

# 로 데이터 분석 웹사이트

# INDEX

1

프로젝트 역할 및 기여도

2

프로젝트 개요 및 소개

3

프로젝트 구성 및 수행

4

프로젝트 수행 소감

# 1 프로젝트 역할 및 기여도

---

## 프로젝트 역할 및 기여도

### 역할

- 프론트엔드
- 백엔드 설계
- CSS
- 챔피언 분석 구현(빌드,아이템)
- 데이터베이스 관리
- 파이썬 데이터 수집 및 전처리
- 데이터 전처리 모듈 구현

### 기여도



## 2 프로젝트 개요 및 소개

---

## 롤 데이터 분석을 선택한 이유 ?



1.

### PC방 점유율 1위

- 19년 기준 동시접속자 800만명
- 16년도에 세계 최초로 월 접속자수 1억명 달성

2.

### E-SPORT 활성화

- LCK, 롤드컵, 아시안 게임 등 다양한 리그 및 대회 존재
- 2020년 롤드컵 최고 시청자 수 4400만명 달성
- LCK 활성화로 40대 인구 유입

3.

### 다양한 분석사이트 존재

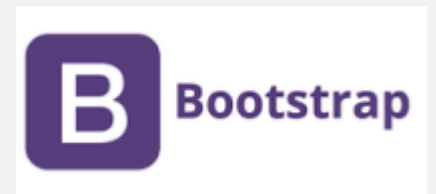
- 160개가 넘는 챔피언과 다양한 룬과 아이템
- 상대 챔피언에 따라 선택할 수 있는 전략이 다양함
- 각각의 사이트마다 고유의 특징과 장단점 존재

# 개발 환경

## 데이터 수집 & 전처리



## 서버 및 프론트



# 3 프로젝트 구성 및 수행

---



# 데이터 수집

## 모듈

```
def get_rawData(tier):
    division_list=['I', 'III', 'III', 'IV']
    p=random.randrange(1,100)
    lst=[]
    for division in division_list:
        url='https://kr.api.riotgames.com/lol/league/v4/entries/RANKED_SOLO_5x5/'+tier+'/'+division+'?page='+str(p)+'&api_key='+lol_api_key
        res=requests.get(url).json()
        lst+=sample(res,3)

    print('get SummonerName.....')
    summonerName_lst=list(map(lambda x:x['summonerName'],lst))

    print('get puuid.....')
    puuid_lst=[]
    for n in tqdm(summonerName_lst):
        try:
            puuid_lst.append(get_puuid(n))
        except:
            print(n)
            continue

    print('get match_id.....')
    matchid_lst=[]
    for p in tqdm(puuid_lst):
        matchid_lst.extend(get_matchid(p,3))

    print('get matches & timeline....')
    match_timeline_lst=get_matches_timelines(matchid_lst)

    df=pd.DataFrame(match_timeline_lst, columns=['gameId','matches','timeline'])
    print('complete!')

    return df
```

- 티어별로 데이터를 구분해서 RAW DATA를 수집
- RIOT-API에서 제공하는 데이터를 JSON 형태로  
변환 후에 데이터 프레임으로 변환
- MATCH & TIMELINE 데이터 수집

# 데이터 전처리

## 모듈

```
def get_match_timeline_df(df):
    url='http://ddragon.leagueoflegends.com/cdn/12.15.1/data/ko_KR/summoner.json'
    spell_res=requests.get(url).json()
    spell_dic={spell_res['data'][i]['key']:i for i in spell_res['data'].keys()}
    print('match_df 생성중...')
    df_creator = []

    for i in tqdm(range(len(df))): # 클래식 게임 중 버전 12.13, 12.14 데이터만, 커스텀 제외
        if df.iloc[i].matches['info']['gameMode']=='CLASSIC' and df.iloc[i].matches['info']['queueId']!=830 and \
            ((df.iloc[i].matches['info']['gameVersion']).startswith('12.13')) or (df.iloc[i].matches['info']['gameVersion']).startswith('12.14')):
            g_id = df.iloc[i].gameId
            match = df.iloc[i].matches['info']
            timeline = df.iloc[i].timeline['info']

            start = localtime(match['gameStartTimestamp']/1000) # 게임 시작한 날짜, 시간
            end = localtime(match['gameEndTimestamp']/1000) # 게임 끝난 날짜, 시간
            duration=localtime(match['gameDuration']) # 게임 플레이 시간

            for j in range(10):
                tmp = []
                tmp.append(g_id)
                tmp.append(str(start.tm_year)+'년 '+str(start.tm_mon)+'월 '+str(start.tm_mday)+'일 '+\
                    str(start.tm_hour)+'시 '+str(start.tm_min)+'분')
                tmp.append(str(end.tm_year)+'년 '+str(end.tm_mon)+'월 '+str(end.tm_mday)+'일 '+str(end.tm_hour)+'시 '+str(end.tm_min)+'분')
                tmp.append(str(duration.tm_min)+'분 '+str(duration.tm_sec)+'초')
                tmp.append(match['gameDuration'])
                tmp.append(match['gameVersion'])
                tmp.append(match['participants'][j]['summonerName'])
                tmp.append(match['participants'][j]['summonerLevel'])
                tmp.append(match['participants'][j]['participantId'])
                tmp.append(match['participants'][j]['championName'])
                tmp.append(match['participants'][j]['championId'])
                tmp.append(match['participants'][j]['teamPosition'])
                tmp.append(match['participants'][j]['teamId'])

            # 밴픽
            if len(match['teams'][0]['bans'])>0:
                if j<5:
                    tmp.append(match['teams'][0]['bans'][j]['championId'])
                else:
                    tmp.append(match['teams'][1]['bans'][j-5]['championId'])
            else:
                tmp.append(0)

            tmp.append(match['participants'][j]['win'])
            tmp.append(match['participants'][j]['kills'])
            tmp.append(match['participants'][j]['deaths'])
            tmp.append(match['participants'][j]['assists'])

            blue_kill=match['teams'][0]['objectives']['champion']['kills']
            red_kill=match['teams'][1]['objectives']['champion']['kills']
```

```
# 킬관여율
if j<5:
    hap=match['participants'][j]['kills']+match['participants'][j]['assists']
    tmp.append(round((hap/blue_kill)*100,2))
else:
    hap=match['participants'][j]['kills']+match['participants'][j]['assists']
    tmp.append(round((hap/red_kill)*100,2))

tmp.append(match['participants'][j]['totalDamageDealtToChampions'])
tmp.append(match['participants'][j]['totalDamageTaken'])

# object data, 타워, 억제기
if j<5:
    # 블루팀
    tmp.append(match['teams'][0]['objectives']['tower']['first']) # 포블 True, False
    tmp.append(match['teams'][0]['objectives']['tower']['kills']) # 타워 파괴
    tmp.append(match['teams'][0]['objectives']['inhibitor']['kills']) # 억제기 파괴

    tmp.append(match['teams'][0]['objectives']['riftHerald']['first']) # 전령
    tmp.append(match['teams'][0]['objectives']['riftHerald']['kills'])
    tmp.append(match['teams'][0]['objectives']['baron']['first']) # 바론
    tmp.append(match['teams'][0]['objectives']['baron']['kills'])
    tmp.append(match['teams'][0]['objectives']['dragon']['first']) # 용
    tmp.append(match['teams'][0]['objectives']['dragon']['kills'])
    tmp.append(match['participants'][0]['challenges']['teamElderDragonKills']) # 장로
else:
    # 레드팀
    tmp.append(match['teams'][1]['objectives']['tower']['first'])
    tmp.append(match['teams'][1]['objectives']['tower']['kills'])
    tmp.append(match['teams'][1]['objectives']['inhibitor']['kills'])

    tmp.append(match['teams'][1]['objectives']['riftHerald']['first'])
    tmp.append(match['teams'][1]['objectives']['riftHerald']['kills'])
    tmp.append(match['teams'][1]['objectives']['baron']['first'])
    tmp.append(match['teams'][1]['objectives']['baron']['kills'])
    tmp.append(match['teams'][1]['objectives']['dragon']['first'])
    tmp.append(match['teams'][1]['objectives']['dragon']['kills'])
    tmp.append(match['participants'][5]['challenges']['teamElderDragonKills'])
```

# 데이터 전처리

## 모듈

```
# 5분~35분 골드 data(컬럼 g5_35)
total_gold_list=list(map(lambda x: str(x['participantFrames'][str(j+1)]['totalGold']), timeline['frames']))[5:36]
tmp.append(''.join(total_gold_list))

#15분 골드
try:
    g15=timeline['frames'][15]['participantFrames'][str(j+1)]['totalGold']
except:
    g15=0

tmp.append(g15)
tmp.append(match['participants'][j]['goldEarned']) # 최종 골드

# cs
tmp.append(match['participants'][j]['totalMinionsKilled']+match['participants'][j]['neutralMinionsKilled'])

# 분당 cs
total_cs=match['participants'][j]['totalMinionsKilled']+match['participants'][j]['neutralMinionsKilled']
tmp.append(round(total_cs/(match['gameDuration']/60),2))

# 멀티킬 data
tmp.append(match['participants'][j]['doubleKills'])
tmp.append(match['participants'][j]['tripleKills'])
tmp.append(match['participants'][j]['quadraKills'])
tmp.append(match['participants'][j]['pentaKills'])

# 스펠 data (컬럼 spell1, spell2)
spell_key1 = match['participants'][j]['summoner1Id']
spell_key2 = match['participants'][j]['summoner2Id']

tmp.append(spell_dic[str(spell_key1)])
tmp.append(spell_dic[str(spell_key2)])
```

```
# item & item 구매시간 data
item_dic={}

item=list(map(lambda x: list(filter(lambda z: z['type']=='ITEM_PURCHASED' , x['events'])), timeline['frames']))
item=[element for array in item for element in array]

for n in range(len(item)):
    for slot in range(7):
        if item[n]['participantId']==j+1 and item[n]['itemId']==match['participants'][j]['item'+str(slot)]:
            item_dic[str(match['participants'][j]['item'+str(slot)])]=str(item[n]['timestamp'])

        elif match['participants'][j]['item'+str(slot)] in up_item_dic#
            and up_item_dic[match['participants'][j]['item'+str(slot)]]==item[n]['itemId']:
                # 최종아이템 id가 up_item_dic의 key값이면서 최종 아이템의 하위템 id와 item[j]['itemId']값이 같으면
                item_dic[str(match['participants'][j]['item'+str(slot)])]=str(item[n]['timestamp'])

items=''.join(item_dic.keys())
item_time=''.join(item_dic.values())

tmp.append(items)
tmp.append(item_time)
tmp.append(match['participants'][j]['visionWardsBoughtInGame']) # 평와
tmp.append(match['participants'][j]['wardsKilled'])
tmp.append(match['participants'][j]['wardsPlaced'])

df_creator.append(tmp)
columns = ['gameId', 'startGameDate', 'endGameDate', 'duration', 'gameDuration', 'gameVersion', 'summonerName', 'summonerLevel',
'participantId', 'championName', 'championId', 'teamPosition', 'teamId', 'bans', 'win', 'kills', 'deaths', 'assists',
'killParticipation', 'totalDamageDealtToChampions', 'totalDamageTaken', 'firstTower', 'towerKills', 'inhibitorKills',
'firstRiftHeraldKills', 'riftHeraldKills', 'firstBaron', 'baronKills', 'firstDragon', 'dragonKills', 'teamElderDragonKills',
'g5_35', 'g15', 'goldEarned', 'cs', 'perMinute_cs', 'doubleKills', 'tripleKills', 'quadraKills', 'pentaKills', 'spell1', 'spell2',
'main_rune', 'main_under1', 'main_under2', 'main_under3', 'main_under4', 'sub_rune', 'sub_under1', 'sub_under2', 'statPerks',
'champLevel', 'skill_build', 'start_item', 'items', 'item_times', 'visionWardsBoughtInGame', 'wardsKilled', 'wardsPlaced']

df = pd.DataFrame(df_creator, columns=columns)
print('complete! 현재 df의 수는 %d입니다' % len(df))
if len(df)==0:
    print(df_creator)
return df
```

## 챔피언 분석

### 실행 화면



- home.jsp에서 챔피언분석 카테고리 클릭
- DB에서 검색해온 값과 함께 championHome.jsp로 forwarding
- RelationPage : championHome.jsp

### Controller

```
@GetMapping(value = "/championHome")
public ModelAndView championDetail() {
    String tier = "platinum";
    String lane = "TOP";
    mav = champmm.getChampionInfo(tier, lane);
    return mav;
}
```

- championHome.jsp에서 라인별 챔피언 순위를 출력할 default값으로 tier와 lane을 정해준다
- ServiceClass인 ChampionDetailMM 클래스의 getChampionInfo(tier, lane) 함수를 실행
- ModelAndView를 return

### Service

### getChampionInfo

```
public ModelAndView getChampionInfo(String tier, String lane) {  
  
    mav = new ModelAndView();  
  
    // 옵션에 띄울 티어/ 티어 색깔  
    mav.addObject("tier", tier);  
  
    // 챔피언 사진에 들어갈 value들  
    List<Champion> nameIdList = champDao.getChampionList();  
    mav.addObject("nameIdList", makechampList(nameIdList));  
  
    // 챔피언티어 정보에 들어갈 value들  
    List<Champion> tierList = champDao.getTierList(tier, lane);  
    mav.addObject("tierList", makeTierList(tierList));  
    mav.setViewName("Detail/championHome");  
  
    return mav;  
}
```

- tier값을 ModelAndView에 담는다
- Dao인 IchampionDao interface의 getChampionList()와 getTierList()를 실행
- 결과를 파라미터로 makeChampionList(nameList)와 makeTierList(tierList)를 실행
- championHome을 setViewName한 후 mav return

## 챔피언 분석

### Dao

```
List<Champion> getChampionList();  
List<Champion> getTierList(@Param("tier") String tier, @Param("lane") String lane);
```

### Mapper

```
<select id="getChampionList" resultType="Champion">  
    SELECT /*+INDEX_DESC(STATS_SKILL_DATA STATS_SKILL_PK)*/  
    CHAMPIONNAME, CHAMPIONID, CHAMPION_KR_NAME  
    FROM STATS_SKILL_DATA  
    ORDER BY CHAMPION_KR_NAME  
</select>
```

interface로 받았던 파라미터를 통해 챔피언이름,  
챔피언아이디, 챔피언한글이름을 챔피언 한글을  
이름 순으로 정렬해서 검색(왼쪽 리스트)

```
<select id="getTierList" resultType="Champion">  
    SELECT /*+INDEX_DESC(${tier}_DETAIL ${tier}_DETAIL_PK)*/ CHAMPIONNAME,  
    CHAMPIONID, CHAMPION_KR_NAME, CHAMPTIER, WINRATE AS WINRATE1, PICKRATE,  
    BANRATE AS BANRATE1, COUNTER1, COUNTER2, COUNTER3  
    FROM ${tier}_DETAIL  
    WHERE LANE = #{lane}  
    ORDER BY CHAMPTIER, WINRATE1  
</select>
```

interface로 받았던 파라미터를 통해 챔피언이름,  
챔피언아이디, 챔피언한글이름, 챔피언티어, 승률,  
픽률, 밴률, 카운터1,2,3을 챔피언티어와 승률 순  
으로 정렬해서 검색(오른쪽 리스트)

### Service

#### makechampList

```
private String makechampList(List<Champion> nameIdList) {  
    StringBuilder sb = new StringBuilder();  
  
    for (int i = 0; i < nameIdList.size(); i++) {  
        sb.append("<div class = 'champion' data-championId = '" + nameIdList.get(i).getChampionId() + "'>");  
        sb.append("<img class = 'listing' src = https://ddragon.leagueoflegends.com/cdn/12.16.1/img/champion/"  
            + nameIdList.get(i).getChampionName() + ".png>");  
        sb.append("<br><small class = 'championName'>" + nameIdList.get(i).getChampion_kr_name() + "</small>");  
        sb.append("</div>");  
    }  
  
    return sb.toString();  
}
```

Dao로 검색해온 값들을 이용해 jsp에 출력할 태그를 String 형태로 생성



## 챔피언 분석 – 오른쪽 리스트

Service

makeTierList

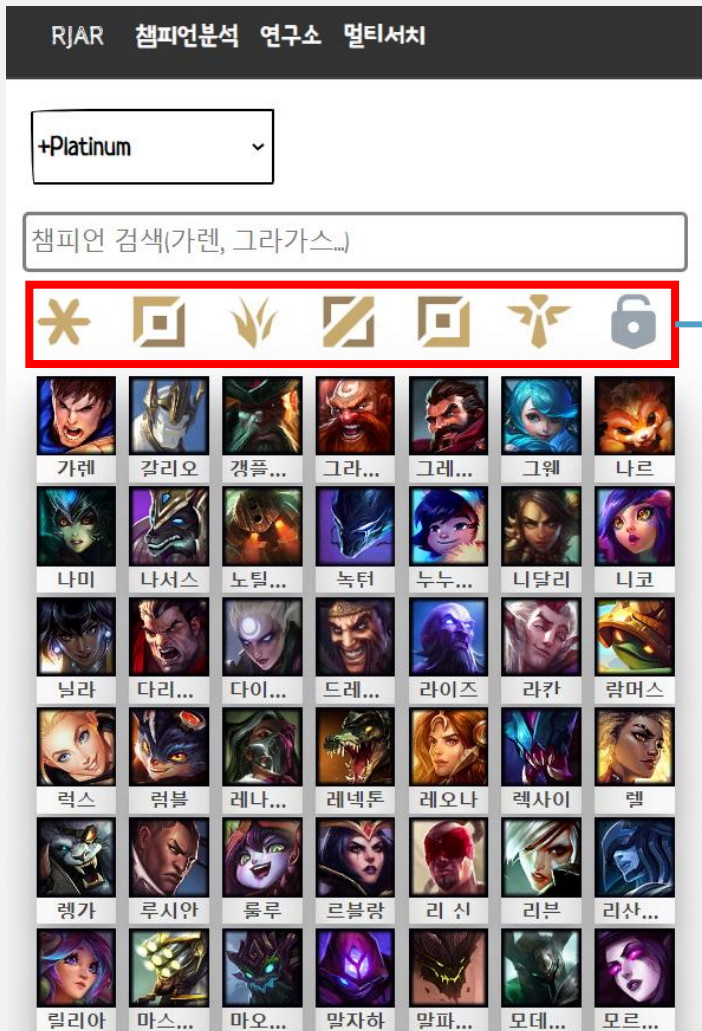
```
private String makeTierList(List<Champion> tierList) {  
  
    StringBuilder sb = new StringBuilder();  
  
    sb.append("<tr class = 'jool'>");  
    sb.append("<th>순위</th>");  
    sb.append("<th colspan='2' style='padding-left: 45px;'>챔피언</th>");  
    sb.append("<th>타여</th>");  
    sb.append("<th>승률</th>");  
    sb.append("<th>픽률</th>");  
    sb.append("<th>밴률</th>");  
    sb.append("<th>상대하기 어려운 챔피언</th>");  
    sb.append("</tr>");  
  
    for (int i = 0; i < tierList.size(); i++) {  
  
        sb.append("<tr class = 'jool'>");  
  
        sb.append("<td style = 'width:50px;'><font style = 'text-align: center; text-weight: bold;'>" + (i + 1)  
            + "</font></td>");  
  
        sb.append("<td>");  
        sb.append("<div class = 'tierChamp' data-championId = '" + tierList.get(i).getChampionId() + "'>");  
        sb.append("<img class = 'tierChampionimg' src = https://ddragon.leagueoflegends.com/cdn/12.16.1/img/champion/"  
            + tierList.get(i).getChampionName() + ".png>");  
        sb.append("</div>");  
        sb.append("</td>");  
  
        sb.append("<td class = 'kr_name'><small style = 'font-weight: bolder'"  
            + tierList.get(i).getChampion_kr_name() + "</small></td>");  
  
        sb.append("<td><font class = 'tier'>" + tierList.get(i).getChampTier() + "</font></td>");  
        sb.append("<td class = 'rate'><font>" + tierList.get(i).getWinRate1() + "</font></td>");  
        sb.append("<td class = 'rate'><font>" + tierList.get(i).getBanRate1() + "</font></td>");  
        sb.append("<td class = 'rate'><font>" + tierList.get(i).getPickRate() + "</font></td>");  
  
        sb.append("<td>");  
        sb.append("<div class = 'counter' value = " + tierList.get(i).getCounter1() + ">");  
        sb.append("<img class = 'counterimg' src = https://ddragon.leagueoflegends.com/cdn/12.16.1/img/champion/"  
            + tierList.get(i).getCounter1() + ".png>");  
        sb.append("</div>");  
  
        sb.append("<div class = 'counter' value = " + tierList.get(i).getCounter2() + ">");  
        sb.append("<img class = 'counterimg' src = https://ddragon.leagueoflegends.com/cdn/12.16.1/img/champion/"  
            + tierList.get(i).getCounter2() + ".png>");  
        sb.append("</div>");  
  
        sb.append("<div class = 'counter' value = " + tierList.get(i).getCounter3() + ">");  
        sb.append("<img class = 'counterimg' src = https://ddragon.leagueoflegends.com/cdn/12.16.1/img/champion/"  
            + tierList.get(i).getCounter3() + ".png>");  
        sb.append("</div>");  
        sb.append("</td>");  
  
        sb.append("</tr>");  
  
    }  
  
    return sb.toString();  
}
```

Dao로 검색해온 값들을 이용해 jsp에 출력  
할 태그를 String 형태로 생성



## 챔피언 분석 – 왼쪽 리스트


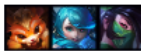

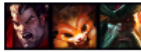

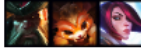

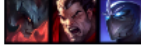

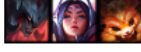

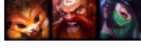

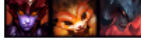

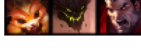

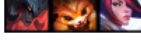

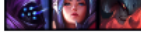

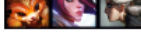


### 실행 화면



- 처음엔 이름 순으로 정렬된 챔피언 리스트 보여줌
- 라인 버튼 클릭 시 해당 라인의 챔피언 리스트 보여줌
- 해당 챔피언의 초상화 클릭시 상세 페이지 이동

## 챔피언 분석 – 오른쪽 리스트

### 실행 화면

TOP		JUNGLE		MIDDLE		BOTTOM		SUPPORTER		
순위	챔피언	티어	승률	픽률	밴률	상대하기 어려운 챔피언				
1		자크	1	62.58	0.86	1.35				
2		탐 켄치	1	65.1	0.71	1.24				
3		쉬바나	2	53.27	4.64	5.95				
4		스카너	2	53.33	0.09	0.62				
5		볼리베어	2	53.72	1.05	2.46				
6		스웨인	2	53.85	0.85	0.86				
7		세주아니	2	54.2	0.44	2.86				
8		사일러스	2	54.57	7.94	2.9				
9		퀸	2	54.6	0.34	1.44				
10		가렌	2	55.06	0.8	2.95				
11		오른	2	55.6	0.15	4.0				
12		웁웁	2	55.69	0.23	1.39				

- 탑의 챔피언 순위를 보여줌
- 라인 버튼 클릭 시 해당 라인의 챔피언 순위를 보여줌
- 해당 챔피언의 초상화 클릭시 상세 페이지 이동

## 챔피언 분석 - 라인 선택 (왼쪽 리스트)

### Controller

```
@GetMapping(value = "/laneImg", produces="text/plain;charset=utf-8")
public String restChampionList(Champion champ) {
    String championList = champmm.restChampionList(champ);
    return championList;
}
```

- Champion Bean으로 파라미터를 받는다.
- ServiceClass인 ChampionDetailMM 클래스의 restChampionList(champ) 함수 실행

### Service

```
public String restChampionList(Champion champ) {
    String lane = champ.getLane();
    String tier = champ.getTier();
    String all = "all";

    List<Champion> restChampionList = null;

    if (lane.equals(all)) {
        restChampionList = champDao.getChampionList();
    } else {
        restChampionList = champDao.getLaneChamp(lane, tier);
    }

    String championList = makechampList(restChampionList);
    return championList;
}
```

- 파라미터를 변수에 저장 후 문자열 "all"과 비교
- 파라미터로 넘어온 lane값이 "all"일 경우  
getChampionList()를 실행
- 그렇지 않을 경우 getLaneChamp(lane, tier)를 실행
- 둘 중 하나를 실행한 결과로  
makechampList(restChampionList)를 실행
- 실행결과 return

## 챔피언 분석 - 라인 선택 (왼쪽 리스트)

### Dao & Mapper

```
List<Champion> getChampionList();
```

```
<select id="getChampionList" resultType="Champion">
    SELECT /*+INDEX_DESC(STATS_SKILL_DATA STATS_SKILL_PK)*/
    CHAMPIONNAME, CHAMPIONID, CHAMPION_KR_NAME
    FROM STATS_SKILL_DATA
    ORDER BY CHAMPION_KR_NAME
</select>
```

선택한 이미지가 "all"일 경우

- 해당 라인(혹은 모든 라인)의 챔피언이름, 챔피언아이디, 챔피언한글이름 검색
- 챔피언한글이름 순으로 정렬

### Dao & Mapper

```
List<Champion> getLaneChamp(@Param("lane")String lane, @Param("tier")String tier);
```

```
<select id="getLaneChamp" resultType="Champion">
    SELECT /*+INDEX_DESC(${tier}_DETAIL ${tier}_DETAIL_PK)*/
    CHAMPIONNAME, CHAMPIONID, CHAMPION_KR_NAME
    FROM ${tier}_DETAIL
    WHERE LANE = #{lane}
    ORDER BY CHAMPION_KR_NAME
</select>
```

특정 라인인 경우

## 챔피언 분석 - 라인 선택 (왼쪽 리스트 로테이션)

### Front

```
$('#free').click(function () {  
    $.ajax({  
        type : 'get',  
        url : 'rotation',  
    }).done( function(data) {  
        console.log('성공');  
        $('#champList').html(data);  
        $('#tier').text  
    }).fail(function(err) {  
        console.log("에러");  
        console.log(err);  
    })  
});
```

- championHome.jsp에서 로테이션 이미지 클릭
- ajax(비동기통신)를 통해 금주의 로테이션 챔피언 정보를 검색 출력
- 검색해온 정보로 championHome.jsp의 해당 위치에 출력
- RelationPage : championHome.jsp

### Controller

```
@GetMapping(value = "/rotation", produces="text/plain;charset=utf-8")  
public String getRotationChamp() {  
    String rotationChamp = champmm.getRotationChamp();  
    return rotationChamp;  
}
```

getRotationChamp() 함수 실행

## 챔피언 분석 – 라인 선택 (왼쪽 리스트 로테이션)

### Service

```
public String getRotationChamp() {
    String url = "https://kr.api.riotgames.com/lol/platform/v3/champion-rotations?api_key=RGAPI-5abbd2a5-6403-43ab-a67b-bdc1c426bcac";

    RestTemplate restTemplate = new RestTemplate();
    String apiResult = restTemplate.getForObject(url, String.class);

    // 받은 내용 json으로 파싱
    JsonParser parser = new JsonParser();
    JsonObject jsonObj = (JsonObject) parser.parse(apiResult);

    // 원하는 value값 꺼내고 그걸 다시 String 배열로 파싱
    Gson gson = new Gson();
    String[] freeChampionList = gson.fromJson(jsonObj.get("freeChampionIds"), String[].class);

    // 챔피언 아이디로 검색한 후 리스트에 담음
    int[] freeList = new int[16];
    for (int i = 0; i < freeChampionList.length; i++) {
        freeList[i] = Integer.parseInt(freeChampionList[i]);
    }
    List<Champion> rotationChampion = champDao.getRotation(freeList[0], freeList[1], freeList[2], freeList[3],
        freeList[4], freeList[5], freeList[6], freeList[7], freeList[8], freeList[9], freeList[10],
        freeList[11], freeList[12], freeList[13], freeList[14], freeList[15]);
    // 리스트를 통해 태그 생성
    String rotationImg = makeChampList(rotationChampion);

    return rotationImg;
}
```

- RestTemplate 클래스의 인스턴스를 이용해 Riot API의 내용을 String 형태로 서버에 받아온다.
- 받아온 내용을 JsonParser클래스의 인스턴스를 이용해 Json 형태로 parsing
- Gson라이브러리를 이용해 Json형태의 내용을 String 배열 형태로 다시 parsing 한다.
- 검색 속도의 향상을 위해 배열에 있는 모든 요소를 한번에 파라미터로 넘겨 getRotation()함수를 실행
- 실행결과로 makeChampList(rotationChampion)을 실행하여 String형태의 태그 생성

## 챔피언 분석 – 라인 선택 (왼쪽 리스트 로테이션)

### Dao

```
List<Champion> getRotaion(@Param("championId1") int championId1, @Param("championId2") int championId2, @Param("championId3") int championId3, @Param("championId4") int championId4,
    @Param("championId5") int championId5, @Param("championId6") int championId6, @Param("championId7") int championId7, @Param("championId8") int championId8,
    @Param("championId9") int championId9, @Param("championId10") int championId10, @Param("championId11") int championId11, @Param("championId12") int championId12,
    @Param("championId13") int championId13, @Param("championId14") int championId14, @Param("championId15") int championId15, @Param("championId16") int championId16);
```

### Mapper

```
<select id="getRotaion" resultType="Champion">
    SELECT /*+INDEX_DESC(STATS_SKILL_DATA STATS_SKILL_PK)*/ CHAMPIONNAME, CHAMPIONID, CHAMPION_KR_NAME
    FROM STATS_SKILL_DATA
    WHERE CHAMPIONID IN({championId1},{championId2},{championId3},{championId4},{championId5},{championId6},
        {championId7},{championId8},{championId9},{championId10},{championId11},{championId12},{championId13},{championId14},
        {championId15},{championId16})
    ORDER BY CHAMPION_KR_NAME
</select>
```

받은 파라미터들을 WHERE절 조건으로 걸어 챔피언이름, 챔피언아이디,  
챔피언 한글이름을 검색한다.



## 챔피언 분석 - 라인 선택 (오른쪽 리스트)

### Front

```
$('.lane_').click(function () {
    let tier = $('#selectOption').val();
    let lane = $(this).val();

    $.ajax({
        type : 'get',
        url : 'tierList',
        data : {tier: tier, lane: lane},

    }).done( function(data) {
        console.log(data);
        $('#tierList').html(data).trigger("create");

    }).fail(function(err) {
        console.log("에러");
        console.log(err);

    })

});
```

- championHome.jsp에서 라인 버튼 클릭
- ajax(비동기통신)를 통해 해당 라인에 대한 정보로 다시 검색 후 출력
- RelationPage : championHome.jsp
- 티어 정보와 라인 정보를 변수에 담아 해당 url로 보낸다.
- 검색해온 정보로 championHome.jsp에서 해당 위치에 출력



## 챔피언 분석 - 라인 선택 (오른쪽 리스트)

### Controller

```
@GetMapping(value = "/tierList" , produces="text/plain;charset=utf-8")
public String restTierList(Champion champ) {
    String tierList = champmm.restTierList(champ);
    return tierList;
}
```

- Champion Bean으로 파라미터를 받는다.
- ServiceClass인 ChampionDetailMM 클래스의 restTierList(champ) 함수를 실행

### Service

```
public String restTierList(Champion champ) {
    List<Champion> restTierList = champDao.getTierList(champ.getTier(), champ.getLane());
    String tierList = makeTierList(restTierList);
    return tierList;
}
```

- 파라미터로 받은 tier와 lane정보로 getChampionList() 실행
- makeTierList(restTierList) 실행

## 챔피언 분석 - 라인 선택 (오른쪽 리스트)

### Dao

```
List<Champion> getTierList(@Param("tier") String tier, @Param("lane") String lane);
```

### Mapper

```
<select id="getTierList" resultType="Champion">
  SELECT /*+INDEX_DESC(${tier}_DETAIL ${tier}_DETAIL_PK)*/ CHAMPIONNAME,
  CHAMPIONID, CHAMPION_KR_NAME, CHAMPTIER, WINRATE AS WINRATE1, PICKRATE,
  BANRATE AS BANRATE1, COUNTER1, COUNTER2, COUNTER3
  FROM ${tier}_DETAIL
  WHERE LANE = #{lane}
  ORDER BY CHAMPTIER, WINRATE1
</select>
```

- interface로 받았던 파라미터를 통해 챔피언이름, 챔피언아이디,  
챔피언한글이름, 챔피언티어, 승률, 픽률, 밴률, 카운터1,2,3을  
챔피언티어와 승률 순으로 정렬해서 검색



## 실행 와면



- RelationPage : championDetail.jsp

## Controller

```
@GetMapping(value = "/clickDetail")
public ModelAndView clickDetail(Champion champ) {
    String tier = "platinum";
    mav = champmm.clickDetail(champ.getChampionId(), tier);
    return mav;
}
```

- championDetail.jsp에서 챔피언 상세정보를 출력할 default값으로 tier를 정해준다
- ServiceClass인 ChampionDetailMM 클래스의 clickDetail(championId, tier) 함수를 실행
- ModelAndView return

## 챔피언 분석 - 상세페이지 (빌드 탭: 룬)

### Service

```
// 해당 챔피언 룬 (승률이 가장 높은 2개) 가져오기
List<ChampionDetail> championRunes = champDao.getChampionRunes(champion_eg_name, lane1, tier);

// 승률 1번째로 높은 룬
ChampionDetail runes1 = championRunes.get(0);

// 보조 능력치 쪼개기
String statperks = championRunes.get(0).getStatperks();
StringTokenizer st1 = new StringTokenizer(statperks, "|");

// 룬 이미지 태그 만들어서 가져오기
List<ChampionDetail> mainRunePng = selectRunes(runes1.getMain_rune());
List<ChampionDetail> subRunePng = selectRunes(runes1.getSub_rune());

mav.addObject("runes1", runes1);
mav.addObject("mainRunePng", makeIngTag(mainRunePng, runes1));
mav.addObject("subRunePng", makeIngTag(subRunePng, runes1));
mav.addObject("statperks1", st1.nextToken());
mav.addObject("statperks2", st1.nextToken());
mav.addObject("statperks3", st1.nextToken());

// 승률 2번째로 높은 룬
ChampionDetail runes2 = null;
for (int i = 1; i < championRunes.size(); i++) {
    if (championRunes.get(0).getSub_rune() != championRunes.get(i).getSub_rune()) {
        runes2 = championRunes.get(i);
        break;
    } else if (championRunes.get(0).getMain_rune() != championRunes.get(i).getMain_rune()) {
        runes2 = championRunes.get(i);
        break;
    }
}

// 보조 능력치 쪼개기
String statperks2 = championRunes.get(1).getStatperks();
StringTokenizer st2 = new StringTokenizer(statperks2, "|");

// 룬 이미지 가져오기
List<ChampionDetail> mainRunePng2 = selectRunes(runes2.getMain_rune());
List<ChampionDetail> subRunePng2 = selectRunes(runes2.getSub_rune());
```

```
// 승률 가장 높은 2가지 룬에 따른 게임 수/승률 검색
List<ChampionDetail> rune_pickWin = new ArrayList<>();
for (int i = 0; i < 2; i++) {
    int main_rune = championRunes.get(i).getMain_rune();
    int main_under1 = championRunes.get(i).getMain_under1();
    int main_under2 = championRunes.get(i).getMain_under2();
    int main_under3 = championRunes.get(i).getMain_under3();
    int main_under4 = championRunes.get(i).getMain_under4();
    int sub_rune = championRunes.get(i).getSub_rune();
    int sub_under1 = championRunes.get(i).getSub_under1();
    int sub_under2 = championRunes.get(i).getSub_under2();
    String statperks3 = championRunes.get(i).getStatperks();

    rune_pickWin.add(champDao.rune_pickWin(main_rune, main_under1, main_under2, main_under3, main_under4,
        sub_rune, sub_under1, sub_under2, statperks3, champion_eg_name, lane1, tier));
}

mav.addObject("rune_win1", rune_pickWin.get(0).getRune_winRate());
mav.addObject("rune_win2", rune_pickWin.get(1).getRune_winRate());
mav.addObject("rune_pick1", rune_pickWin.get(0).getRune_pick());
mav.addObject("rune_pick2", rune_pickWin.get(1).getRune_pick());
mav.addObject("tier", tier);
```

- 승률이 가장 높은 룬 2가지 검색
- 가장 승률이 높은 룬의 보조능력치를 StringTokenizer()을 이용해 3가지로 쪼갬다.
- 승률이 가장 높은 2가지 룬을 selectRunes(메인룬, 서브룬)을 통해 선택한 룬 뿐 아니라 관련 룬을 모두 검색
- 검색해온 룬 데이터들을 makeIngTag()을 통해 태그를 만들고 String형태로 반환 받아서 ModelAndView에 저장
- 검색한 해당 2가지 룬의 승률 및 게임 수 역시 티어정보와 함께 ModelAndView에 저장

## 챔피언 분석 - 상세페이지 (빌드 탭: 룬)

### Service

```
private List<ChampionDetail> selectRunes(int rune) {  
  
    List<ChampionDetail> runePngList = null;  
  
    switch (rune) {  
  
        case 8000: // 정밀  
            String query1 = "SELECT RUNES_ID, RUNES_ICON FROM RUNES WHERE RUNES_ICON LIKE '%Precision%' "  
                + "OR RUNES_ID = 8299";  
            runePngList = champDao.getRunePng(query1);  
            break;  
  
        case 8100: // 지배  
            String query2 = "SELECT RUNES_ID, RUNES_ICON FROM RUNES WHERE RUNES_ICON LIKE '%Domination%'";  
            runePngList = champDao.getRunePng(query2);  
            break;  
  
        case 8200: // 마법  
            String query3 = "SELECT RUNES_ID, RUNES_ICON FROM RUNES WHERE RUNES_ICON LIKE '%Sorcery%' "  
                + "AND RUNES_ID != 8299 AND RUNES_ID != 8242";  
            runePngList = champDao.getRunePng(query3);  
            break;  
  
        case 8300: // 영감  
            String query4 = "SELECT RUNES_ID, RUNES_ICON FROM RUNES WHERE RUNES_ICON LIKE '%Inspiration%' "  
                + "OR RUNES_ID IN (8300, 8410)";  
            runePngList = champDao.getRunePng(query4);  
            break;  
  
        case 8400: // 결의  
            String query5 = "SELECT RUNES_ID, RUNES_ICON FROM RUNES WHERE RUNES_ICON LIKE '%Resolve%' "  
                + "AND RUNES_ID != 8410 OR RUNES_ID = 8242";  
            runePngList = champDao.getRunePng(query5);  
            break;  
    }  
  
    return runePngList;  
}
```

파라미터로 받은 룬 데이터를 기준으로 switch문을 사용하여 해당 룬 데이터와 관련된 룬을 모두 검색한 후 list 형태로 반환한다.

## 챔피언 분석 - 상세페이지 (빌드 탭: 룬)

### Dao

```
List<ChampionDetail> getChampionRunes(@Param("championName") String champion_eg_name, @Param("lane") String lane1, @Param("tier") String tier);

ChampionDetail rune_pickWin(@Param("main_rune")int main_rune, @Param("main_under1")int main_under1, @Param("main_under2")int main_under2,
    @Param("main_under3")int main_under3, @Param("main_under4")int main_under4, @Param("sub_rune")int sub_rune, @Param("sub_under1")int sub_under1,
    @Param("sub_under2")int sub_under2, @Param("statperks3")String statperks3, @Param("championName")String championName, @Param("lane")String lane,
    @Param("tier")String tier);
```

### Mapper

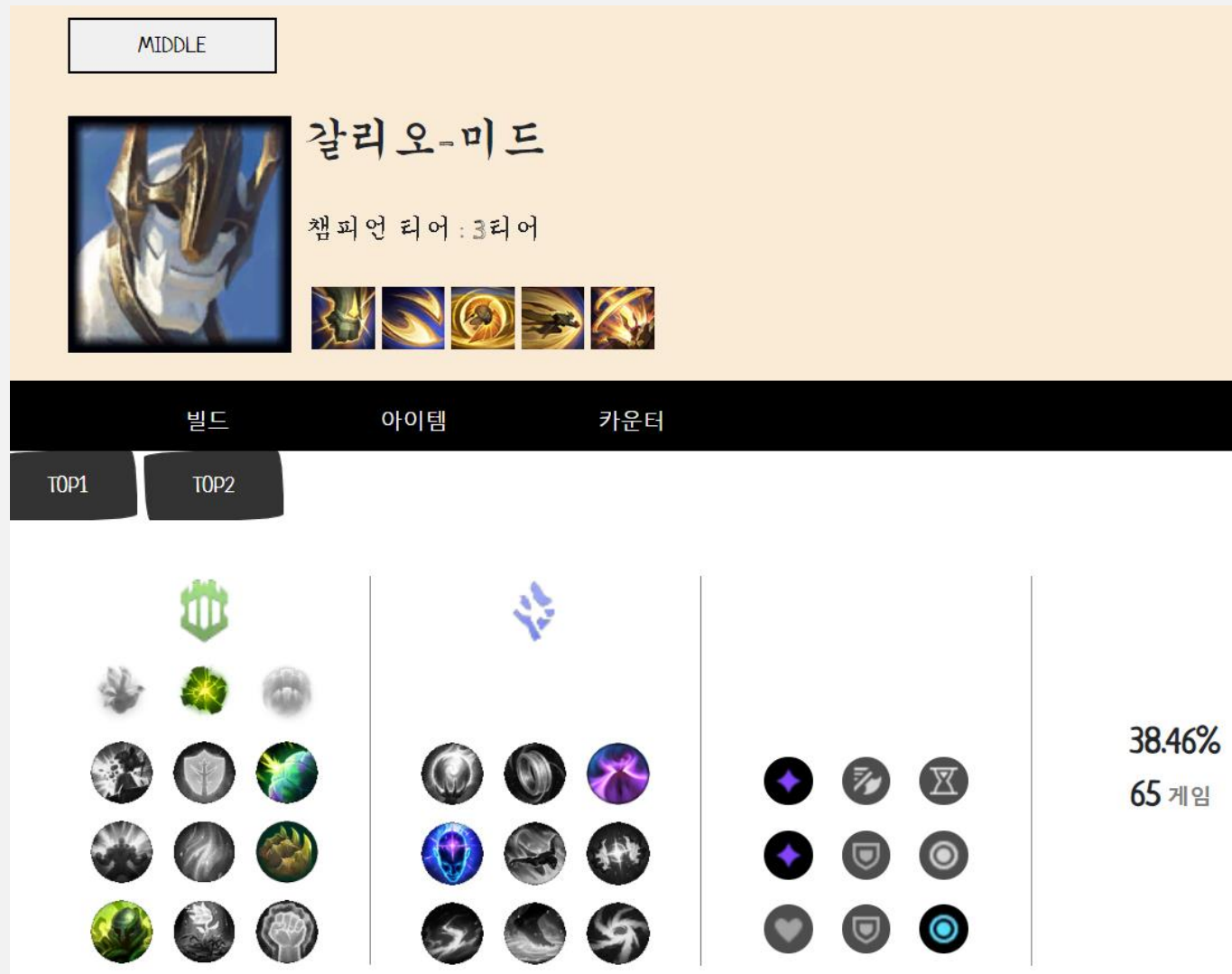
```
<select id="getChampionRunes" resultType="ChampionDetail">
    SELECT /*+ FULL(${tier}) */ MAIN_RUNE, MAIN_UNDER1, MAIN_UNDER2, MAIN_UNDER3,
    MAIN_UNDER4, SUB_RUNE, SUB_UNDER1, SUB_UNDER2, STATPERKS
    FROM ${tier}
    WHERE CHAMPIONNAME = #{championName} AND TEAMPOSITION = #{lane} AND WIN = 'True'
    GROUP BY MAIN_RUNE, MAIN_UNDER1, MAIN_UNDER2, MAIN_UNDER3,
    MAIN_UNDER4, SUB_RUNE, SUB_UNDER1, SUB_UNDER2, STATPERKS
    ORDER BY COUNT(MAIN_RUNE) DESC, COUNT(MAIN_UNDER1) DESC, COUNT(MAIN_UNDER2) DESC, COUNT(MAIN_UNDER3) DESC,
    COUNT(MAIN_UNDER4) DESC, COUNT(SUB_RUNE) DESC, COUNT(SUB_UNDER1) DESC, COUNT(SUB_UNDER2) DESC, COUNT(STATPERKS) DESC
</select>
```

```
<select id="rune_pickWin" resultType="ChampionDetail">
    SELECT B.RUNE_PICK AS RUNE_PICK, ROUND(A.RUNE_WIN/B.RUNE_PICK, 4)*100 AS RUNE_WINRATE
    FROM
    (SELECT CHAMPIONNAME, COUNT(*) AS RUNE_WIN FROM ${tier}
    WHERE MAIN_RUNE = ${main_rune} AND MAIN_UNDER1=${main_under1} AND MAIN_UNDER2=${main_under2} AND MAIN_UNDER3=${main_under3}
    AND MAIN_UNDER4=${main_under4} AND SUB_RUNE =${sub_rune} AND SUB_UNDER1=${sub_under1} AND SUB_UNDER2=${sub_under2}
    AND STATPERKS= #{statperks3} AND CHAMPIONNAME = #{championName} AND TEAMPOSITION = #{lane} AND WIN = 'True'
    GROUP BY CHAMPIONNAME) A
    RIGHT JOIN
    (SELECT CHAMPIONNAME, COUNT(*) AS RUNE_PICK FROM ${tier}
    WHERE MAIN_RUNE = ${main_rune} AND MAIN_UNDER1=${main_under1} AND MAIN_UNDER2=${main_under2} AND MAIN_UNDER3=${main_under3}
    AND MAIN_UNDER4=${main_under4} AND SUB_RUNE =${sub_rune} AND SUB_UNDER1=${sub_under1} AND SUB_UNDER2=${sub_under2}
    AND STATPERKS= #{statperks3} AND CHAMPIONNAME = #{championName} AND TEAMPOSITION = #{lane}
    GROUP BY CHAMPIONNAME) B
    ON A.CHAMPIONNAME = B.CHAMPIONNAME
</select>
```

- 파라미터로 사용할 라인, 챔피언아이디, 챔피언이름, 챔피언 한글 이름을 검색
- 해당 챔피언의 승률이 가장 높은 룬 2가지를 검색
- 룬의 승률과 게임 수를 검색

## 챔피언 분석 - 상세페이지 (빌드 탭: 룬)

### 실행 화면



- 상세페이지 첫 화면은 빌드 탭
- 선택 한 챔피언이 들고 간 룬 빌드
- 픽률, 게임 횟수 나타냄



## 챔피언 분석 - 상세페이지 (빌드 탭: 소환사 주문)

### Service

```
// 스텝 편수, 승률, 픽률
List<ChampionDetail> spells = champDao.getSpell(champion_eg_name, lane1);
ChampionDetail spell;

int cnt=0;
for(int i=0;i<spells.size();i++) {
    for(int j=i;j<spells.size();j++) {
        if (spells.get(i).getSpell1().equals(spells.get(j).getSpell2()) && spells.get(i).getSpell2().equals(spells.get(j).getSpell1())) {
            int spell_cnt = spells.get(i).getSpell_cnt()+spells.get(j).getSpell_cnt();
            int spell_win = spells.get(i).getSpell_win()+spells.get(j).getSpell_win();
            double winrate=((double)spell_win/spell_cnt)*100;
            double pickrate=((double)spell_cnt/spells.get(i).getSpell_total()*100;

            // 소숫점 두자리
            winrate=Math.round(winrate*100)/100.0;
            String spell_winrate=String.format("%.2f", winrate);

            pickrate=Math.round(pickrate*100)/100.0;
            String spell_pickrate=String.format("%.2f", pickrate);

            cnt++;
        }
    }
}
```

- 게임횟수가 가장 많은 스텝 검색
- 스텝이 위치한 순서에 따라 같은 스텝일지라도 다른 데이터로 인식하기 때문에 if문으로 비교
- 같은 값은 게임 횟수와 승수 합산, 승률, 픽률 다시 계산한 값을 set으로 값 교체
- 소숫점 두자리까지 나타내기 위해 String.format 사용
- TOP2 스텝만 가져올 것이기 때문에 cnt가 3이면 break

```
if (cnt==1) {
    spell=spells.get(j);
    spells.set(i, spell.setSpell1(spell.getSpell1()));
    spells.set(i, spell.setSpell2(spell.getSpell2()));
    spells.set(i, spell.setSpell_cnt(spell_cnt));
    spells.set(i, spell.setSpell_win(spell_win));
    spells.set(i, spell.setSpell_winrate(spell_winrate));
    spells.set(i, spell.setSpell_pick(spell_pickrate));
    spells.set(i, spell.setLane(ChampionLane(spell.getLane())));
    mav.addObject("spell", spell);
}
else if(cnt==2) {
    spell=spells.get(j);
    spells.set(i, spell.setSpell1(spell.getSpell1()));
    spells.set(i, spell.setSpell2(spell.getSpell2()));
    spells.set(i, spell.setSpell_cnt(spell_cnt));
    spells.set(i, spell.setSpell_win(spell_win));
    spells.set(i, spell.setSpell_winrate(spell_winrate));
    spells.set(i, spell.setSpell_pick(spell_pickrate));
    mav.addObject("spell2", spell);
}
if (cnt==3) break;
```



## 챔피언 분석 - 상세페이지 (빌드 탭: 소환사 주문)

### Dao

```
List<ChampionDetail> getSpell(@Param("championName") String champion_eg_name, @Param("lane") String lane1);
```

### Mapper

```
<select id="getSpell" resultType="ChampionDetail">
  SELECT * FROM BRONZE_PLUS
  WHERE CHAMPIONNAME=#{championName} AND LANE=#{lane}
  ORDER BY SPELL_CNT DESC
</select>
```

- interface로 받았던 파라미터를 통해 테이블의 모든 컬럼을 가져와서 게임횟수가 가장 높은 순으로 정렬

## 챔피언 분석 - 상세페이지 (빌드 탭: 시작아이템, 신발)

### Service

```
//아이템 판수, 승률, 픽률
List<ChampionDetail> start_items = champDao.getStart_items(champion_eg_name, lane1);
ChampionDetail start1=start_items.get(0);

mav.addObject("start1",start1);

if (start_items.get(1).getStart1()!=null){
    ChampionDetail start2=start_items.get(1);
    mav.addObject("start2",start2);
}

//신발 판수, 승률, 픽률
List<ChampionDetail> boots = champDao.getBoots(champion_eg_name, lane1);
ChampionDetail boots1=boots.get(0);
ChampionDetail boots2=boots.get(1);

mav.addObject("boots1",boots1);
System.out.println("boots1="+boots1);

mav.addObject("boots2",boots2);
System.out.println("boots2="+boots2);
```

- Dao인 IchampionDao interface의  
getStart\_items()와 getBoots()를 실행
- 게임횟수가 가장 많은 시작 아이템, 신발 검색
- TOP2까지만 가져옴
- 시작 아이템과 신발을 ModelAndView에 담는다

## 챔피언 분석 – 상세페이지 (빌드 탭: 시작 아이템, 신발)

### Dao

```
List<ChampionDetail> getStart_items(@Param("championName") String champion_eg_name, @Param("lane") String lane1);  
List<ChampionDetail> getBoots(@Param("championName") String champion_eg_name, @Param("lane") String lane1);
```

### Mapper

```
<select id="getStart_items" resultType="ChampionDetail">  
  SELECT *  
  FROM (SELECT * FROM START_ITEM  
        WHERE CHAMPIONNAME=#{championName} AND TEAMPOSITION=#{lane}  
        ORDER BY START_CNT DESC) A  
  INNER JOIN (SELECT * FROM CHAMP_IMAGE) B  
  ON A.CHAMPIONID=B.CHAMPIONID  
  WHERE <![CDATA[ROWNUM<=2]]>  
</select>
```

```
<select id="getBoots" resultType="ChampionDetail">  
  SELECT *  
  FROM (SELECT *  
        FROM boots  
        WHERE CHAMPIONNAME=#{championName} AND TEAMPOSITION=#{lane}  
        ORDER BY BOOTS_CNT DESC)  
  WHERE <![CDATA[ROWNUM<=2]]>  
</select>
```

- interface로 받았던 파라미터를 통해 테이블의 모든 컬럼을 가져와서 게임횟수가 가장 높은 두 가지만 검색

## 챔피언 분석 - 상세페이지 (빌드 탭: 스펠, 시작 아이템, 신발)

### 실행 화면

소환사 주문		픽률	승률
		77.28% 534게임	48.88%
		5.07% 35게임	51.43%

시작 아이템		픽률	승률
		85.5% 348게임	50.86%
		9.58% 39게임	43.59%

신발		픽률	승률
		43.68% 242게임	55.37%
		24.73% 게임	47.45%

- 선택한 챔피언이 많이 선택한 소환사 주문 Top2
- 선택한 챔피언이 많이 선택한 시작 아이템 Top2
- 선택한 챔피언이 많이 선택한 신발Top2

## 챔피언 분석 - 상세페이지 (빌드 탭: 스킬 빌드)

### Service

```
//스킬 빌드
List<ChampionDetail> skill_build=champDao.getSkill_build(champion_eg_name, lane1);

ChampionDetail skill1=skill_build.get(0);
System.out.println("skill1="+skill1);
skill1=skill_build.get(0);
skill_build.set(0, skill1.setLv1(championSKill(skill1.getLv1())));
skill_build.set(0, skill1.setLv2(championSKill(skill1.getLv2())));
skill_build.set(0, skill1.setLv3(championSKill(skill1.getLv3())));
skill_build.set(0, skill1.setLv4(championSKill(skill1.getLv4())));
skill_build.set(0, skill1.setLv5(championSKill(skill1.getLv5())));
skill_build.set(0, skill1.setLv6(championSKill(skill1.getLv6())));
skill_build.set(0, skill1.setLv7(championSKill(skill1.getLv7())));
skill_build.set(0, skill1.setLv8(championSKill(skill1.getLv8())));
skill_build.set(0, skill1.setLv9(championSKill(skill1.getLv9())));
skill_build.set(0, skill1.setLv10(championSKill(skill1.getLv10())));

skill_build.set(0, skill1.setLv11(championSKill(skill1.getLv11())));
skill_build.set(0, skill1.setLv12(championSKill(skill1.getLv12())));
skill_build.set(0, skill1.setLv13(championSKill(skill1.getLv13())));
skill_build.set(0, skill1.setLv14(championSKill(skill1.getLv14())));
skill_build.set(0, skill1.setLv15(championSKill(skill1.getLv15())));

mav.addObject("skill1",skill1);
```

```
// 스킬 숫자 -> q w e
public String championSKill(String s) {
    if(s.equals("1")) return "Q";
    else if(s.equals("2")) return "W";
    else if(s.equals("3")) return "E";
    else if(s.equals("4")) return "R";
    return null;
}
```

- Dao인 IchampionDao interface의 getSkill\_build를 실행
- 게임횟수가 가장 많은 스킬 빌드 하나만 가져옴
- championSkill()을 통해 1,2,3,4로 되어있는 스킬을 영문으로 바꿔 줌
- ModelAndView에 스킬을 담음

### Service

```
String[] skill_lst=new String[] {skill1.getLv1(),skill1.getLv2(),skill1.getLv3(),skill1.getLv4(),skill1.getLv5(),
                                skill1.getLv6(),skill1.getLv7(),skill1.getLv8(),skill1.getLv9(),skill1.getLv10()};

int q=0,w=0,e=0;
for(int i=0;i<skill_lst.length;i++) {
    if(skill_lst[i].equals("Q")) q++;
    else if(skill_lst[i].equals("W")) w++;
    else if(skill_lst[i].equals("E")) e++;
}
```

```
if(q>e && q>w && e>w) {
    mav.addObject("master1",start1.getQ());
    mav.addObject("master2",start1.getE());
    mav.addObject("master3",start1.getW());
}
else if(q>e && q>w && w>e) {
    mav.addObject("master1",start1.getQ());
    mav.addObject("master2",start1.getW());
    mav.addObject("master3",start1.getE());
}
else if(e>q && e>w && q>w) {
    mav.addObject("master1",start1.getE());
    mav.addObject("master2",start1.getQ());
    mav.addObject("master3",start1.getW());
}
```

```
else if(e>q && e>w && w>q) {
    mav.addObject("master1",start1.getE());
    mav.addObject("master2",start1.getW());
    mav.addObject("master3",start1.getQ());
}
else if(w>q && w>e && q>e) {
    mav.addObject("master1",start1.getW());
    mav.addObject("master2",start1.getQ());
    mav.addObject("master3",start1.getE());
}
else if(w>q && w>e && e>q) {
    mav.addObject("master1",start1.getW());
    mav.addObject("master2",start1.getE());
    mav.addObject("master3",start1.getQ());
}
```

- 스킬을 다 찍은 순서를 가져오기 위해 skill\_lst에 Lv1~lv10까지 찍은 스킬을 담음
- 리스트 안에 있는 Q, W, E의 개수를 카운트
- 조건문을 통해 ModelAndView에 스킬 이미지를 담음

## 챔피언 분석 - 상세페이지 (빌드 탭: 아이템 빌드)

### Service

```
// 아이템 빌드
List<ChampionDetail> item_build=champDao.getItem_build(champion_eg_name, lane1, tier);
ChampionDetail build=item_build.get(0);
mav.addObject("build",build);
System.out.println("build="+build);

ChampionDetail build2=item_build.get(1);
mav.addObject("build2",build2);
System.out.println("build2="+build2);

ChampionDetail build3=item_build.get(2);
mav.addObject("build3",build3);
System.out.println("build3="+build3);

ChampionDetail build4=item_build.get(3);
mav.addObject("build4",build4);
System.out.println("build4="+build4);
```

- Dao인 IchampionDao interface의 getItem\_build()를 실행
- 게임횟수가 가장 많은 아이템 빌드 TOP4 가져옴
- ModelAndView에 아이템을 담음

## 챔피언 분석 - 상세페이지 (빌드 탭: 스킬 빌드, 아이템 빌드)

### Dao

```
List<ChampionDetail> getSkill_build(@Param("championName") String champion_eg_name, @Param("lane") String lane1);  
List<ChampionDetail> getItem_build(@Param("championName") String champion_eg_name, @Param("lane") String lane1, @Param("tier") String tier);
```

### Mapper

```
<select id="getSkill_build" resultType="ChampionDetail">  
    SELECT *  
    FROM (SELECT * FROM SKILL_BUILD  
          WHERE CHAMPIONNAME=#{championName} AND TEAMPOSITION=#{lane}  
          ORDER BY SKILL_PICK DESC)  
    WHERE ROWNUM=1  
</select>
```

```
<select id="getItem_build" resultType="ChampionDetail">  
    SELECT CHAMPIONNAME, CHAMPIONID, TEAMPOSITION, ITEM1, ITEM2, ITEM3, ITEM_TOTAL,  
          ITEM_CNT, ITEM_WIN, ITEM_WINRATE, ITEM_PICK  
    FROM (SELECT * FROM ${tier}_BUILD3  
          WHERE CHAMPIONNAME=#{championName} AND TEAMPOSITION=#{lane} AND ITEM3!=0  
          ORDER BY ITEM_CNT DESC)  
    WHERE <![CDATA[ROWNUM<=4]]>  
</select>
```

- interface로 받았던 파라미터를 통해 스킬 테이블의 모든 컬럼을 가져와서 게임 횟수가 가장 많은 한 가지 검색
- 아이템 테이블의 챔피언 이름, 아이디, 팀포지션, 아이템1, 아이템2, 아이템3, 해당 라인의 총 판수, 아이템 판수, 아이템 승수, 아이템 승률, 아이템 픽률 컬럼을 가져옴
- 게임횟수가 가장 많은 TOP4 검색



## 챔피언 분석 - 상세페이지 (빌드 탭: 스킬 빌드, 아이템 빌드)

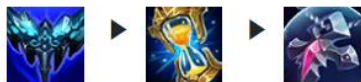
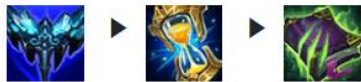
### 실행 화면

#### 스킬 빌드



Q W E Q Q R Q E Q E R E E W W 43.28%  
67게임

#### 아이템 빌드



픽률 승률

757%  
37게임 56.76%

3.07%  
15게임 53.33%

2.66%  
13게임 53.85%


143%  
7게임 57.14%

- 선택한 챔피언이 많이 선택한 스킬 빌드
- 선택한 챔피언이 많이 선택한 아이템 빌드 TOP4

## 챔피언 분석 – 상세페이지 (아이템 탭)







### 실행 화면

MIDDLE



갈리오-미드










챔피언 티어 : 3티어






빌드

**아이템**

카운터

아이템 빌드	픽률	승률
 ▶  ▶ 	771% 46게임	5217%
 ▶  ▶ 	3.85% 23게임	56.52%
 ▶  ▶ 	235% 14게임	5714%

신발	픽률	승률
	35.68% 213게임	5211%
	2764% 165게임	5273%
	16.75% ...	53.0%

- championDetail.jsp 에서 아이템 탭 클릭
- DB에서 검색해온 값과 함께 item.jsp로 forwarding
- RelationPage : item.jsp

### Controller

```
@GetMapping(value = "/itemInfo")
public ModelAndView itemInfo(Champion champ) {
    if (champ.getLane().equals("탑"))
        champ.setLane("TOP");
    else if (champ.getLane().equals("정글"))
        champ.setLane("JUNGLE");
    else if (champ.getLane().equals("미드"))
        champ.setLane("MIDDLE");
    else if (champ.getLane().equals("원딜"))
        champ.setLane("BOTTOM");
    else if (champ.getLane().equals("서포터"))
        champ.setLane("UTILITY");

    mav = champmm.itemInfo(champ.getChampionName(), champ.getLane(), champ.getTier(), champ.getChampionId());
    return mav;
}
```

- if- else문을 통해 한글 라인 정보를 영어로 바꿔 줌
- itemInfo() 함수를 실행

## 챔피언 분석 - 상세페이지 (아이템 탭)

### Service

```
List<ChampionDetail> championItem = champDao.getChampionItem(championName, lane);
ChampionDetail item_lst;

item_lst=championItem.get(0);
mav.addObject("item",item_lst);

item_lst=championItem.get(1);
mav.addObject("item1",item_lst);

item_lst=championItem.get(2);
mav.addObject("item2",item_lst);

item_lst=championItem.get(3);
mav.addObject("item3",item_lst);

item_lst=championItem.get(4);
mav.addObject("item4",item_lst);

item_lst=championItem.get(5);
mav.addObject("item5",item_lst);

item_lst=championItem.get(6);
mav.addObject("item6",item_lst);
```

```
List<ChampionDetail> championBoots = champDao.getChampionBoots(championName, lane);
ChampionDetail boots_lst;

boots_lst=championBoots.get(0);
mav.addObject("boots",boots_lst);

boots_lst=championBoots.get(1);
mav.addObject("boots1",boots_lst);

boots_lst=championBoots.get(2);
mav.addObject("boots2",boots_lst);

boots_lst=championBoots.get(3);
mav.addObject("boots3",boots_lst);

mav.setViewName("Detail/item");
return mav;
```

- getChampionItem(championName, lane) 함수로 검색해온 아이템  
10가지 빌드 정보를 ModelAndView에 담는다
- getChampionBoots(championName, lane) 함수를 통해 해당 챔피  
언의 신발 4가지 정보를 ModelAndView에 담는다
- item을 setViewName한 후 mav return

## 챔피언 분석 - 상세페이지 (아이템 탭)

### Dao

```
List<ChampionDetail> getChampionItem(@Param("championName") String champion_eg_name, @Param("lane") String lane);  
List<ChampionDetail> getChampionBoots(@Param("championName") String champion_eg_name, @Param("lane") String lane);
```

### Mapper

```
<select id="getChampionItem" resultType="ChampionDetail">  
    SELECT *  
    FROM(SELECT CHAMPIONNAME, TEAMPOSITION, ITEM1, ITEM2, ITEM3 , ITEM_WIN, ITEM_CNT,  
    ITEM_TOTAL, ITEM_WINRATE, ITEM_PICK  
    FROM GOLD_BUILD3  
    GROUP BY CHAMPIONNAME, TEAMPOSITION, ITEM1, ITEM2, ITEM3, ITEM_WIN,  
    ITEM_CNT, ITEM_TOTAL, ITEM_WINRATE, ITEM_PICK  
    HAVING ITEM1!=0 AND ITEM2!=0 AND ITEM3!=0 AND CHAMPIONNAME=#{championName}  
    AND TEAMPOSITION=#{lane}  
    ORDER BY ITEM_CNT DESC)  
    WHERE <![CDATA[ROWNUM<=10]]>  
</select>
```





```
<select id="getChampionBoots" resultType="ChampionDetail">  
    SELECT *  
    FROM (SELECT *  
    FROM GOLD_PLUS_BOOTS  
    WHERE CHAMPIONNAME=#{championName} AND TEAMPOSITION=#{lane} AND BOOTS!=0  
    ORDER BY BOOTS_CNT DESC)  
    WHERE <![CDATA[ROWNUM<=4]]>  
</select>
```

- interface로 받았던 파라미터를 통해 아이템 테이블의 챔피언 이름, 팀포지션, 아이템1,아이템2, 아이템3, 해당 라인의 총 판수, 아이템 판수, 아이템 승수, 아이템 승률, 아이템 픽률 컬럼을 가져옴
- 게임횟수가 가장 많은 TOP10 검색
- 아이템 테이블의 모든 컬럼을 가져와서 게임 횟수가 가장 많은 TOP4 검색

## 챔피언 분석 – 상세페이지 (아이템 탭)

### 실행 화면

아이템 빌드	픽률	승률
 ▶  ▶ 	771% 46게임	52.17%
 ▶  ▶ 	3.85% 23게임	56.52%
 ▶  ▶ 	235% 14게임	57.14%
 ▶  ▶ 	201% 12게임	50.0%
 ▶  ▶ 	134% 8게임	75.0%
 ▶  ▶ 	101% 6게임	100.0%
 ▶  ▶ 	0.84% 5게임	0.0%

신발	픽률	승률
	35.68% 213게임	52.11%
	27.64% 165게임	52.73%
	16.75% 100게임	53.0%
	12.9% 77게임	55.84%

- 해당 챔피언이 많이 선택한 아이템 빌드 TOP10
- 해당 챔피언이 많이 선택한 신발의 TOP4

## 4 프로젝트 수행 소감

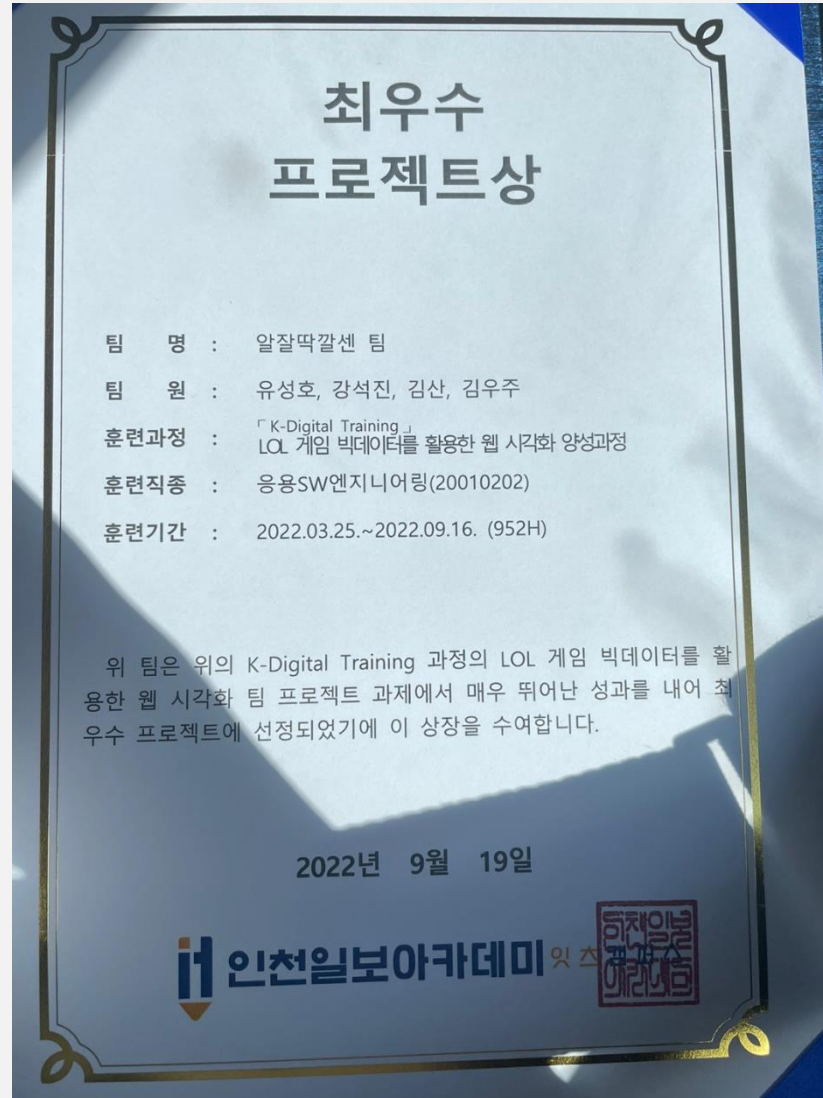
---

평상시에 궁금했고 구현하고 싶었던 영역이다 보니 상당히 즐겁게 프로젝트를 진행하였다.

프로젝트를 진행하면서 팀원들과 소통이 원활하게 진행되었고, 무언가를 만들어야 한다는 압박감으로 가득 차기보다는 즐겁게 만들고 배워간다는 생각으로 프로젝트를 임했던 것 같다.

팀원들과의 의사소통이나 협업의 중요성에 대해 다시 한번 깨닫게 되는 계기가 되기도 했고, 프로젝트 기간동안 힘든 점도 있었지만, 팀원들과 잘 극복해서 좋은 결과물 만들어 낸 것 같아 기쁘고 아주 유익한 시간이었던 것 같다.

## 프로젝트 결과





· THANK ·  
YOU