한 데이터를
DB에 저장하고,
포지션 별로 데이터를 추출하여
VIEW에서 출력

핵심 기능 및 로직

이미지 업로드 (1/2)

①

```
<!-- cos.jar -->
<dependency>
  <groupId>servlets.com</groupId>
  <artifactId>cos</artifactId>
  <version>05Nov2002</version>
</dependency>
```

②

```
<form action="/insertBoard.do" method="post" enctype="multipart/form-data">
```

③

```
@Autowired
private ServletContext sc;

@RequestMapping(value="/view/insertBoard.do")
public String insertBoard(BoardVO bvo,HttpServletRequest req) throws IOException {
    String realFolder = sc.getRealPath("file");
    int maxSize = 1024*1024*200;
    String encType = "UTF-8";
    MultipartRequest mr = new MultipartRequest(req,realFolder,maxSize,encType,new DefaultFileRenamePolicy());
    String frn = mr.getFilesystemName("bimg");
    bvo.setMid(mr.getParameter("mid"));
    bvo.setMnick(mr.getParameter("mnick"));
    bvo.setBtitle(mr.getParameter("btitle"));
    bvo.setBimg(frn);
    bvo.setBcontent(mr.getParameter("bcontent"));
    boardService.insertBoard(bvo);
    return "redirect:main.do";
}
```

1. pom.xml에 이미지 업로드를 위한 cos.jar dependency 추가
2. View의 form 태그에 multipart enctype 속성을 추가 후 이미지 정보를 받을 수 있게 구현
3. 컨트롤러에서 해당 요청을 받으면 실제 사진이 저장될 realFolder 명을 정하고 MultipartRequest를 사용하여 View에서 넘어온 데이터를 받아서 set한 뒤 비즈니스 메서드를 사용하여 DB에 저장

(org.eclipse.wst.server.core
폴더 안에 해당 프로젝트 내에
realFolder 명으로 정한
이름으로 폴더 생성)

이미지 업로드 (2/2)

①

```
private final String B_INSERT = "insert into board (bid,mid,mnick,btitle,bimg,bcontent)"
    + " values((select nvl(max(bid),0)+1 from board),?,?,?,?); // 글 등록
// 글 등록
public void insertBoard(BoardVO vo) {
    conn = JDBCUtil.connect();
    try {
        pstmt = conn.prepareStatement(B_INSERT);
        pstmt.setString(1, vo.getMid());
        pstmt.setString(2, vo.getMnick());
        pstmt.setString(3, vo.getBtitle());
        pstmt.setString(4, vo.getBimg());
        pstmt.setString(5, vo.getBcontent());
        pstmt.executeUpdate();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } finally {
        JDBCUtil.disconnect(pstmt, conn);
    }
}
```

1. CRUD 중 C에 해당하는 비즈니스 메서드로 PK bid를 select nvl를 사용하여 처리하였고 컨트롤러에서 set 해서 보내준 데이터들을 board table에 저장
2. View에서 저장된 이미지 파일을 불러올 때 시스템 실제 경로 + table에 저장된 이미지명으로 불러올 수 있음

②

```

```


게시글+댓글+대댓글 (1/4)

```
① @RequestMapping(value="/view/detail.do")
    public String userDetails(HttpSession session, BoardVO bvo, ReplyVO rvo, LikeVO lvo, Model model) {
        String mid=(String)session.getAttribute("mid");
        BoardVO data = boardService.getBoard(bvo);
        if(mid==null||!mid.equals(data.getMid())) {
            boardService.updateBoard_hitsUp(bvo);
        }
        data = boardService.getBoard(bvo);
        model.addAttribute("bdata",data);
        ArrayList<ReplySet> datas = replyService.getReplyList(rvo);
        model.addAttribute("rdatas",datas);
        lvo = likeService.getLike(lvo);
        model.addAttribute("ldata",lvo);
        return "detail.jsp";
    }
```

```
② package com.test.app.model;

import java.util.ArrayList;

public class ReplySet {
    private ReplyVO replyVO;
    private ArrayList<Reply2VO> reply2List = new ArrayList<Reply2VO>();

    public ReplyVO getReplyVO() {
        return replyVO;
    }
    public void setReplyVO(ReplyVO replyVO) {
        this.replyVO = replyVO;
    }
    public ArrayList<Reply2VO> getReply2List() {
        return reply2List;
    }
    public void setReply2List(ArrayList<Reply2VO> reply2List) {
        this.reply2List = reply2List;
    }

    @Override
    public String toString() {
        return "ReplySet [replyVO=" + replyVO + ", reply2List=" + reply2List + "]";
    }
}
```

1. View에서 사용자에게 의해
게시글 상세 페이지로
가는 detail.do 요청을
컨트롤러에서 받으면
비로그인 이용자일 때 또는
session으로 받은 mid 정보와
해당 게시글의 정보에서 가져
온 작성자 mid가 다를 때
즉, 현재 사용자와
게시글 작성자가 다르다면
조회수이 올라가도록 구현

* 작성자가 자신의 글을
들어갔을 때 조회수가
올라가는 것을 방지

이 후 올라간 조회수를
반영하여 DB에서 다시
게시글 데이터를 가져오고,
ReplySet으로 이루어진
해당 게시글의 댓글+대댓글
목록 데이터를 가져오고,
데이터가 null인지 아닌지
여부에 따라 게시글의
좋아요를 확인하는 데이터를
가져온 뒤 model에 담아
View로 데이터들을 전달

2. 댓글과 대댓글 리스트로
이루어진 ReplySet 클래스

게시글+댓글+대댓글 (2/4)

```
private final String B_SELECTONE_BID = "select * from board where bid=?"; // 글 정보 불러오기
private final String L_SELECTCNT_BLIKE = "select count(*) from blike where bid=?"; // 좋아요 수 카운트
private final String B_UPDATE_BLIKECNT = "update board set blike=? where bid=?"; // 좋아요 수 업데이트

// 글 정보 불러오기
public BoardVO getBoard(BoardVO vo) {
    conn = JDBCUtil.connect();
    BoardVO data = null;
    try {
        pstmt = conn.prepareStatement(B_SELECTONE_BID);
        pstmt.setInt(1, vo.getBid());
        ResultSet rs = pstmt.executeQuery();
        if(rs.next()) {
            data = new BoardVO();
            data.setBid(rs.getInt("bid"));
            data.setMid(rs.getString("mid"));
            data.setMnick(rs.getString("mnick"));
            data.setBtitle(rs.getString("btitle"));
            data.setBimg(rs.getString("bimg"));
            data.setBcontent(rs.getString("bcontent"));
            data.setBhits(rs.getInt("bhits"));

            pstmt = conn.prepareStatement(L_SELECTCNT_BLIKE);
            pstmt.setInt(1, vo.getBid());
            ResultSet rs2 = pstmt.executeQuery();
            rs2.next();
            int blikeCnt = rs2.getInt(1);
            rs2.close();

            pstmt = conn.prepareStatement(B_UPDATE_BLIKECNT);
            pstmt.setInt(1, blikeCnt);
            pstmt.setInt(2, vo.getBid());
            pstmt.executeUpdate();

            data.setBlike(blikeCnt);
            data.setBdate(rs.getString("bdate"));
        }
        rs.close();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } finally {
        JDBCUtil.disconnect(pstmt, conn);
    }
    return data;
}
```

CRUD 중 R에 해당하는 비즈니스
메서드로 board table에서
게시글 정보를 가져오는데 사용됨

게시글의 PK bid를 사용해
해당하는 게시글의 정보를
불러올 때 좋아요 수의 경우
blike table에서 해당 bid가 속하는
데이터 개수를 count하고
count한 개수를 이용해 해당 bid의
좋아요 수를 update한 뒤
좋아요 수를 set해
게시글 정보를 가져옴

* 이러한 방식으로 로직을
구현한 이유는 회원탈퇴 시
외래키 연결에 의해
좋아요 정보들도 함께 삭제돼서
이를 반영하기 위한 목적 때문

게시글+댓글+대댓글 (3/4)

```
private final String RS_SELECTALL_BID = "select * from reply where bid=? order by rid asc"; // 댓글 등록순으로 불러오기
private final String R2_SELECTALL_RID = "select * from reply2 where rid=? order by rrid asc"; // 대댓글 등록순으로 불러오기
// 댓글+대댓글 등록순으로 불러오기
public ArrayList<ReplySet> getReplyList(ReplyVO vo) {
    conn = JDBCUtil.connect();
    ArrayList<ReplySet> datas = new ArrayList<ReplySet>();
    try {
        pstmt = conn.prepareStatement(RS_SELECTALL_BID);
        pstmt.setInt(1, vo.getBid());
        ResultSet rs = pstmt.executeQuery();
        while(rs.next()) {
            ReplySet rps = new ReplySet();

            ReplyVO replyVO = new ReplyVO();
            replyVO.setRid(rs.getInt("rid"));
            replyVO.setBid(rs.getInt("bid"));
            replyVO.setMid(rs.getString("mid"));
            replyVO.setMnick(rs.getString("mnick"));
            replyVO.setRcontent(rs.getString("rcontent"));
            replyVO.setRdate(rs.getString("rdate"));
            rps.setReplyVO(replyVO);

            ArrayList<Reply2VO> reply2List = new ArrayList<Reply2VO>();
            pstmt=conn.prepareStatement(R2_SELECTALL_RID);
            pstmt.setInt(1, rs.getInt("rid"));
            ResultSet rs2 = pstmt.executeQuery();
            while(rs2.next()) {
                Reply2VO reply2VO = new Reply2VO();
                reply2VO.setRrid(rs2.getInt("rrid"));
                reply2VO.setRid(rs2.getInt("rid"));
                reply2VO.setMid(rs2.getString("mid"));
                reply2VO.setMnick(rs2.getString("mnick"));
                reply2VO.setRrcontent(rs2.getString("rrcontent"));
                reply2VO.setRrdate(rs2.getString("rrdate"));
                reply2List.add(reply2VO);
            }
            rs2.close();
            rps.setReply2List(reply2List);

            datas.add(rps);
        }
        rs.close();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } finally {
        JDBCUtil.disconnect(pstmt, conn);
    }
    return datas;
}
```

CRUD 중 R에 해당하는 비즈니스 메서드로 reply, reply2 table에서 댓글+대댓글 정보를 리스트로 가져오는 로직

해당 게시글의 bid(PK)로 댓글 정보를 불러오고
불러온 댓글의 rid(PK)로 대댓글 정보 리스트를 불러온 뒤 ReplySet에 담아 리스트에 추가하는 것을 반복하는 로직으로 해당 게시글의 댓글+대댓글 정보들을 모두 불러오게 됨

게시글+댓글+대댓글 (4/4)

```
private final String L_SELECTONE = "select * from blike where bid=? and mid=?"; // 좋아요 상태 확인용
// 좋아요 상태 확인용
public LikeVO getLike(LikeVO vo) {
    conn = JDBCUtil.connect();
    LikeVO data = null;
    try {
        pstmt = conn.prepareStatement(L_SELECTONE);
        pstmt.setInt(1, vo.getBid());
        pstmt.setString(2, vo.getMid());
        ResultSet rs = pstmt.executeQuery();
        if(rs.next()) {
            data = new LikeVO();
            data.setBid(rs.getInt("bid"));
            data.setMid(rs.getString("mid"));
        }
        rs.close();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } finally {
        JDBCUtil.disconnect(pstmt, conn);
    }
    return data;
}
```

CRUD 중 R에 해당하는 비즈니스 메서드로 blike table에서 현재 사용자의 mid와 게시글의 bid를 이용하여 데이터가 있으면 좋아요를 눌렀던 게시글이고 데이터가 null이면 좋아요를 누르지 않았던 게시글로 좋아요 여부를 확인할 수 있게 하는 메서드

페이징 처리 + 검색 필터링 (1/3)

```
<choi:pagination pageSize="10" queryStringName="page" recordCount="${recordCount}" keywords="${keywords}" category="${category}" searchCondition="${searchCondition}"/>
```

①

②

```
<%
int recordCount = (Integer) jspContext.getAttribute("recordCount");
int pageSize = (Integer) jspContext.getAttribute("pageSize");
String name = (String) jspContext.getAttribute("queryStringName");
String keywords = (String) jspContext.getAttribute("keywords");
String category = (String) jspContext.getAttribute("category");
String searchCondition = (String) jspContext.getAttribute("searchCondition");
```

③

```
int currentPage = 1;
if (request.getParameter(name) != null)
    currentPage = Integer.parseInt(request.getParameter(name));

int pageCount = recordCount / pageSize;
if (pageCount * pageSize < recordCount)
    ++pageCount;
```

```
String queryString = request.getQueryString();
if(queryString != null){
    int idx = queryString.indexOf("&");
    if(idx != -1){
        queryString = queryString.substring(0, idx);
    }
}
```

④

```
if (queryString == null)
    queryString = name + "@@@";
else if (queryString.matches(".*" + name + "[0-9]+.*"))
    queryString = queryString.replaceAll(name + "[0-9]+", name + "@@@");
else
    queryString = queryString + "&" + name + "@@@";
String url;
```

```
if (keywords == "" && category == "" && searchCondition == "") {
    url = request.getAttribute("javax.servlet.forward.request_uri") + "?" + queryString;
} else {
    url = request.getAttribute("javax.servlet.forward.request_uri") + "?" + queryString + "&keywords=" + keywords
        + "&category=" + category + "&searchCondition=" + searchCondition;
}
```

⑤

```
if (currentPage > pageCount)
    currentPage = pageCount;
int base = ((currentPage - 1) / 10) * 10;

ArrayList<Page> pages = new ArrayList<Page>();
if (base > 0)
    pages.add(new Page(base, "&lt;"));
for (int i = 1; i <= 10; ++i) {
    int n = base + i;
    if (n > pageCount)
        break;
    pages.add(new Page(n, String.valueOf(n)));
}
int n = base + 11;
if (n <= pageCount)
    pages.add(new Page(n, "&gt;"));
}%>
```

1. View에서 커스텀 태그로 불러오는 코드로 페이징 처리 부분을 태그로 분리해서 사용하고 있고 6개의 값을 태그로 넘겨줌

2. 구글링을 통해 가져온 페이징 처리 코드를 수정해서 사용

기존 코드에서
페이징 처리에 사용되는 값은
recordCount(게시글이 몇 개인지),
pageSize(한 페이지에 몇 개의
게시글을 보게 할 것인지),
queryStringName(요청할 때 url에
어떤 이름으로 보여줄 것인지)
이렇게 3개가 필요한데 추가로
검색 후 페이징 번호를 눌렀을 때도
검색된 데이터를 유지시키기 위해
검색에 필요한 값들을 추가하고
페이징 코드를 수정해서 사용

3. 내가 만든 검색 로직에 사용되는
3가지 값: keywords(입력 정보),
category(제목, 작성자 검색 여부),
searchCondition(정렬)을 페이징
코드에서도 받을 수 있게 추가

4. 페이징 처리와 검색 로직을 묶을 시
요청 url을 중복해서 받아오는
문제가 발생하는데 이를 해결하기
위해 받은 url에서 (&)가 몇 번째에
있는지 찾은 다음 substring으로
나머지를 잘라서 해결

5. 전체 게시글을 불러오는 요청이
있을 때와 검색을 사용하는 요청을
나누어 url을 바꿔주는 로직 추가

페이징 처리 + 검색 필터링 (2/3)

```
@RequestMapping(value="/view/search.do")
public String search(HttpServletRequest req, BoardVO bvo, Model model) {
    int cnt = 0;
    int recordCount = 0;
    if(req.getParameter("page")==null) {
        cnt = 1;
    }else {
        cnt = Integer.parseInt(req.getParameter("page"));
    }
    ArrayList<BoardVO> datas = null;
    if(req.getParameter("keywords")==null||req.getParameter("keywords")=="" ) {
        recordCount=boardService.getBoardCntAll(bvo);
        datas = boardService.getBoardList(bvo,cnt);
    }else {
        if(req.getParameter("category").equals("btitle")) {
            bvo.setBtitle(req.getParameter("keywords"));
            datas = boardService.getBoardList_title(bvo,cnt);
            recordCount=boardService.getBoardCntTitle(bvo);
        }else if(req.getParameter("category").equals("mnick")) {
            bvo.setMnick(req.getParameter("keywords"));
            datas = boardService.getBoardList_nick(bvo,cnt);
            recordCount=boardService.getBoardCntNick(bvo);
        }
    }
    model.addAttribute("category", req.getParameter("category"));
    model.addAttribute("searchCondition", bvo.getSearchCondition());
    model.addAttribute("keywords", req.getParameter("keywords"));
    model.addAttribute("recordCount",recordCount);
    model.addAttribute("bdatas",datas);
    return "main.jsp";
}
```

컨트롤러에 사용자의 검색 요청이 오게 되면 처음 화면을 띄울 때 1~10에 해당하는 게시글을 띄워야 하는데 페이징 처리된 번호를 누르지 않아 page 값을 받아오지 못하므로 값을 받아오지 못할 때 page 값에 해당하는 cnt를 1로 설정해주고 page 값을 받아올 때는 그 값을 사용할 수 있도록 구현

이 후 검색한 값 없이 정렬만 바꾸고 싶을 때와 검색한 값이 있을 때로 분기가 나뉘고 검색한 값이 있을 때 제목 검색이냐 작성자 검색이냐에 따라 분기가 다시 나뉘어 비즈니스 메서드를 사용

컨트롤러에서 View로 데이터를 넘길 때 검색한 데이터를 유지하기 위해 검색 데이터들을 모두 model에 담고 페이징 코드에 필요한 값도 model에 담아 View로 데이터를 전달

페이징 처리 + 검색 필터링 (3/3)

```
private final String B_SELECTALL_NEW = "select * from (select rownum r,a.* from (select * from board order by bid desc) a) where r between ? and ?"; // 글 목록 최신순으로 불러오기
private final String B_SELECTALL_HITS = "select * from (select rownum r,a.* from (select * from board order by bhits desc) a) where r between ? and ?"; // 글 목록 조회순으로 불러오기
private final String B_SELECTALL_LIKE = "select * from (select rownum r,a.* from (select * from board order by blike desc) a) where r between ? and ?"; // 글 목록 좋아요순으로 불러오기
```

①

②

```
public ArrayList<BoardVO> getBoardList(BoardVO vo,int cnt) {
    conn = JDBCUtil.connect();
    ArrayList<BoardVO> datas = new ArrayList<BoardVO>();
    try {
        if(vo.getSearchCondition()==null) {
            pstmt = conn.prepareStatement(B_SELECTALL_NEW);
        }else {
            if(vo.getSearchCondition().equals("new")) {
                pstmt = conn.prepareStatement(B_SELECTALL_NEW);
            } else if(vo.getSearchCondition().equals("hits")) {
                pstmt = conn.prepareStatement(B_SELECTALL_HITS);
            } else if(vo.getSearchCondition().equals("like")) {
                pstmt = conn.prepareStatement(B_SELECTALL_LIKE);
            }
        }
        pstmt.setInt(1, cnt*10-9);
        pstmt.setInt(2, cnt*10);
        ResultSet rs = pstmt.executeQuery();
        while(rs.next()) {
            BoardVO data = new BoardVO();
            data.setBid(rs.getInt("bid"));
            data.setMid(rs.getString("mid"));
            data.setMnick(rs.getString("mnick"));
            data.setBtitle(rs.getString("btitle"));
            data.setBimg(rs.getString("bimg"));
            data.setBcontent(rs.getString("bcontent"));
            data.setBhits(rs.getInt("bhits"));

            pstmt = conn.prepareStatement(L_SELECTCNT_BLIKE);
            pstmt.setInt(1, rs.getInt("bid"));
            ResultSet rs2 = pstmt.executeQuery();
            rs2.next();
            int blikeCnt = rs2.getInt(1);
            rs2.close();

            pstmt = conn.prepareStatement(B_UPDATE_BLIKECNT);
            pstmt.setInt(1, blikeCnt);
            pstmt.setInt(2, rs.getInt("bid"));
            pstmt.executeUpdate();

            data.setBlike(blikeCnt);
            data.setBdate(rs.getString("bdate"));
            datas.add(data);
        }
        rs.close();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } finally {
        JDBCUtil.disconnect(pstmt, conn);
    }
    return datas;
}
```

1. 페이징 처리를 위해 게시글을 10개씩 끊어서 불러와야 되는데 일반적인 쿼리문으로 rownum 1~10까지의 정보는 불러올 수 있지만 rownum 11~20까지 정보는 불러올 수 없는 문제가 있음

이 문제는 서브쿼리를 중첩으로 사용해 해결할 수 있었는데 a라는 별칭의 임시 테이블 내에 rownum으로 지정한 r칼럼 선택해 between으로 범위를 지정해주면 원하는 데이터를 가져와 페이징 처리를 구현할 수 있게 됨

2. CRUD 중 R에 해당하는 비즈니스 메서드로 필터링 정렬에 따라, 페이지 번호에 따라 10개씩 해당 게시글을 불러오는 로직

메인 페이지에서 게시글 목록을 불러올 때 searchCondition(정렬) 정보가 없어 기본 최신순(new)을 사용하고 View 검색 필터에서 원하는 정렬을 선택하면 받은 searchCondition 값에 따라 최신순(new), 조회순(hits), 좋아요순(like) SQL문을 사용할 수 있도록 분기처리 하였고 이 후 받은 페이징 번호(cnt)에 따라 해당 번호에 해당하는 게시글 10개를 가져올 수 있도록 구현

이 후 로직은 이미지 업로드에서 설명한 좋아요 수 즉각 반영을 위한 로직과 동일하고 제목 검색 메서드와 작성자 검색 메서드는 이와 유사

좋아요 (1/2)

```
<script type="text/javascript">
$("#likebtn").on("click", function() {

    if(${mid} != null) {

        var bid = "${bdata.bid}";
        var mid = "${mid}";

        $.ajax({
            type : "post",
            url : "likeAjax.do",
            data : {
                "bid" : bid,
                "mid" : mid
            },
            dataType : "json",
            success : function(result) {

                if(${ldata} == null){
                    if (result == 1) {
                        $("#likebtn").css('background', '#696cff').css('border-radius', '50%').css('cursor', 'pointer');
                        $("#likecnt").text("좋아요 ${bdata.blike+1}");
                    } else if(result == 0) {
                        $("#likebtn").css('background', 'none').css('cursor', 'pointer');
                        $("#likecnt").text("좋아요 ${bdata.blike}");
                    }
                }
                else{
                    if (result == 1) {
                        $("#likebtn").css('background', '#696cff').css('border-radius', '50%').css('cursor', 'pointer');
                        $("#likecnt").text("좋아요 ${bdata.blike}");
                    } else if(result == 0) {
                        $("#likebtn").css('background', 'none').css('cursor', 'pointer');
                        $("#likecnt").text("좋아요 ${bdata.blike-1}");
                    }
                }
            },
            error : function() {
                console.log('에러 발생');
                console.log(err.status + " | " + err.errText);
            }
        });
    }
    else{
        alert('로그인 후 이용 부탁드립니다.');
```

ajax를 사용해 페이지 넘김 없이
좋아요를 하기 위해 구현한
script 코드

좋아요를 클릭했을 때 session에
저장된 mid 정보가 있는지 없는지
판단해 비로그인 이용자의 경우
좋아요를 하지 못하게 구현하였고
로그인한 상태라면 bid(게시글 PK)
와 mid(아이디 PK)를 컨트롤러로
보내 로직을 처리하고 success
function으로 result가 반환됨

게시글에 들어왔을 때
좋아요 정보가 있냐 없냐에 따라
좋아요를 +1할지 -1할지가 정할 수
있기 때문에 분기로 나누었고
result가 1일 때 좋아요를 한
상태가 되고 0일 때 좋아요를
취소한 상태가 되도록 구현

좋아요 (2/2)

①

```
@ResponseBody
@RequestMapping(value="/view/likeAjax.do")
public String likeAjax(LikeVO lvo) {
    LikeVO data = likeService.getLike(lvo);
    String result;
    if(data==null) {
        likeService.insertLike(lvo);
        result="1";
    }else {
        likeService.deleteLike(lvo);
        result="0";
    }
    return result;
}
```

②

```
// 좋아요 상태 확인용
public LikeVO getLike(LikeVO vo) {
    conn = JDBCUtil.connect();
    LikeVO data = null;
    try {
        pstmt = conn.prepareStatement(L_SELECTONE);
        pstmt.setInt(1, vo.getBid());
        pstmt.setString(2, vo.getMid());
        ResultSet rs = pstmt.executeQuery();
        if(rs.next()) {
            data = new LikeVO();
            data.setBid(rs.getInt("bid"));
            data.setMid(rs.getString("mid"));
        }
        rs.close();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } finally {
        JDBCUtil.disconnect(pstmt, conn);
    }
    return data;
}
```

```
private final String L_INSERT = "insert into blike (bid,mid) values(?,?)"; // 좋아요 등록
private final String L_SELECTONE = "select * from blike where bid=? and mid=?"; // 좋아요 상태 확인용
private final String L_DELETE = "delete blike where bid=? and mid=?"; // 좋아요 해제

// 좋아요 등록
public void insertLike(LikeVO vo) {
    conn = JDBCUtil.connect();
    try {
        pstmt = conn.prepareStatement(L_INSERT);
        pstmt.setInt(1, vo.getBid());
        pstmt.setString(2, vo.getMid());
        pstmt.executeUpdate();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } finally {
        JDBCUtil.disconnect(pstmt, conn);
    }
}

// 좋아요 해제
public void deleteLike(LikeVO vo) {
    conn = JDBCUtil.connect();
    try {
        pstmt = conn.prepareStatement(L_DELETE);
        pstmt.setInt(1, vo.getBid());
        pstmt.setString(2, vo.getMid());
        pstmt.executeUpdate();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } finally {
        JDBCUtil.disconnect(pstmt, conn);
    }
}
```

1. ajax 스크립트에서 컨트롤러로 bid와 mid정보를 넘겨주면 좋아요를 했는지 안했는지 확인하기 위해 비즈니스 메서드(R) getLike()를 사용

blike table에 해당 정보가 없다면 좋아요를 하지 않은 상태여서 받은 bid와 mid를 blike table에 넣어주고 result 값 1을 반환하고 해당 정보가 있다면 blike table에서 삭제한 후 result값 0을 반환

(ajax를 사용할 때 컨트롤러에 @ResponseBody를 설정해야 ajax에서 값을 반환 받을 수 있음)

2. ajax 좋아요 요청에 사용되는 비즈니스 메서드(C, R, D)

모두 bid와 mid를 인자로 받아 메서드가 수행되고 bid나 mid에 PK설정이 없는 이유는 같은 회원이 여러 개의 글에 좋아요를 할 수 있고 반대로 하나의 글에 여러 개의 좋아요가 달릴 수 있기 때문에 blike table은 누가 어느 글에 좋아요를 했다는 것을 확인하기 위한 table이 됨

아이디+닉네임 중복검사 (1/2)

```
<script type="text/javascript">
$("#mid").on("focusout", function() {

    var mid = $(this).val();

    $.ajax({
        type : "post",
        url : "idCheckAjax.do",
        data : {
            "mid" : mid
        },
        dataType : "json",
        success : function(result) {

            if ($("#mid").val() == "") {
                $("#idCheck").text("필수 정보입니다.");
                $("#idCheck").css("color", "black");
            } else {
                if (result == 0) {
                    $("#idCheck").text("사용 가능한 아이디입니다.");
                    $("#idCheck").css("color", "green");
                } else {
                    $("#idCheck").text("사용 불가능한 아이디입니다.");
                    $("#idCheck").css("color", "red");
                    $("#mid").val('');
                }
            }

        },
        error : function() {
            console.log('에러 발생');
            console.log(err.status + " | " + err.errText);
        }
    });
});
</script>
```

회원가입 시 아이디와 닉네임 중복을 피하기 위해 중복검사 로직을 구현하는데 중복검사를 할 때마다 페이지가 바뀌고 입력한 정보가 날라간다면 사용자의 입장에선 불편한 서비스가 되기 때문에 페이지 넘김 없이 입력한 정보를 유지시키기 위해 ajax를 사용해 로직을 구현

아이디 중복검사에 사용되는 ajax 스크립트 코드로 View에서 아이디를 입력하고 focusout이 발생하면 입력한 아이디 정보를 컨트롤러로 보내 로직을 처리하고 success function으로 result 값을 반환 받음

아이디 입력란이 공백일 때와 아닐 때를 분기로 나누고 공백이 아닐 때 반환된 result값이 0이면 사용가능한 아이디라는 문구가 생기고 result값이 0이 아닐 때 사용 불가능한 아이디라는 문구가 생기고 아이디 입력란을 공백으로 비워 구현

아이디+닉네임 중복검사 (2/2)

①

```
@ResponseBody
@RequestMapping(value="/view/idCheckAjax.do")
public String idCheckAjax(MemberVO mvo) {
    MemberVO data = memberService.getMember_idCheck(mvo);
    String result;
    if(data == null){
        result = "0"; // 아이디 만들기 가능
    }else{
        result = "1"; // 아이디 이미 있음
    }
    return result;
}
```

②

```
private final String M_SELECTONE_ID = "select * from member where mid=?"; // 아이디 중복검사, 내 정보 불러오기
// 아이디 중복검사, 내 정보 불러오기
public MemberVO getMember_idCheck(MemberVO vo) {
    conn = JDBCUtil.connect();
    MemberVO data=null;
    try {
        pstmt = conn.prepareStatement(M_SELECTONE_ID);
        pstmt.setString(1, vo.getMid());
        ResultSet rs = pstmt.executeQuery();
        if(rs.next()) {
            data = new MemberVO();
            data.setMid(rs.getString("mid"));
            data.setMpw(rs.getString("mpw"));
            data.setMnick(rs.getString("mnick"));
            data.setMemail(rs.getString("memail"));
        }
        rs.close();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } finally {
        JDBCUtil.disconnect(pstmt, conn);
    }
    return data;
}
```

1. ajax에서 보낸 mid정보를 컨트롤러에서 받아 비즈니스 메서드® getMember_idCheck()를 사용해 받은 mid정보가 member table에 있는지 없는지 확인하고 정보가 없다면 result값 0을 ajax로 반환하고 정보가 있다면 result값 1을 ajax로 반환

2. 아이디 중복검사에 사용되는 비즈니스 메서드®로 인자로 mid를 받아 해당 mid의 회원 정보들을 가져오는 로직

아이디 중복검사 로직과 같은 형태로 닉네임 중복검사도 구현

이메일 인증 (1/2)

```
<script type="text/javascript">
var keyCode = null;
$("#sendEmail").on("click",function() {

    var memail = $("#email").val();

    $.ajax({
        type : "post",
        url : "emailCheckAjax.do",
        data : {
            "memail" : memail
        },
        dataType : "text",
        success : function(result) {

            keyCode = result;
            $("#emailCheck").removeAttr("disabled");
            alert("인증코드를 보냈습니다. 메일을 확인해주세요!");

        },
        error : function(request, status, error) {
            console.log("code:" + request.status + "message:" + request.responseText + "error:" + error);
        }
    });
});

$("#emailCheck").on("focusout", function() {

    var input = $(this).val();

    if ($("#emailCheck").attr("readonly") == "readonly") {
        $("#codeCheck").text("이미 인증을 완료하였습니다.");
    } else {

        if ($("#emailCheck").val() == "") {
            $("#codeCheck").text("인증 필수");
            $("#codeCheck").css("color", "black");
        } else {
            if (input != keyCode) {
                $("#codeCheck").text("인증번호 불일치 재인증 요망");
                $("#codeCheck").css("color", "red");
                $("#emailCheck").val('');
            } else {
                $("#codeCheck").text("인증 성공");
                $("#codeCheck").css("color", "green");
                $("#emailCheck").attr("readonly", "readonly");
                $("#email").attr("readonly", "readonly");
                $("#sendEmail").attr("disabled", "disabled");
            }
        }
    }
});
</script>
```

이메일 인증도 페이지 넘김 없이 구현하기 위해 ajax를 활용

회원가입에서 이메일을 입력하고 인증코드 전송 버튼을 클릭하게 되면 입력한 이메일 값을 컨트롤러로 보내고 success function에서 result 값을 반환 받음

반환 받은 result 값을 전역변수로 설정한 keyCode에 넣고 인증코드 입력란의 disabled 속성 제거 후 메일을 확인하라는 알람창을 띄움

인증코드 입력란에 입력한 값과 keyCode 값이 같지 않을 때 인증번호 불일치라는 문구를 띄운 후 입력란을 비우고 입력한 값과 keyCode 값이 같을 때 인증성공이라는 문구들 띄운 후 이메일 입력란과 인증코드 입력란에 readonly 속성을 걸어 사용자가 인증 후 정보를 변경하지 못하도록 구현

이메일 인증 (2/2)

```
@ResponseBody
@RequestMapping(value="/view/emailCheckAjax.do")
public String emailCheckAjax(MemberVO mvo, RandomCode rc, SendMail sm) {
    String code = rc.randomCode();
    sm.sendMail(mvo.getMemail(), code);
    return code;
}
```

①

②

```
<!-- 난수 생성 -->
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>3.12.0</version>
</dependency>
<!-- mail -->
<dependency>
    <groupId>javax.mail</groupId>
    <artifactId>mail</artifactId>
    <version>1.4.7</version>
</dependency>
<dependency>
    <groupId>javax.activation</groupId>
    <artifactId>activation</artifactId>
    <version>1.1.1</version>
</dependency>
```

③

```
public String randomCode() {
    String code = RandomStringUtils.randomAlphanumeric(5);
    return code;
}
```

⑤ ☆ LOL 커뮤니티 인증코드 발송

보낸사람 VIP <sungkwang0908@naver.com>
받는사람 <sungkwang0908@naver.com>

안녕하세요, LOL 커뮤니티입니다.
인증코드는 [n6xqh] 입니다.

④

```
public void sendMail(String email, String code) {

    String host = "smtp.naver.com"; // 사용할 사이트
    final String user = "sungkwang0908@naver.com"; // ID
    final String password = " "; // PW

    String to = email; // 보낼 이메일 주소

    // Get the session object
    Properties props = new Properties();
    props.put("mail.smtp.starttls.enable", "true");
    props.put("mail.smtp.host", host);
    props.put("mail.smtp.auth", "true");
    props.put("mail.smtp.port", "587");
    props.put("mail.smtp.ssl.protocols", "TLSv1.2");

    Session session = Session.getDefaultInstance(props, new javax.mail.Authenticator() {
        protected PasswordAuthentication getPasswordAuthentication() {
            return new PasswordAuthentication(user, password);
        }
    });

    // Compose the message
    try {
        MimeMessage message = new MimeMessage(session);
        message.setFrom(new InternetAddress(user));
        message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));

        // Subject
        message.setSubject("LOL 커뮤니티 인증코드 발송");

        // Text
        message.setText("안녕하세요, LOL 커뮤니티입니다.\r\n 인증코드는 [" + code + "] 입니다.");

        // send the message
        Transport.send(message);
        System.out.println("이메일 전송 성공!");
        System.out.println(code);

    } catch (MessagingException e) {
        e.printStackTrace();
    }
}
```

1. 컨트롤러는 ajax로부터 이메일 정보를 받고 난수를 생성하는 메서드 randomCode()를 사용하여 난수를 생성한 뒤 받은 이메일 정보와 난수를 메일 전송용 메서드 sendMail()에 담아 메일을 전송하고 난수를 다시 ajax로 반환
2. 난수 생성 로직은 commons-lang3.jar를 사용하고 메일 api는 mail.jar, activation.jar를 사용하기 때문에 pom.xml에 dependency를 추가해준 모습
3. 난수 생성 로직을 메서드화 시켜 필요할 때 객체를 불러와 사용할 수 있도록 구현
4. 메일 api를 사용하기 위해 구글링한 코드로 난수 생성 로직과 마찬가지로 메서드화 시켜 사용하고 인자에 email과 code를 추가해 email과 code 정보를 받아 사용할 수 있도록 구현
5. 메일을 전송했을 때 받은 메일의 모습

크롤링

```
public static void main(String[] args) {
    final String url="https://poro.gg/champions?format=stats";
    Document doc=null;

    try {
        doc=Jsoup.connect(url).get();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    String key = "top"; // 포지션 별 데이터 크롤링
    Elements eles=doc.select("div.champion-sub-list__content[data-role-key="+key+"] div.champion-sub-list__item");
    Iterator<Element> itr=eles.iterator();
    Iterator<Element> itr2=eles.select("img").iterator();
    Iterator<Element> itr3=eles.select("div.rank").iterator();
    Iterator<Element> itr4=eles.select("div.champion").iterator();
    Iterator<Element> itr5=eles.select("div.rate").iterator();

    itr3.next();
    itr4.next();
    itr5.next();
    itr5.next();
    itr5.next();
    itr5.next();

    int i = 0;
    while(itr.hasNext()) {
        i++;

        String tag1 = String.valueOf(itr2.next());
        String[] arr1 = tag1.split("\\");
        String tag2 = String.valueOf(itr2.next());
        String[] arr2 = tag2.split("\\");

        String rank = itr3.next().text();
        String champImg = arr1[1];
        String champ = itr4.next().text();
        String tierImg = arr2[1];
        String winRate = itr5.next().text();
        String pickRate = itr5.next().text();

        itr5.next();

        StatsVO data = new StatsVO();
        data.setPosition(key);
        data.setRank(rank);
        data.setChampsrsrc(champImg);
        data.setSchamp(champ);
        data.setStiersrc(tierImg);
        data.setSwinrate(winRate);
        data.setSpickrate(pickRate);

        StatsDAO statsDAO = new StatsDAO();
        statsDAO.insertStats(data);

        if(i == 10) {
            break;
        }
    }
}
```

원하는 사이트에서
원하는 데이터를 크롤링하기 위해
작성한 코드

사이트를 분석한 결과 원하는
데이터들이 div 속성
data-role-key 값에 따라 바뀌는 걸
확인해서 key라는 변수를 만들고
값을 바꿔가며 크롤링 및 DB에
데이터 저장 진행

원하는 데이터 중 text가 아니라
element에 존재하는 값을
크롤링하기 위해 String.valueOf()를
사용하여 타입을 element에서
String으로 바꿔준 후 split()을
사용하여 (\\)기준으로 나누고
원하는 데이터를 가져와 사용

text를 크롤링할 때 불필요한
정보들은 itr.next()로 넘기고
원하는 데이터만 수집

최종적으로 원하는 데이터를
가져온 뒤 StatsVO에 담아
비즈니스 메서드(C) insertStats()를
사용하여 stats table에
데이터 저장

소감 및 추후 개선사항

- 소감

프로젝트를 진행하면서 항상 느끼는 생각이지만 설계가 부족했다는 생각이 들었다. 한 번도 구현해보지 못했던 로직의 경우 생각만으로 완벽히 설계하는 것이 어려웠고 그로 인해 코드를 작성하고 시연할 때마다 에러사항을 해결하고 설계를 수정하고 기존에 작성했던 코드들을 고치는 일이 잦았다.

설계할 때 충분한 시간을 가지고 꼼꼼하게 체크하면서 설계해야 한다는 걸 알고는 있지만 직접 부딪히고 나서 깨닫게 되는 경우가 많아 설계에 있어 항상 아쉬움이 남는 거 같다.

팀프로젝트에서 담당파트를 나눠 작업하던 것을 개인 프로젝트에서는 전부 다루다 보니 MVC 패턴의 전체흐름을 보다 잘 이해할 수 있게 되었고 각 파트별로 놓치는 부분이 있을 때 어떠한 에러사항이 생길지 컴파일을 해보지 않고, 로그를 남기지 않아도 어느 정도 짐작할 수 있어 발생하는 여러 에러에 대해 능숙하게 대처할 수 있게 되었다.

- 추후 개선사항

불필요한 로직, 코드에 대해 지속적으로 리팩토링 하고 유지보수에 용이한 코드가 될 수 있도록 비즈니스 메서드의 형태를 JdbcTemplate, Mybatis를 사용하는 형태로 바꿔주는 것이 목표이다. 기존에 구현한 비즈니스 메서드는 하나의 메서드에서 여러 SQL문을 사용하는 형태로 구현했기 때문에 유지보수가 힘들고 가독성도 떨어지는 결합도가 높은 코드여서 사용하는 SQL문 하나당 하나의 메서드로 모듈화 시켜주고 컨트롤러에서 모듈화 시킨 메서드들을 사용해 원래 구상했던 로직을 사용할 수 있게 구현해준 뒤 유지보수가 편리한 결합도가 낮고 응집도가 높은 코드가 될 수 있도록 JdbcTemplate이나 Mybatis를 활용해 비즈니스 메서드를 버전업할 예정이다.