

NAGNEO.GG - 리그오브레전드 전적 검색

기간: 10/26 - 11/02

이름: 임호균 이우람

목차

개요

주요기능

일자별 구현기능

테이블명세서

ERD

주요기능에 대한 설명

구현영상 및 깃허브

후기

개요

E-SPORTS 시장의 종주국이라 불리는 한국에서도 점유율 1위인 리그오브레전드의 전적 검색기능과 챔피언 정보를 제공하는 사이트

사용기술



RIOT API 10.22.1



Spring RELEASE 3.9.11



Eclipse 2019-09 4.13.0



Java SE 1.8.0



Oracle 11g 11.2.0

주요기능

잭슨을 이용한 라이엇 API 파싱을 이용한 전적 검색

자바스크립트를 이용한 회원가입 유효성검사

세션을 이용한 로그인 상태유지

쿠키를 이용한 검색기록 남기기

해쉬맵과 세션을 이용한 검색한 내용 저장

페이징을 응용한 전적 더보기

라이엇 API 파싱을 이용한 챔피언 정보

일자별 구현 기능

10월 26일

- 데이터베이스 구축 및 마이바티스를 이용한 컨넥션 연결
- 기본적인 JSP 뷰 구현
- 기본적인 컨트롤러, 인터셉터, 서비스단 구현

10월 27일

- 클라이언트가 검색한 소환사의 기본적인 정보 파싱
- 검색한 정보를 출력할 JSP 구현 및 파싱한 데이터와 매치되는 이미지 파싱
- API 파싱을 위한 VO 구현

10월 28일

- API를 이용한 소환사 대전 기록 파싱
- 쿠키를 이용한 검색기록 남기기 구현
- 속도 향상을 위한 챔피언 정보 DB 테이블 구현 및 정보 입력

10월 29일

- 파싱을 하는 속도 향상을 위한 최적화 작업 실시
- 자세한 전적 정보를 보기 위한 디테일 작업
- 더 많은 전적 정보를 보기 위한 페이징 작업

일자별 구현 기능

10월 30일

- 챔피언의 기본적인 정보를 입력한 게시판 생성
- 대전 기록 내부의 모든 정보와 챔피언 정보 연동
- 챔피언 정보를 위한 API JSON 파싱 작업 실시

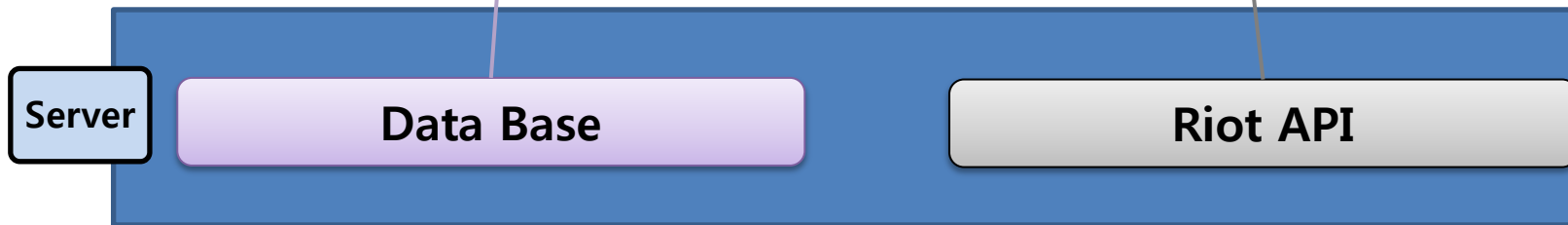
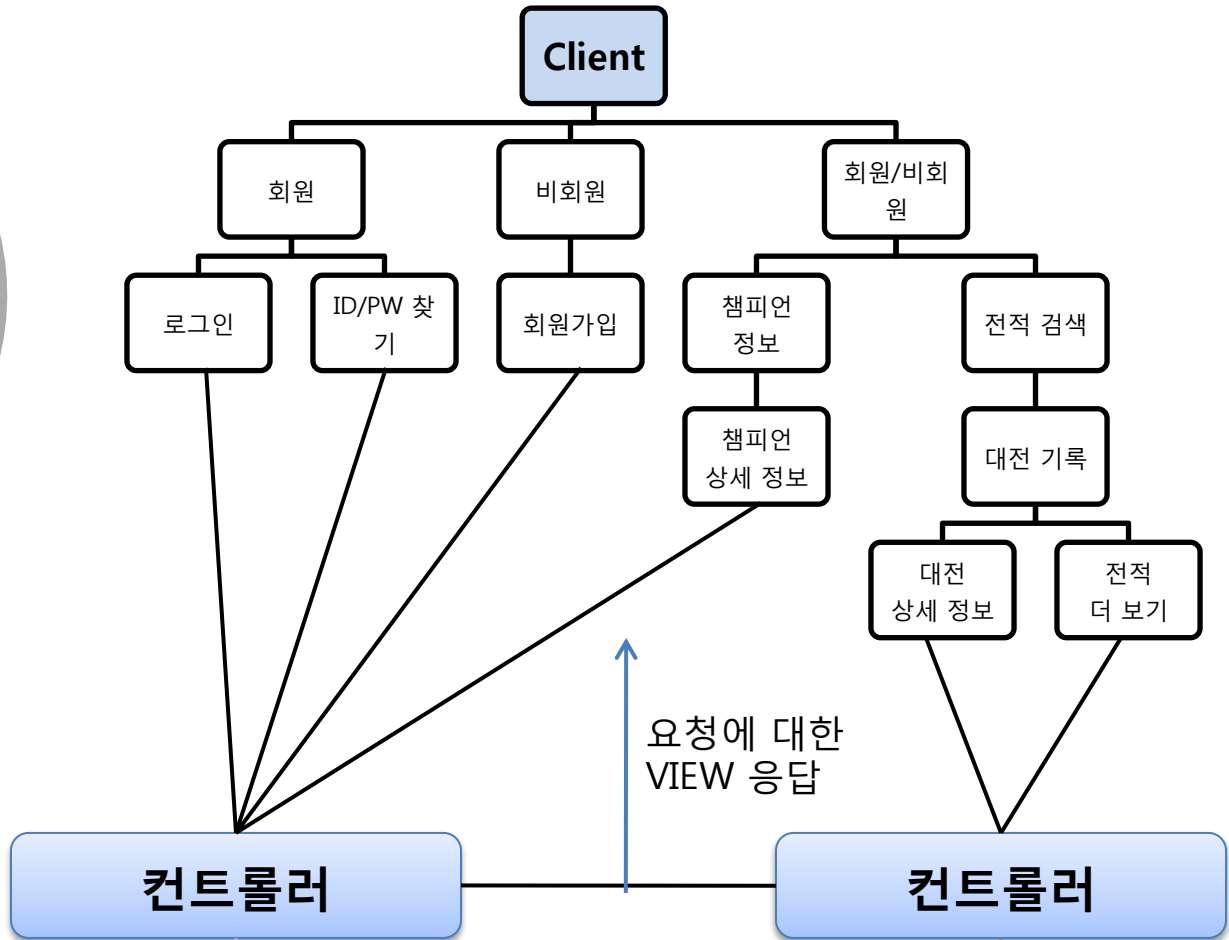
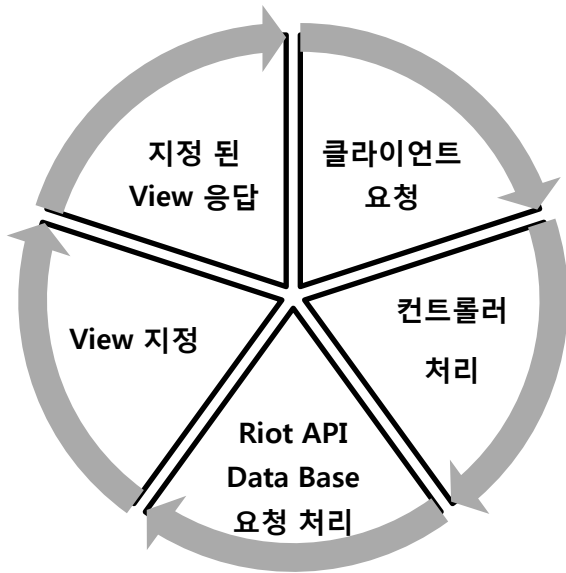
10월 31일

- JSP 최적화 작업
- JSON을 이용하여 받아온 챔피언 DB 테이블 입력 및 추가적인 테이블 구축
- 속도 향상을 위해 HASHMAP을 이용하여 이전 검색된 정보 저장

11월 2일

- 이전 정보를 남기기 위한 마지막 갱신 시간 추가
- 현재까지의 소스 최적화 마무리 작업
- PPT 만들기

흐름도



테이블 명세서

Table : spell

No	컬럼명	타입	NULL	KEY
1	key	number	NO	Primary key
2	engid	varchar2	NO	

Table : rune

No	컬럼명	타입	NULL	KEY
1	key	number	NO	Primary key
2	engid	varchar2	NO	

Table : user

No	컬럼명	타입	NULL	KEY
1	no	number	NO	Primary key
2	id	varchar2	NO	
3	pw	varchar2	NO	
4	phone	varchar2	NO	
5	nickname	Varchar2	NO	

Table : champion

No	컬럼명	타입	NULL	KEY
1	key	number	NO	Primary key
2	engid	varchar2	NO	
3	korid	varchar2	NO	
4	title	varchar2	NO	

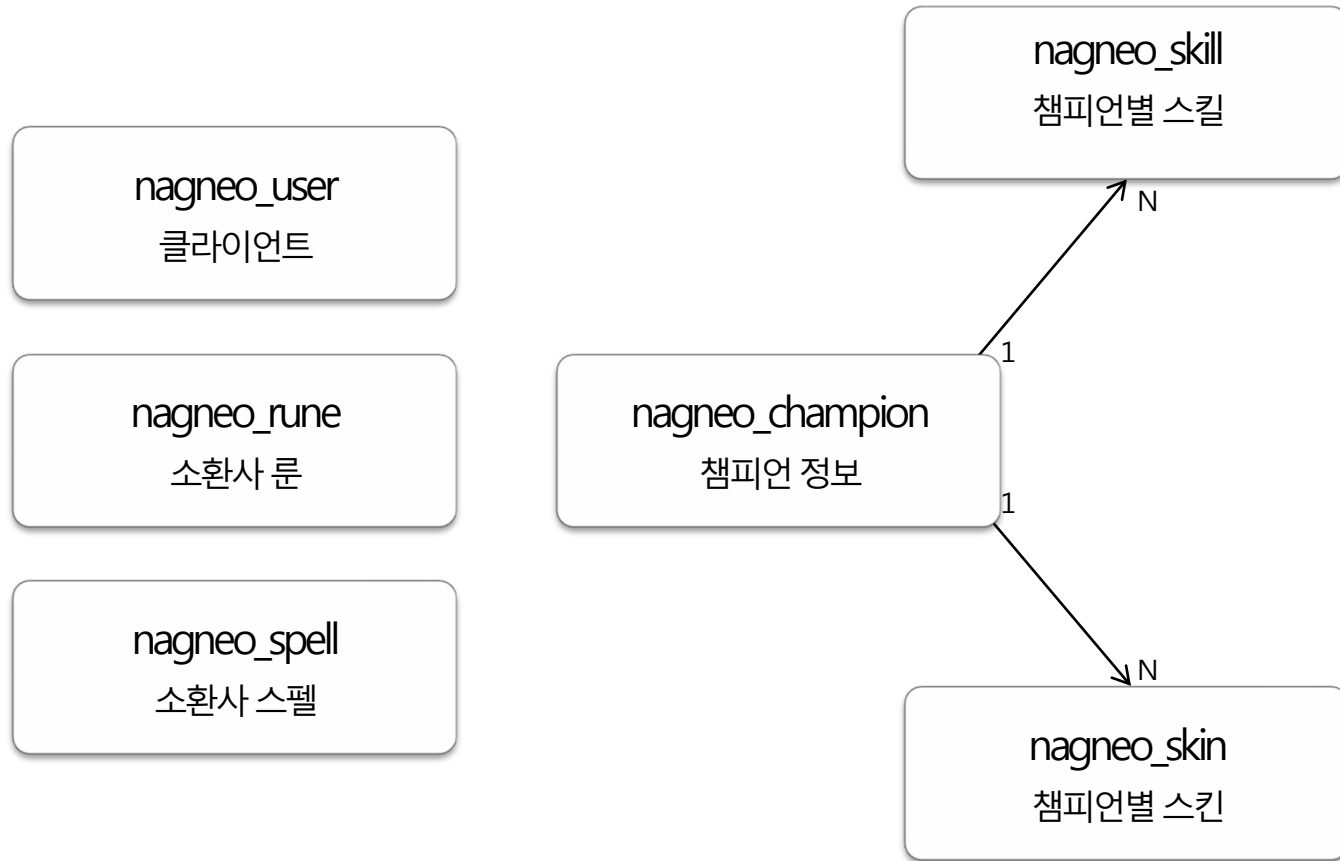
Table : skill

No	컬럼명	타입	NULL	KEY
1	key	number	NO	Foreign key
2	no	number	NO	
3	engid	varchar2	NO	
4	korid	varchar2	NO	
5	description	Varchar2	NO	

Table : skin

No	컬럼명	타입	NULL	KEY
1	key	number	NO	Foreign key
2	num	number	NO	
3	korid	varchar2	NO	

ERD 다이어그램



주요기능에 대한 설명

API란? 개발자가 개발을 하기 위해 필요한 정보를 제공해주는 정보로써 일반적으로
알 수 없는 내부적인 정보를 제공해주는 서비스


파싱(정보 추출)을 위해서는 JSON의 기본적인 구조 파악이 필수
대다수의 API는 JSON을 기반으로 구성되어 있음
제공되는 정보에 맞는 VO를 생성하여 JSON과 VO를 매핑하는 과정으로 정보를 추출

```
ObjectMapper om = new ObjectMapper(); API파싱을 한 뒤에 제이슨을 추출하기 위한 라이브러리(잭슨 Jackson)
HttpClient hc = HttpClientBuilder.create().build(); API를 파싱하기 위한 빌더 생성
HttpGet hg = new HttpGet(url + name + apiKey); 파싱할 서버의 URL 주소 입력
HttpResponse hr = hc.execute(hg); 해당되는 URL에서 데이터 추출
if (hr.getStatusLine().getStatusCode() == 200) {
    ResponseHandler<String> h = new BasicResponseHandler();
    String body = h.handleResponse(hr); 위에 선언한 핸들러를 이용하여 JSON 데이터를 스트링 형태로 파싱
    om.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES, false);
    sVO = om.readValue(body, SummonerVO.class); 해당되는 VO에 잭슨을 이용하여 매핑
```

주요기능에 대한 설명

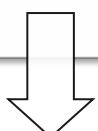
페이징을 응용하여 5개의 전적씩 API 파싱, 해당되는 정보를 기존 정보에 추가하여 클라이언트 전송

```
ArrayList<Long> arrayKey = league.getMatchesData(sVO.getAccountId(), String.valueOf(index),  
    String.valueOf(Integer.valueOf(index) - 5));  
List<MatchVO> mList = league.getMatchData(arrayKey, Integer.valueOf(index) - 5);  
ArrayList<SearchUserVO> arrayTitle = league.getTitleList(mList, sVO.getName(), Integer.valueOf(index) - 5);
```



로그인 시도 시에 DB 접속 해당되는 정보가 있을 시에 세션에 등록 후에, 메인 화면 전환,
아닐 경우에는 다시 로그인 화면으로 이동

```
if (u.selectOne(uVO)) {  
    session.setAttribute("logChk", "로그아웃");  
    session.setAttribute("login", uVO);  
    return "index";  
}  
return "redirect:/login";
```



주요기능에 대한 설명

쿠키를 이용하여 정보가 있는 소환사의 기록을 쿠키로 저장,

인터셉터를 이용하여 최근 5개의 검색기록 표시



```
try {
    if (!sVO.getName().equals(" 존재하지않는소환사")) {
        Cookie cookie = new Cookie(URLEncoder.encode(String.valueOf(request.getCookies().length + 1), "UTF-8"),
            URLEncoder.encode(sVO.getName() + "," + String.valueOf(sVO.getProfileIconId()), "UTF-8"));
        response.addCookie(cookie);
    }
}

@Override
public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
    throws Exception {
    // TODO Auto-generated method stub
    if(request.getCookies() != null) {
        HttpSession session = request.getSession();
        HashMap<String, String> log = new HashMap<String, String>(5);
        String string = "";
        for (int i = request.getCookies().length - 1; i >= 0; i--) {
            String[] temp = URLDecoder.decode(request.getCookies()[i].getValue(), "UTF-8").split(",");
            if(temp.length > 1 && !temp[0].equals(string)) {
                string = temp[0];
                log.put(temp[0], temp[1]);
            }
            if(log.size() == 5) break;
        }
        session.setAttribute("log", log);
    }
    return true;
}
```

주요기능에 대한 설명

검색한 기록의 정보를 해쉬맵을 이용하여 저장, 후에 같은 기록일 경우에는
인터셉터에서 해쉬맵에 있던 데이터를 전송(검색 속도 향상)

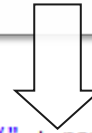


```
@Override
public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
    throws Exception {
    // TODO Auto-generated method stub
    HashMap<String, TotalListVO> saveList;
    if (request.getSession().getAttribute("saveList") != null) {
        saveList = (HashMap<String, TotalListVO>) request.getSession().getAttribute("saveList");
    } else {
        return true;
    }

    if (request.getParameter("action").equals("reset")) {
        saveList.get(request.getParameter("name")).reset();
        saveList.get(request.getParameter("name")).setLastSearch();
        return true;
    } else if (saveList.get(request.getParameter("name")) != null) {
        request.getSession().setAttribute("tlVO", saveList.get(request.getParameter("name")));
        request.getSession().setAttribute("index", saveList.get(request.getParameter("name")).getIndex());
        saveList.get(request.getParameter("name")).setLastSearch();
        response.sendRedirect(request.getContextPath() + "/list");
        return false;
    } else {
        return true;
    }
}
```

주요기능에 대한 설명

챔피언 각각의 정보를 추출하여 DB에 저장하기 위해
라이언트 API의 JSON을 잭슨을 이용하여 파싱



```
String url = "http://ddragon.leagueoflegends.com/cdn/10.22.1/data/ko_KR/champion/" + name + ".json";
try {
    ObjectMapper om = new ObjectMapper();
    HttpClient hc = HttpClientBuilder.create().build();
    HttpGet hg = new HttpGet(url);
    HttpResponse hr = hc.execute(hg);
    if (hr.getStatusLine().getStatusCode() == 200) {
        String body = EntityUtils.toString(hr.getEntity(), "UTF-8");
        om.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES, false);
        ChampionVersionVO cvVO = new ChampionVersionVO();
        cvVO = om.readValue(body, ChampionVersionVO.class);
        LinkedHashMap<String, Object> vo = (LinkedHashMap<String, Object>) cvVO.getData().get(name);

        ArrayList<Object> skins = null;
        ArrayList<Object> spells = null;
        LinkedHashMap<String, Object> passive = null;
        LinkedHashMap<String, Object> lhm = null;

        ChampionVO cVO = new ChampionVO();
        ChampionSkillVO cslVO = null;
        ChampionSkinVO cskVO = null;
        ArrayList<ChampionSkillVO> skill = new ArrayList<ChampionSkillVO>();
        ArrayList<ChampionSkinVO> skin = new ArrayList<ChampionSkinVO>();
    }
}
```

구현영상 및 깃허브



프로그래밍은 맛있다. 카페를 이용해주세요
[구현영상](#)



[깃허브 주소](#)

후기

처음으로 팀 프로젝트를 진행하며, 상호간의 소통이 얼마나 중요한 지 깨달을 수 있었고, 웹 프로젝트의 어려움 역시 느꼈다.

하지만 어려움만큼의 만족도가 있었고, 그만큼의 배움 역시 있었다. 프로젝트를 진행하며 내가 모르던 기능에 대한 배움이 있었고, 그만큼의 지식이 넓어짐을 느낌으로써 팀 프로젝트의 장점과 단점을 느끼게 된 좋은 경험이었다.