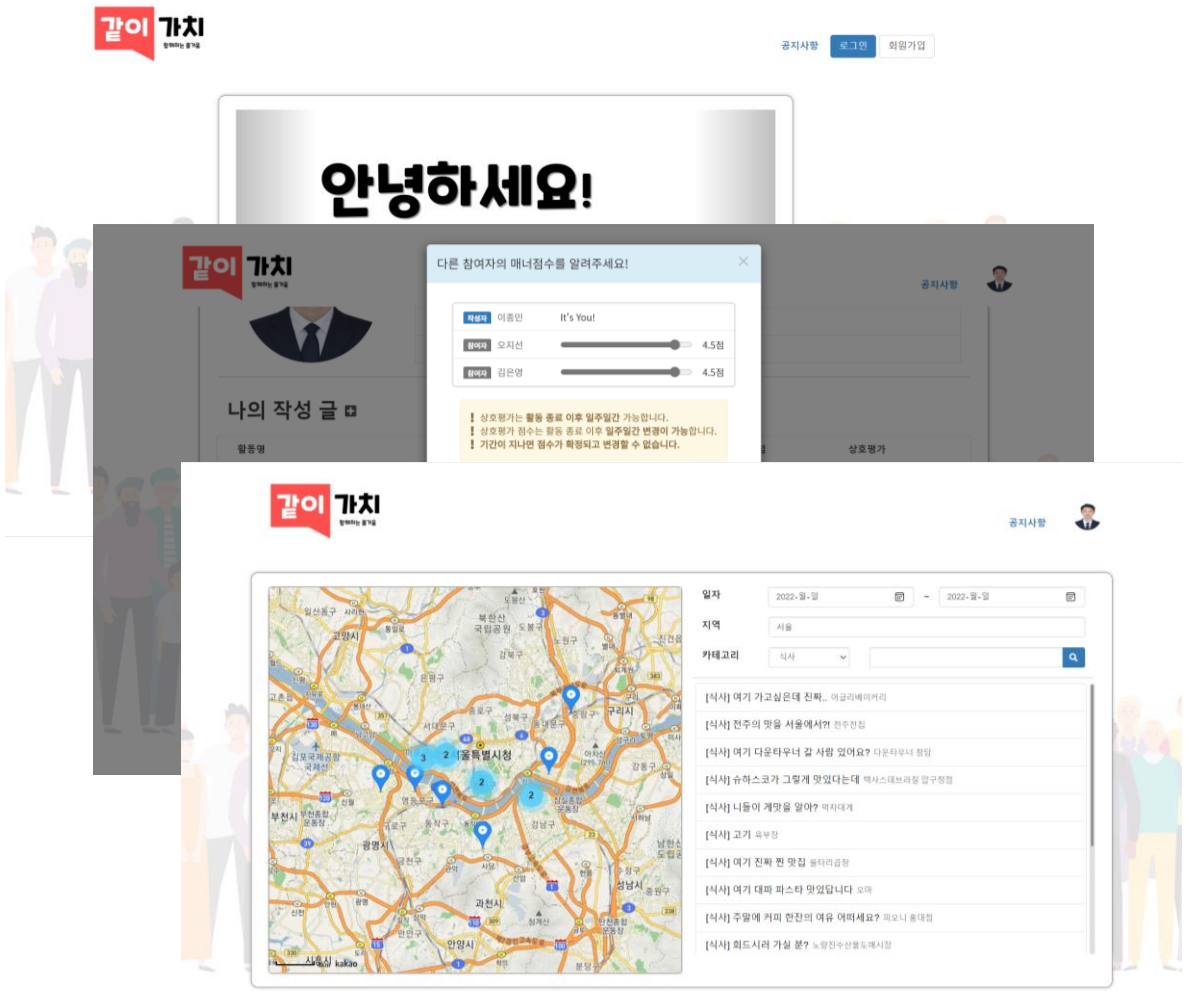
Project 01.

같이가치

About project

카카오맵 API를 활용한 소규모 모임을 위한 웹 프로그램으로 Spring Framework를 활용한 프로그램 개발에 대한 경험과 숙련도 향상을 위해 진행. 인원 모집을 위한 게시판 기능과 모임 장소에 대한 정보를 제공하기 위한 지도 및 장소 정보 제공 기능을 구현.

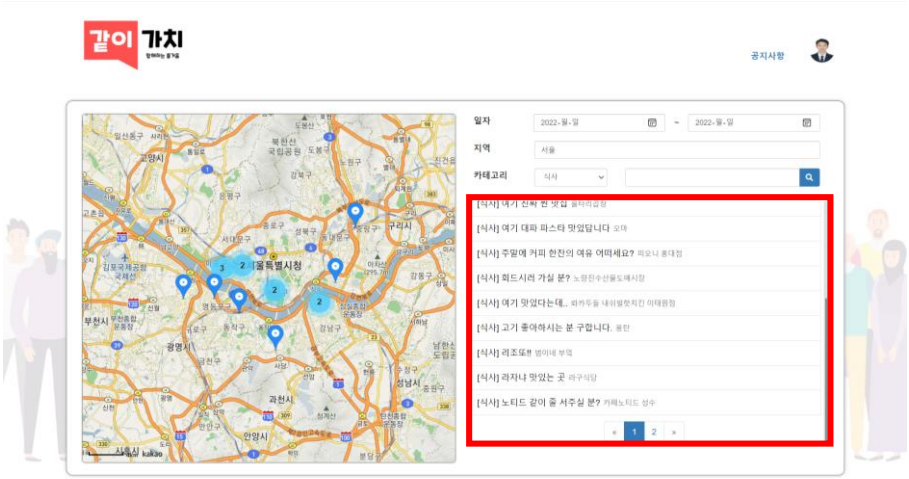
Project 01. 같이가치



Introduce project

작업 기간	2021. 12 ~ 2022. 01 (5주)
인력 구성(기여도)	BE 4명 / FE 1명 (BE 기여도 40%, FE 기여도 100%)
프로젝트 목적	팀 프로젝트로 Spring Framework 사용 경험 및 숙련도 향상을 위하여 진행함.
프로젝트 내용	• 사용자가 원하는 장소, 원하는 테마의 모임 게시글을 검색, 작성. • 카카오맵 API를 활용하여 모임 장소에 대한 정보와 위치를 제공. • 비밀 댓글, 상호 평가 기능을 통해 사용자간의 신뢰성 문제를 해결하고자 함.
주요 업무 및 상세 역할	1) 카카오맵 API를 활용한 장소 검색, 위치 표시 기능 구현 2) Ajax를 활용한 게시글 검색 및 모임 장소 표시 기능 구현 3) Ajax를 활용한 사용자 프로필 확인 기능 구현 4) 상호 평가 기능을 위한 로직 구성
사용언어 및 개발 환경	Java, Spring Framework, JavaScript, Kakaomap API, BootStrap, Oracle, Eclipse
참고 자료	https://github.com/fjswhd/project-gg

Main work 1. Ajax 및 Json 데이터를 활용한 게시글 검색



- **기능 소개**

- Ajax와 json 데이터를 활용한 게시글 검색

- **작업 내용**

- DB 검색 및 검색 결과 페이징을 위한 변수 전달

- Ajax 구현을 위한 Json 형태의 데이터 전달

Representation Layer

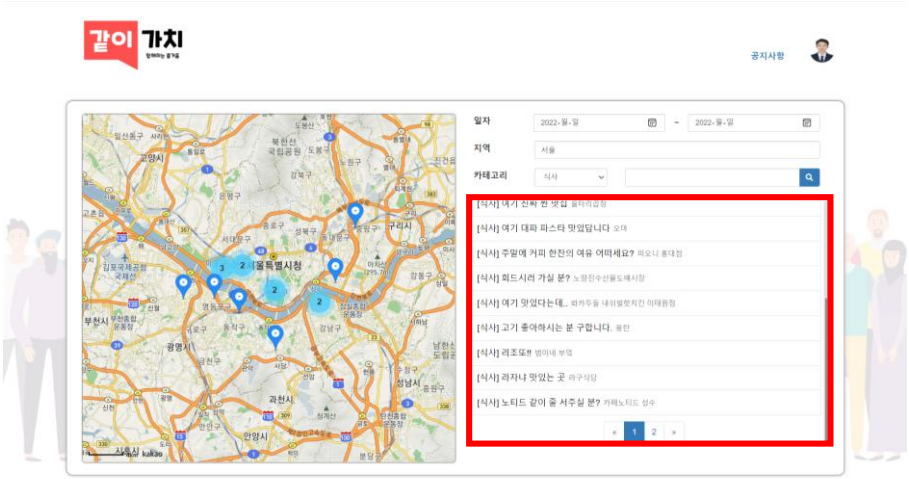
```
@RequestMapping(value = "/search", produces = "application/json;charset=utf-8")
@ResponseBody
public Map<String, Object> searchList(@RequestBody Map<String, Object> param) {
    /*
     *  searchResult
     *  itemList    : boardList (검색된 게시글 리스트)
     *  firstPage   : 페이징버튼 중 첫번째 버튼의 페이지 번호
     *  lastPage    : 페이징 버튼 중 마지막 버튼의 페이지 번호
     *  pageNum     : 현재 페이지 번호
     *
     */
    Map<String, Object> searchResult = bs.searchBoard(param);

    return searchResult;
}
```

Service Layer

```
//게시글 검색
public Map<String, Object> searchBoard(Map<String, Object> param) {
    //param : s_date, e_date, address, c_no, keyword, pageNum

    //검색 조건에 해당하는 총 게시글 개수
    int totalBoard  = bd.getSearchBoardCount(param);

    // 현재 페이지
    int pageNum = 1;

    if(param.containsKey("pageNum")) {
        String str = param.get("pageNum").toString();

        //pageNum이 숫자면
        if(str.matches("^[0-9]+$")) {
            pageNum = Integer.parseInt(str);
        }
    }

    //페이징 및 TopN검색을 위한 페이징 객체 생성 (startRow, endRow, pageNum, firstPage, lastPage)
    Paging paging = Paging.getPaging(totalBoard, pageNum);

    //매개변수에 topN 변수 넣고 전달
    param.put("startRow", paging.getStartRow());
    param.put("endRow", paging.getEndRow());

    List<Board> boardList = bd.searchBoard(param);

    Map<String, Object> resultMap = new HashMap<String, Object>();

    resultMap.put("itemList", boardList);
    resultMap.put("firstPage", paging.getFirstPage());
    resultMap.put("lastPage", paging.getLastPage());
    resultMap.put("pageNum", paging.getPageNum());

    return resultMap;
}
```

Project 01. 같이가치

Main work 1. Ajax 및 Json 데이터를 활용한 게시글 검색



- **기능 소개**

- Ajax와 Json 데이터를 활용한 게시글 검색

- **작업 내용**

- Fetch API를 사용한 Ajax 구현

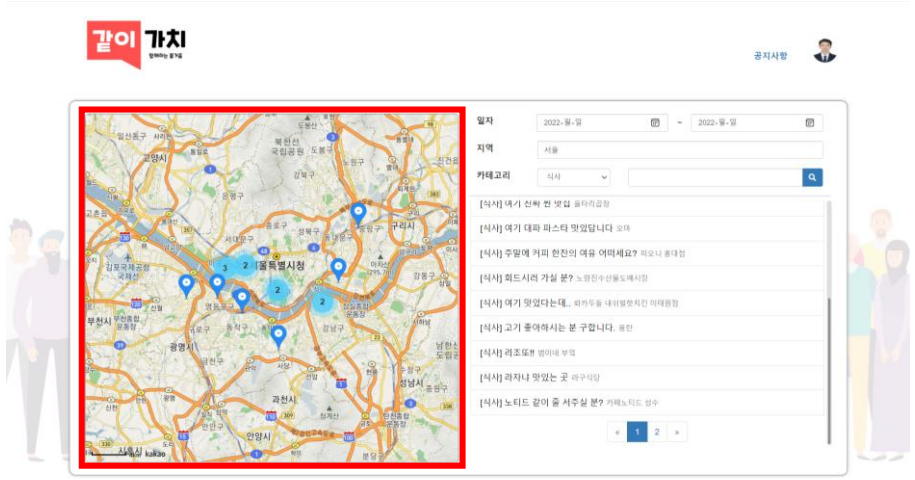
- Json데이터 형식을 활용한 통신 및 동적 화면 구성

Representation Layer

```
function search(sendData, displaySearchResult) {

    //검색조건에 따른 검색 결과 받아오기
    fetch('${_board}/search.do', {
        method: 'POST',
        body: sendData,
        headers: {
            'Content-Type': 'application/json;charset=utf-8'
        }
    }).then(function(response) {
        return response.json();
    }).then(function(resultMap) {
        var itemList = resultMap.itemList,
            pageNum = resultMap.pageNum,
            firstPage = resultMap.firstPage,
            lastPage = resultMap.lastPage;

        //이 url에서 pageNum이 변하지 않게 넘겨준 데이터로 페이지번호 설정
        param.pageNum = pageNum;

        //검색 결과 보여줌
        displaySearchResult(itemList);

        //맵에 마커 표시
        displayMarkers(itemList);

        //페이지 버튼 보여줌
        displayPageButton(pageNum, firstPage, lastPage);
    })
}
```

```
//검색 결과를 보여줌
function displaySearchResult(itemList) {
    var target = document.querySelector('ul.list-group'),
        container = document.querySelector('div#searchList');

    //검색 후 결과창 위로
    container.scrollTo(0, 0);

    //target에 남아있는 li지우기
    removeAllChild(target);

    //검색 결과가 없는 경우
    if (itemList.length == 0) {
        li = document.createElement('li');
        li.classList.add('list-group-item');
        li.innerHTML = '<i class="fas fa-times-circle mg-r-5"></i>검색 결과가 존재하지 않습니다.';
        target.appendChild(li);
        return;
    }

    //검색 결과가 있는 경우
    for(var i = 0; i < itemList.length; i++) {
        var board = itemList[i];
            li = document.createElement('li'),
            a = document.createElement('a'),
            small = document.createElement('small');

        li.classList.add('list-group-item')

        a.setAttribute('href', '${_board}/detail.do?b_no='+board.b_no);
        a.classList.add('cursor');
        a.style.color = '#000';
        a.innerHTML = '['+board.category.c_name+'] '+ board.subject

        small.classList.add('text-muted', 'mg-l-5');
        small.innerText = board.address.substring(0, board.address.lastIndexOf('('));

        li.appendChild(a);
        li.appendChild(small);

        target.appendChild(li);
    }
}
```

Main work 2. kakaomap API를 활용한 장소 표시



- **기능 소개**

- 게시글 검색 결과를 바탕으로 지도에 장소 표시

- **작업 내용**

- kakaomap API를 사용한 marker 및 장소 좌표 검색

- 검색 내용을 바탕으로 해당하는 위치에 마커 표시

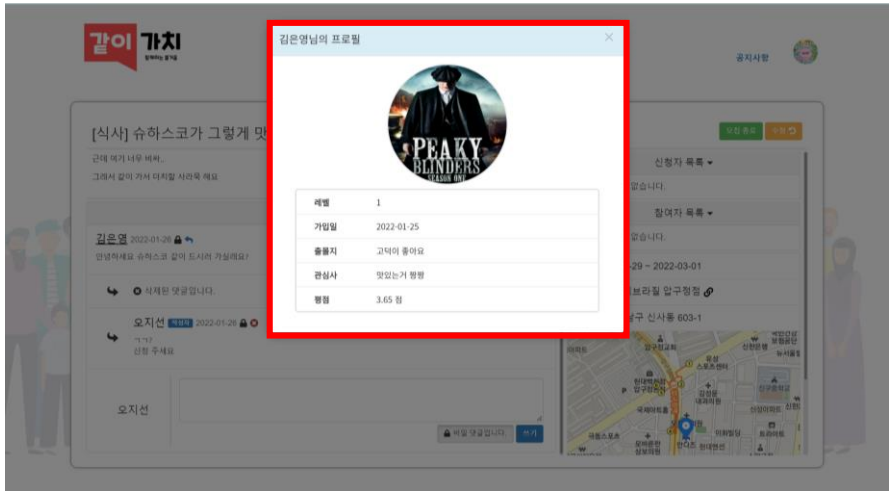
- **개선사항**

- 기획 단계에서 데이터 모델링 실패로 인한 내부함수 제거

Representation Layer

```
//마커 만들고 클러스터러에 넣기
function displayMarkers(itemList) {
    bounds = new kakao.maps.LatLngBounds();

    //markers에 있는 마커와 클러스터러에 있는 마커 지우기
    removeMarker();
    clusterer.clear();

    //검색 결과가 없는 경우
    if (itemList.length == 0) {
        return;
    }

    //검색 결과가 있는 경우
    itemList.map(function(item, index) {
        var address = item.address;
        var place = address.substring(0, address.lastIndexOf('('));
        address = address.substring(address.lastIndexOf('(') + 1, address.lastIndexOf(')'));

        //주소 + 장소명을 검색어로 장소 검색(좌표를 위해)
        var keyword = address + ' ' + place;

        var title = JSON.stringify(item);

        searchPlaces(keyword);

        //keyword로 장소 검색
        function searchPlaces(keyword) {
            ps.keywordSearch(keyword, placesSearchCB);
        }

        //장소검색이 완료됐을 때 호출되는 콜백함수
        function placesSearchCB(data, status, pagination) {
            if (status === kakao.maps.services.Status.OK) {
                var placePosition = new kakao.maps.LatLng(data[0].y, data[0].x),
                    marker = makeMarker(title, placePosition);

                //지도 반경을 검색 결과에 맞춰서 변경
                bounds.extend(placePosition);
                map.setBounds(bounds);

                //생성된 마커를 마커 배열에 넣기, 클러스터러에 추가하기
                markers.push(marker);
                clusterer.addMarker(marker);

            } else if (status === kakao.maps.services.Status.ZERO_RESULT) {
                return;

            } else if (status === kakao.maps.services.Status.ERROR) {
                return;
            }
        }
    })
}
```

Main work 3. Ajax를 활용한 사용자 프로필 확인 기능 구현



- **기능 소개**

- 사용자 아이디 클릭 시 사용자 프로필 창 생성

- **작업 내용**

- Ajax와 Json 데이터 형식을 활용한 데이터 통신

- 사용자 닉네임을 통해 사용자 식별 및 해당 사용자의 프로필정보를 기반으로 모달 창 생성

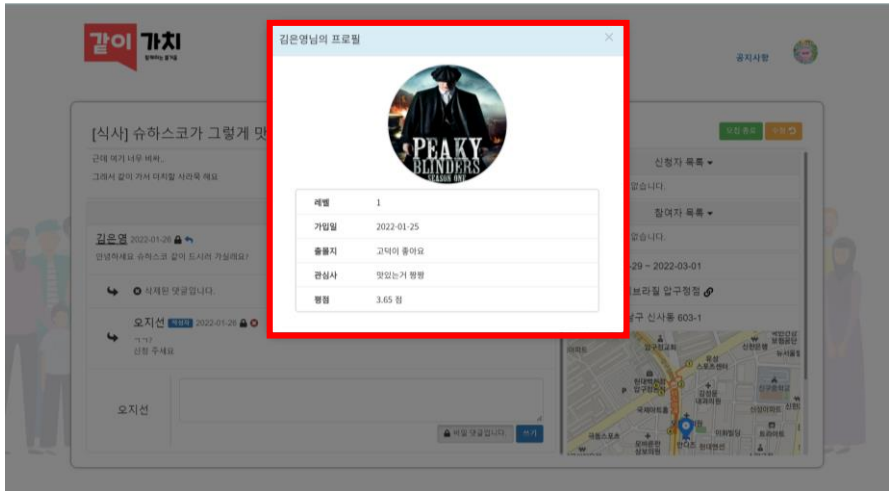
Representation Layer

```
@RequestMapping(value = "/getProfile", produces = "application/json;charset=utf-8")
@ResponseBody
public Map<String, Object> getProfile(@RequestBody Map<String, Object> param) {
    String m_id = (String) param.get("m_id");

    /*  profileResult
     *  nickname : 별명
     *  picture : 프로필 사진파일 명
     *  place : 출몰지
     *  rating : 평점
     *  reg_date : 가입일
     *  tag : 관심사
     *  level : 나이
     */

    Map<String, Object> profileResult = ms.getProfile(m_id);

    return profileResult;
}
```

Service Layer

```
//사용자 프로필 정보 조회 결과 반환
public Map<String, Object> getProfile(String m_id) {
    Member member = md.selectMember(m_id);

    //닉네임 픽쳐 플레이스 레이팅 가입일 태그 생일
    Map<String, Object> result = new HashMap<String, Object>();
    result.put("nickname", member.getNickname());
    result.put("picture", member.getPicture());
    result.put("place", member.getPlace());
    result.put("rating", member.getRating());
    result.put("reg_date", member.getReg_dateInStr());
    result.put("level",  member.getLevel());
    result.put("tag", member.getTag());

    return result;
}
```

Main work 3. Ajax를 활용한 사용자 프로필 확인 기능 구현



- **기능 소개**

- 사용자 아이디 클릭 시 사용자 프로필 창 생성

- **작업 내용**

- Fetch API를 사용한 Ajax 구현

- Json데이터 형식을 활용한 통신 및 동적 화면 구성

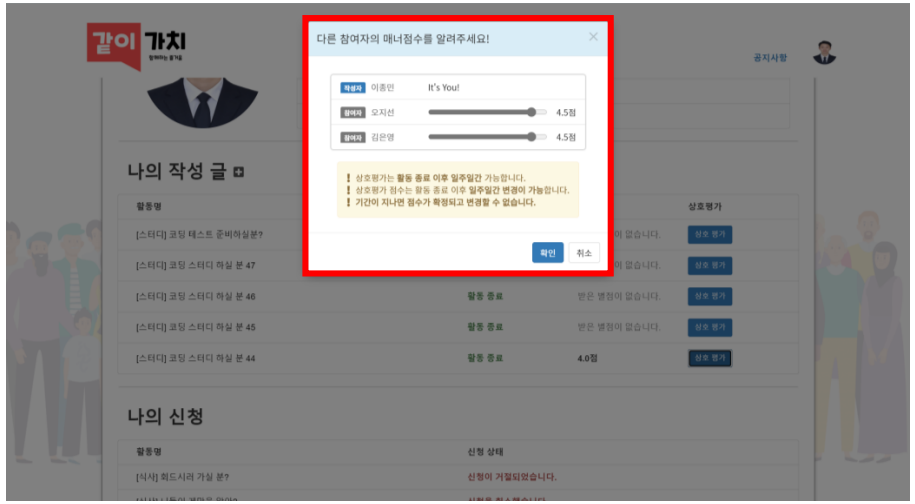
- **개선사항**

- 하드코딩 방식의 동적 화면 구성

Representation Layer

```
function getProfile(m_id) {
    var sendData = '{"m_id":"'+m_id+'"}';

    var title = document.querySelector('#profileTitle'),
        img = document.querySelector('#profileImg'),
        div = document.querySelectorAll('div.profileList');

    fetch('${_member}/getProfile.do', {
        method: 'POST',
        body: sendData,
        headers: {
            'Content-Type':'application/json;charset=utf-8'
        }
    }).then(function(response) {
        return response.json();
    }).then(function(data) {
        //닉네임 픽쳐 플레이스 레이팅 가입일 태그
        title.textContent = data.nickname + '님의 프로필';
        img.src = '${_profile}/' + data.picture;

        div[0].innerHTML = data.level;
        div[1].innerHTML = data.reg_date;
        div[2].innerHTML = data.place;
        div[3].innerHTML = data.tag;

        var rating = '';
        if (data.rating.toString().length == 1) {
            rating = data.rating.toString()+'.00점'
        } else if (data.rating.toString().length == 3) {
            rating = data.rating.toString()+'0점'
        } else {
            rating = data.rating.toString()+' 점'
        }

        div[4].innerHTML = rating;
    })
}
```

Main work 4. 상호 평가 로직 구성



- **기능 소개**

- 모임 종료 후 사용자 상호간의 평가 창 호출

- **작업 내용**

- 게시글 작성자 및 참여자 목록에 따라 평가 점수창 구성

- **개선사항**

- 기획 단계에서 잘못된 설계로 인한 로직 구성의 어려움

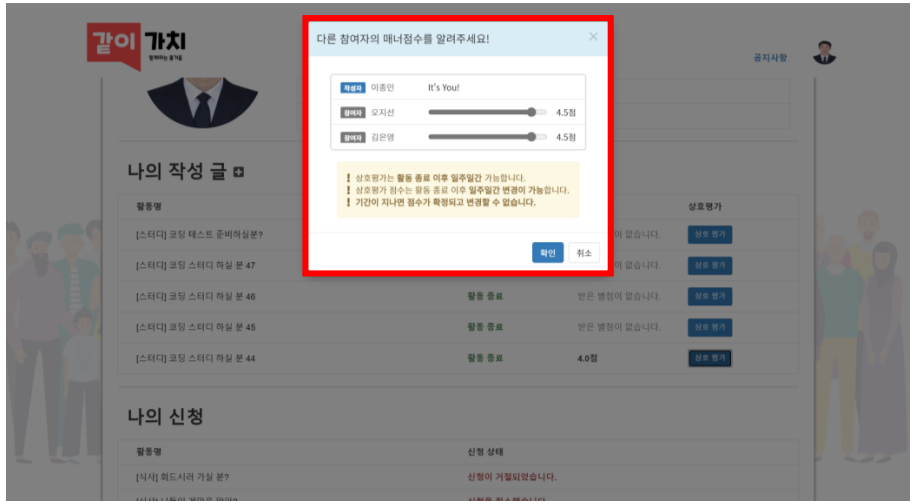
Representation Layer

```
//상호평가모달
@RequestMapping("/evalForm")
public String evalForm(@RequestParam Map<String, Object> param, Model model, HttpSession session, HttpServletRequest request) {
    if (request.getHeader("referer") == null || session.getAttribute("member") == null || param.size() == 0) {
        return "redirect:/error.do";
    }

    Map<String, Object> result = ras.evalForm(param);

    model.addAttribute("board", result.get("board"));
    model.addAttribute("partiList", result.get("partiList"));

    return "myPage/fragment/evalForm";
}
```

Service Layer

```
//평가점수 창 구성에 필요한 데이터 반환
public Map<String, Object> evalForm(Map<String, Object> param) {
    int b_no = Integer.parseInt(param.get("b_no").toString());

    //활동 작성자 + 참여자 리스트
    Board board = bd.getBoard(b_no);
    List<Parti> partiList = pd.ptList(b_no);

    //만약 내가 평가한 내역이 있다면 해당 내역 값을 초기값으로
    List<Rating> ratingList = rd.selectMyRatings(param);

    if(ratingList.size() != 0) {

        for (Rating rating : ratingList) {
            //평가받은 사람 id가 글쓴이이면 board에 해당 스코어 값 넣어줌
            if(board.getM_id().equals(rating.getM_id())) {
                board.setR_score(Math.round(rating.getR_score()*10)/(float)10);
            }

            for (Parti parti : partiList) {
                //평가받은 사람 아이디가 참여자 중에 있으면 해당 평가의 스코어값 넣어줌
                if (parti.getM_id().equals(rating.getM_id())) {
                    parti.setR_score(Math.round(rating.getR_score()*10)/(float)10);
                }
            }
        }
    }

    Map<String, Object> result = new HashMap<String, Object>();

    result.put("board", board);
    result.put("partiList", partiList);

    return result;
}
```

Main work 4. 상호 평가 로직 구성



- **기능 소개**

- 상호 평가 내용 DB에 반영 및 평가 받은 사용자의 평점 업데이트

- **작업 내용**

- 평가 후 평가받은 사람의 평점 업데이트시 트랜잭션 관리

- **개선사항**

- 기획 단계에서 평점을 수정할 수 있도록 한 점으로 인한 평점의 신뢰도 하락

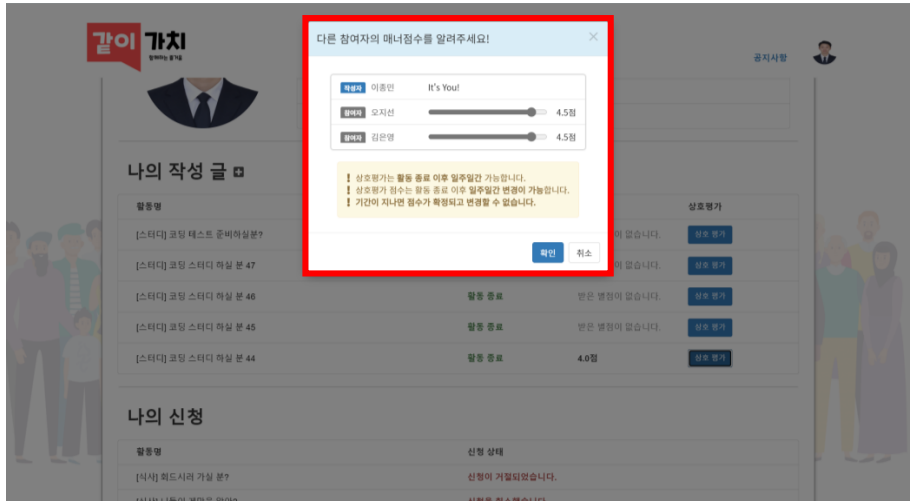
Representation Layer

```
@RequestMapping(value = "/eval", produces = "application/json;charset=utf-8")
@ResponseBody
public String evalForm(@RequestBody List<Rating> ratings, Model model, HttpSession session, HttpServletRequest request) {
    if (request.getHeader("referer") == null || session.getAttribute("member") == null || ratings == null || ratings.size() == 0) {
        return "redirect:/error.do";
    }

    //비지니스 로직에 필요한 데이터 전송
    int b_no = ratings.get(0).getB_no();
    String m_id_eval = ((Member) session.getAttribute("member")).getM_id();

    Map<String, Object> param = new HashMap<String, Object>();
    param.put("b_no", b_no);
    param.put("m_id_eval", m_id_eval);

    int result = ras.evalWithTx(ratings, param);

    return result+"";
}
```

Service Layer

```
//평가 반영
public int evalWithTx(List<Rating> ratings, Map<String, Object> param) {
    List<Rating> ratingList = rd.selectMyRatings(param);

    //1. 해당 rating 내역이 있는지 확인
    //있는지 검사할 때는 해당 글에 나의 평가 내역이 있는지 확인
    boolean hasRatingList = ratingList.size() != 0 ? true : false;

    int result = 0;

    //2. 없으면 insert, 있으면 update
    for (int i = 0; i < ratings.size(); i++) {
        //해당 게시글에 현재 사용자가 평가한 내역이 있으면 그 내역의 평가번호를 갖고 오고 아니면 가장 큰 평가번호 + 1
        int r_no = hasRatingList ? ratingList.get(i).getR_no() : rd.selectMaxR_no() + 1;

        Rating rating = ratings.get(i);

        rating.setR_no(r_no);

        //해당 게시글에 현재 사용자가 평가한 내역이 있으면 그 내역을 업데이트, 아니면 입력
        result += hasRatingList ? rd.updateRating(rating) : rd.insertRating(rating);

        //특정인의 평균 점수 구하기
        float avg = rd.selectAvgScore(rating.getM_id());

        //특정인의 평균 점수 업데이트
        Member member = new Member();
        member.setM_id(rating.getM_id());
        member.setRating(avg);
        int updateResult = md.updateRating(member);
    }

    return result;
}
```

Main work 4. 상호 평가 로직 구성



Root-contex.xml

```
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource"/>
</bean>

<tx:advice id="txAdvice" transaction-manager="transactionManager">
    <tx:attributes>
        <tx:method name="*" propagation="REQUIRED" />
        <!--  name속성값을 *로 지정하면 모든 메서드에 적용 즉 어떠한 메서드를 사용해도 된다 -->
    </tx:attributes>
</tx:advice>
<!-- AOP 설정을 사용해서 Transaction 적용 -->
<aop:config>
    <aop:advisor advice-ref="txAdvice" pointcut="execution(* com.ch.project.service.*Service*.*WithTx(..))"/>
    <!-- service패키지의 Service 포함하는 모든 클래스,인터페이스의 WithTx로 끝나는 메서드를 포인트 컷 한다.    -->
</aop:config>
```

- **기능 소개**

- 상호 평가 내용 DB에 반영 및 평가 받은 사용자의 평점 업데이트

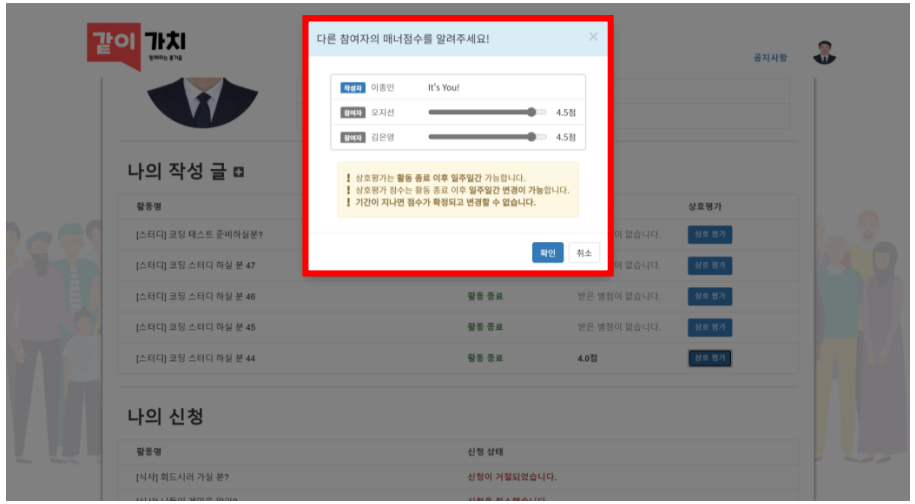
- **작업 내용**

- 평가 후 평가받은 사람의 평점 업데이트시 트랜잭션 관리

- **개선사항**

- 기획 단계에서 평점을 수정할 수 있도록 한 점으로 인한 평점의 신뢰도 하락

Main work 4. 상호 평가 로직 구성



- **기능 소개**

- Jquery를 사용한 ajax 통신을 통한 평가 결과 확인

- **작업 내용**

- Jquery와 Json을 사용한 ajax통신

Representation Layer

```
function evalFormSubmit() {
    //form Data를 Json Array형태로 만들어서 전송, 백엔드에서 List형태로 데이터 받을 수 있게
    var sendData = $('form[name=evalForm]').serializeArray(),
        ratings = [], rating = {};

    sendData.forEach(function(el, idx) {
        rating[el.name] = el.value;
        if (idx%4 == 3) {
            ratings.push(rating);
            rating = {};
        }
    })

    //Jquery를 활용한 ajax통신
    $.ajax({
        url: '${_myPage}/eval.do',
        method: 'POST',
        data: JSON.stringify(ratings),
        dataType: 'html',
        contentType: 'application/json;charset=utf-8'
    }).done(function(data) {
        if(data == ratings.length) {
            alert('상호평가가 반영되었습니다.');
        } else {
            alert('요청을 수행하지 못했습니다.\r\n다시 시도해주세요.');
        }
        $('#eval').modal('hide');
        location.href = '${_myPage}/main.do';
    })
}
```