

# LKH project

## 게시판과 댓글

**신입 개발자 김혜린**

LKH project에서  
제가 맡은 부분들을 자세하게 설명하겠습니다.

<https://github.com/memory0817/project>

## 목 차

01. 동기

02. 개발환경

03. 제작기간

04. 설계구조  
및 작동과정

05. Front &  
Back End

06. 후기

## 1. 동기

- 왜 게시판 인가?

웹 페이지에서 가장 중요한 기능이 무엇이냐고 묻는다면

저는 게시판이라고 생각합니다.

개념으로 배웠던 내용들을 실전 프로젝트로 적용하면서

코딩감과 경험을 쌓기 위해 선택했습니다.

또 게시판과 댓글 구현을 통해 Restful API와 Ajax에 대한

개념들을 더 공부하기 위해서입니다.

## 2. 개발환경

Front-End

**Bootstrap 4**

**JQuery 3**

**Ajax**

### Front-End 개발환경, 선택이유는?

Bootstrap 4를 기반으로 한 홈페이지 디자인이 깔끔하고 직관적이어서 선택했습니다.

Back-End

**Spring 5.1.3**

**JDK 1.8**

**Oracle**

**Java 8**

**Tomcat 8.5**

**Mybatis 3.4.6**

**Maven 2.9**

### Back-End 개발환경, 선택이유는?

현업에서 대체적으로 사용하는 툴들을 사용하여 바로 현업으로 들어갔을 시 빠른 적응을 위해 선택했습니다.

**Mybatis**는 효율성을 위해 선택하였습니다. JDBCTemplate도 좋았지만 **Mybatis**가 유지,보수적인 면에서 뛰어난 효율성을 보였으며, 생산성의 차이 역시 **Mybatis**쪽이 높아 사용하는 툴로써 아주 매력적이었습니다.

### 3. 제작기간



**1.7 ~ 1.11 (5일)** 프로젝트 주제 회의 및 개발환경 설정

**1.14~1.19 (5일)** 지적사항 수정 및 이슈해결

**1.21~1.25 (4일)** 시스템 설계

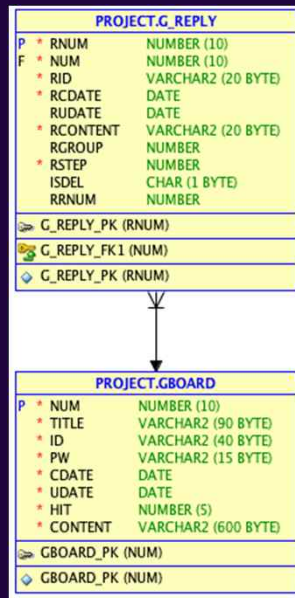
**1.30~2.18 (19일)** 시스템 및 서버 구현시작 & View 구현 시작

**2.19~2.25 (6일)** 버그 수정 및 배포테스트

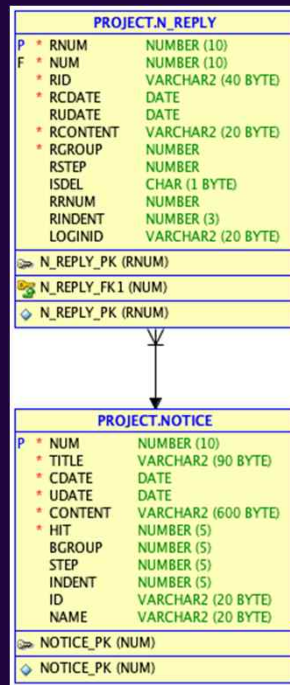
## 4. 설계구조

### - 데이터베이스 모델링

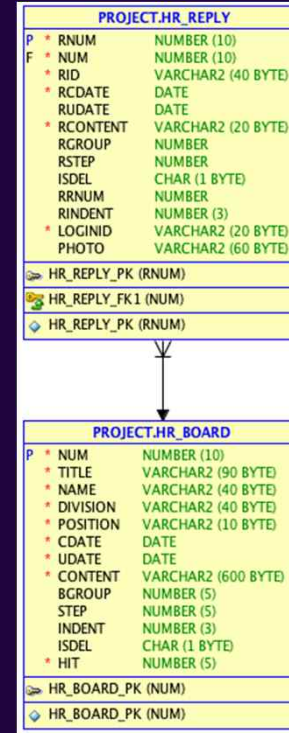
#### 일반 게시판



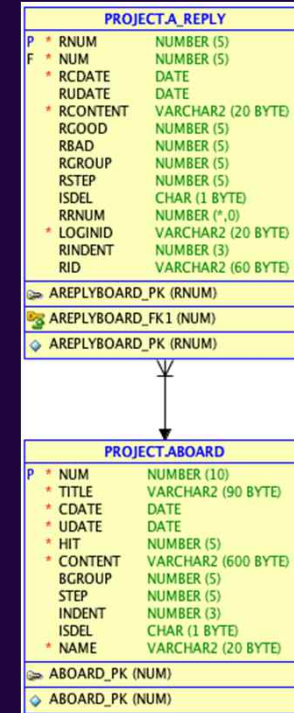
#### 공지사항



#### 부서별 게시판



#### 대나무숲



## 4. 설계구조

### - 패키지 구조

#### 일반 게시판

- com.lkh.myapp.gboard.dao
  - G\_RbbsDAO.java
  - G\_RbbsDAOImplXML.java
  - GbbsDAO.java
  - GbbsDAOImplXML.java
- com.lkh.myapp.gboard.dto
  - G\_RbbsDTO.java
  - GbbsDTO.java
- com.lkh.myapp.gboard.service
  - G\_RbbsSvc.java
  - G\_RbbsSvcImpl.java
  - GbbsSvc.java
  - GbbsSvcImplXML.java

#### 공지사항

- com.lkh.myapp.notice.dao
  - NoticeDAO.java
  - NoticeDAOImplXML.java
  - NoticeRbbsDAO.java
  - NoticeRbbsDAOImplXML.java
- com.lkh.myapp.notice.dto
  - NoticeDTO.java
  - NoticeRbbsDTO.java
- com.lkh.myapp.notice.service
  - NoticeRbbsSvc.java
  - NoticeRbbsSvcImpl.java
  - NoticeSvc.java
  - NoticeSvcImpl.java

#### 부서별 게시판

- com.lkh.myapp.sale.dao
- com.lkh.myapp.test.dao
- com.lkh.myapp.hr.dao
  - IT\_bbsDAO.java
  - IT\_bbsDAOImplXML.java
  - IT\_RbbsDAO.java
  - IT\_RbbsDAOImplXML.java
- com.lkh.myapp.IT.dto
  - IT\_bbsDTO.java
  - IT\_RbbsDTO.java
- com.lkh.myapp.IT.service
  - IT\_bbsSvc.java
  - IT\_bbsSvcImplXML.java
  - IT\_RbbsSvc.java
  - IT\_RbbsSvcImpl.java

#### 대나무숲

- com.lkh.myapp.aboard.dao
  - A\_bbsDAO.java
  - A\_bbsDAOImplXML.java
  - A\_RbbsDAO.java
  - A\_RbbsDAOImplXML.java
- com.lkh.myapp.aboard.dto
  - A\_bbsDTO.java
  - A\_RbbsDTO.java
- com.lkh.myapp.aboard.service
  - A\_bbsSvc.java
  - A\_bbsSvcImpl.java
  - A\_RbbsSvc.java
  - A\_RbbsSvcImplXML.java

## 4. 설계구조

### - 패키지 구조

#### 컨트롤러

- com.lkh.myapp.controller
  - Aboard\_bbsController.java
  - Aboard\_RbbsController.java
  - AdminController.java
  - G\_RbbsController.java
  - Gboard\_bbsController.java
  - HR\_bbsController.java
  - HR\_RbbsController.java
  - IT\_bbsController.java
  - IT\_RbbsController.java
  - LoginController.java
  - Maincontroller.java
  - MemberController.java
  - Notice\_RbbsController.java
  - NoticeController.java
  - SALE\_bbsController.java
  - SALE\_RbbsController.java
  - TEST\_bbsController.java
  - TEST\_RbbsController.java
  - VisitorController.java

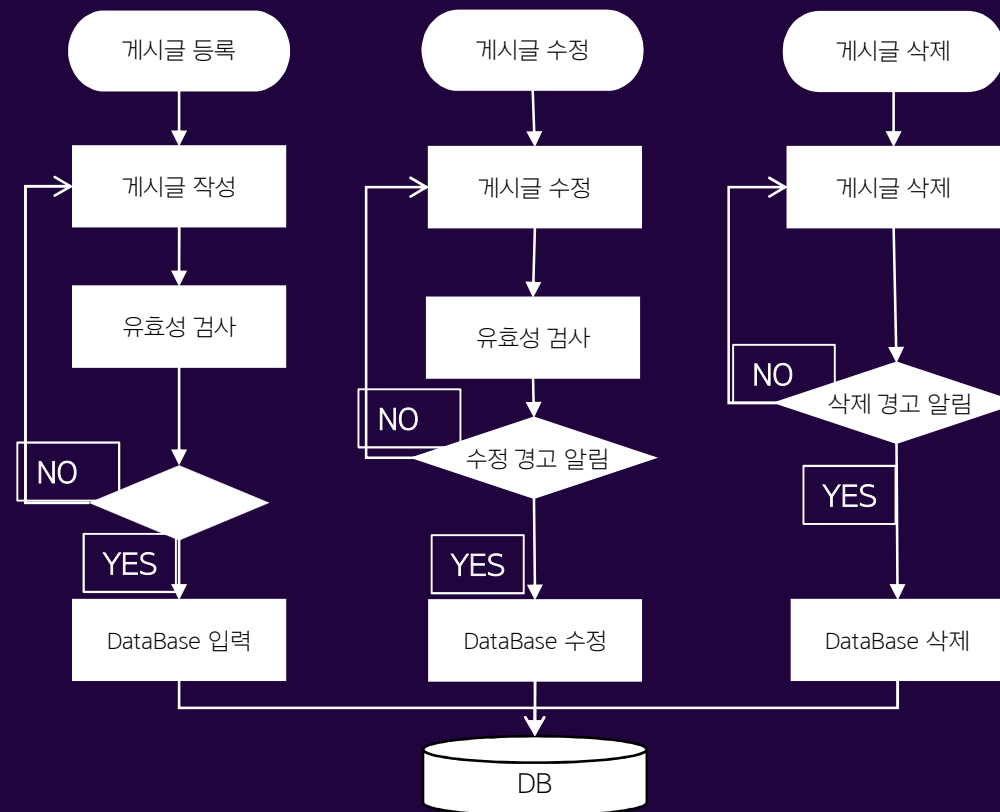
#### View

- views
  - Aboard
    - abbslist.jsp
    - abbsmodify.jsp
    - abbsread.jsp
    - abbsreply.jsp
    - abbswrite.jsp
  - admin
  - gboard
    - gbbslist.jsp
    - gbbsmodify.jsp
    - gbbsread.jsp
    - gbbsread2.jsp
    - gbbswrite.jsp
  - HR
    - HRbbslist.jsp
    - HRbbsmodify.jsp
    - HRbbsread.jsp
    - HRbbsreply.jsp
    - HRbbswrite.jsp
  - IT
    - ITbbslist.jsp
    - ITbbsmodify.jsp
    - ITbbsread.jsp
    - ITbbsreply.jsp
    - ITbbswrite.jsp
- notice
  - adminnoticelist.jsp
  - noticemodify.jsp
  - noticeread.jsp
  - noticereplyform.jsp
  - noticewrite.jsp
- reply
- SALE
  - SALEbbslist.jsp
  - SALEbbsmodify.jsp
  - SALEbbsread.jsp
  - SALEbbsreply.jsp
  - SALEbbswrite.jsp
- TEST
  - TESTbbslist.jsp
  - TESTbbsmodify.jsp
  - TESTbbsread.jsp
  - TESTbbsreply.jsp
  - TESTbbswrite.jsp
- visitboard
  - visitboard.jsp



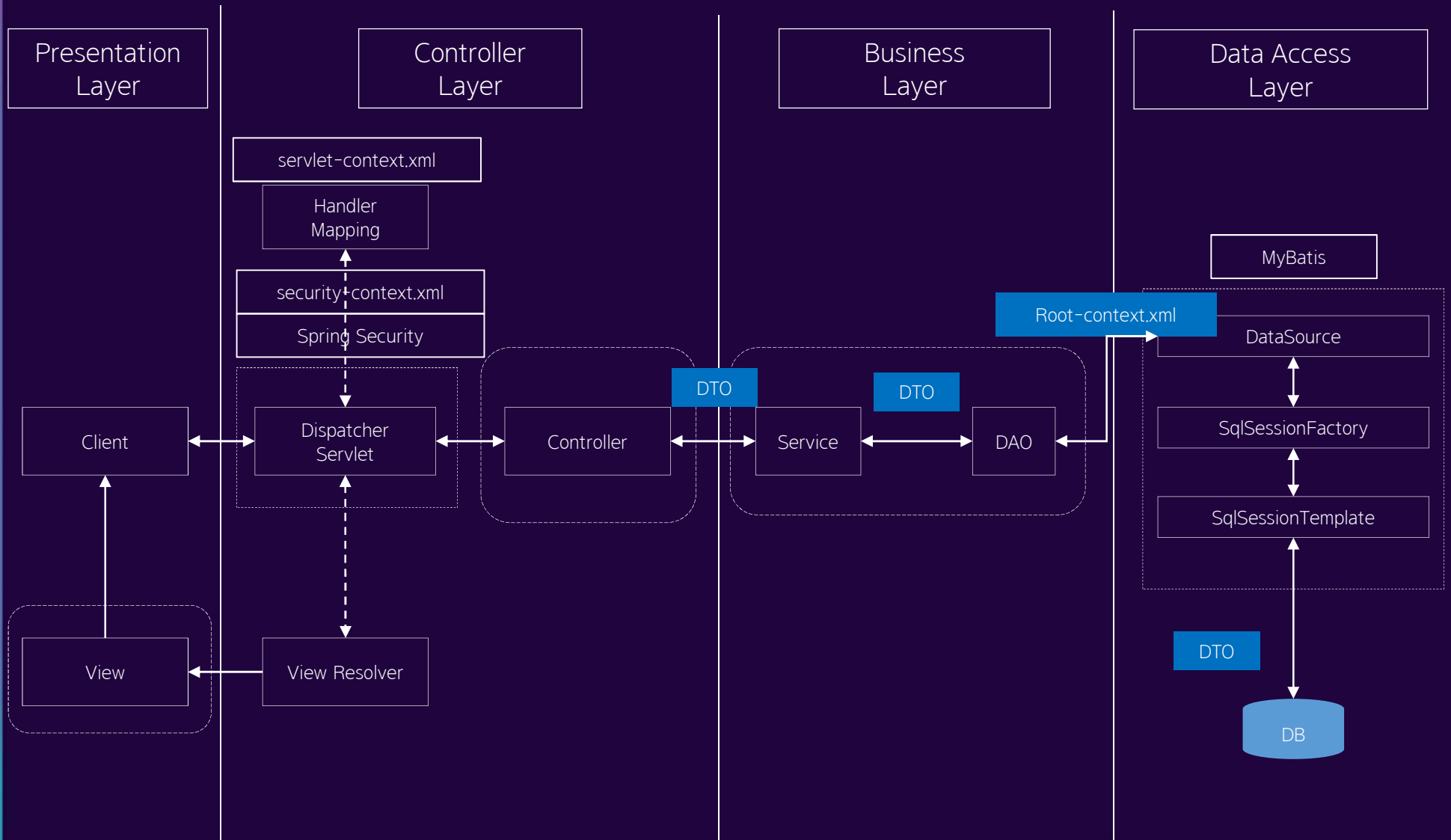
## 4. 설계구조

### - 프로세스설계도



## 4. 작동구조

### - MVC2 패턴



## 5. front-end

HR부서 게시판

검색

제목+내용

검색

글번호	제목	작성자	직급	작성일	조회수
32	zzzzz	관리자	사원	2019-02-25	110
28	삭제된 게시물입니다				
29	[답글]: 1로통통	admin	관리자	2019-02-19	13
31	[답글]: 1로통통	lly4337	과장	2019-02-19	2
26	삭제된 게시물입니다				
27	[답글]: 123123123	admin	관리자	2019-02-19	3
25	1233	관리자	관리자	2019-02-19	2
24	[답글]: 1231	admin	관리자	2019-02-19	2
5	[답글]: 123123	admin	대리	2019-02-15	9

글쓰기

1

HR부서\_게시글 작성

제목

제목을 입력하세요

✓

작성자

관리자

직급

대리

내용

내용을 입력하세요

✓

(0/300)

등록 취소

작성일자: 관리자/ROLE\_ADMIN/대리 2019-02-27 13:04:05

댓글입력

관리자/ROLE\_ADMIN/대리 2019-02-26 00:00:00

@관리자/ROLE\_ADMIN/대리 11

관리자/ROLE\_ADMIN/대리 2019-02-26 00:00:00

@관리자/ROLE\_ADMIN/대리 22

관리자/ROLE\_ADMIN/대리 2019-02-26 00:00:00

1

관리자/ROLE\_ADMIN/대리 2019-02-26 00:00:00

@관리자/ROLE\_ADMIN/대리 11

관리자/ROLE\_ADMIN/대리 2019-02-26 00:00:00

@관리자/ROLE\_ADMIN/대리 22

관리자/ROLE\_ADMIN/대리 2019-02-26 00:00:00

2

관리자/ROLE\_ADMIN/대리 2019-02-26 00:00:00

5

관리자/ROLE\_ADMIN/대리 2019-02-26 00:00:00

1

관리자/ROLE\_ADMIN/대리 2019-02-26 00:00:00

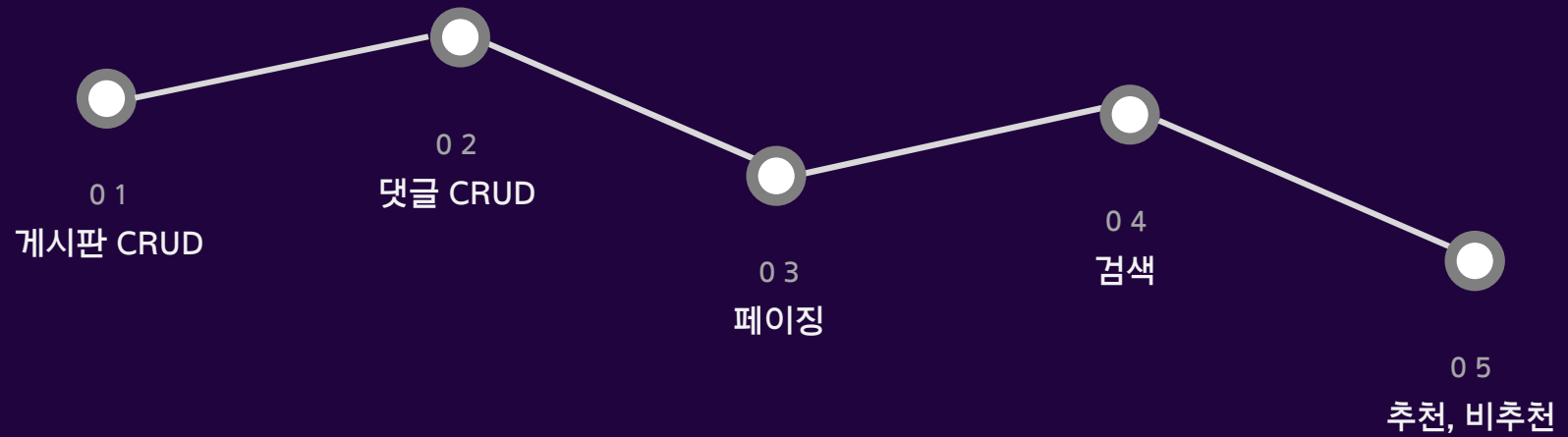
2

1 2

↳ 댓글은 Ajax로 구현해서  
페이지 이동을 해도 새로고침이  
일어나지 않습니다.

## 5. back-end

- Function Index



## 5. back-end

### -게시판 CRUD

관리자	사원	2019-02-25	112
admin	관리자	2019-02-19	14
lly4337	과장	2019-02-19	2
admin	관리자	2019-02-19	3
관리자	관리자	2019-02-19	3
admin	관리자	2019-02-19	2
admin	대리	2019-02-15	9

글쓰기

### HR부서\_게시글 작성

제목	제목을 입력하세요
작성자	관리자
직급	대리
내용	내용을 입력하세요

```
@Inject
HR_bbsSvc hrsvc;

//게시글 등록양식-write_form
@RequestMapping("/write")
public String write(Model model) {
    HR_bbsDTO hrbbstdto = new HR_bbsDTO();
    model.addAttribute("hr_bbsDTO", hrbbstdto);
    return "/HR/HRbbswrite";
}

//게시글 등록처리
@RequestMapping("/writeOK")
public String writeOK(@Valid @ModelAttribute("hr_bbsDTO") HR_bbsDTO hrbbstdto,
    @RequestParam("cnt") int cnt) {
    logger.info("String writeOK 호출됨:" + hr_bbstdto);

    int cnt = 0;
    if(result.hasErrors()) {
        logger.info(result.toString());
        return "/HR/HRbbswrite";
    }

    try {
        cnt = hrsvc.write(hr_bbsDTO);
        logger.info("게시글 등록건수:" + cnt);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return "redirect:/HR/list";
}
```

```
<mapper namespace="mappers.hrbbs">

    <!-- 글쓰기 -->
    <insert id="write">
        INSERT INTO hr_board
        (num,title,name, division, position, hit, content)
        values(hr_boardnum_seq.nextval,#{title},#{name},#{division},#{position},#{hit},#{content})
    </insert>

    @Repository(value = "hr_bbsDAOImplXML")
    public class HR_bbsDAOImplXML implements HR_bbsDAO {

        private static final Logger logger = LoggerFactory.getLogger(HR_bbsDAOImplXML.class);

        @Inject
        SqlSession sqlSession;

        // 글쓰기
        @Override
        public int write(HR_bbsDTO hr_bbsDTO) throws Exception {
            return sqlSession.insert("mappers.hrbbs.write");
        }

        <form:form modelAttribute="hr_bbsDTO" action="/HR/writeOK" method="post">
            <form:hidden path="division" value="공정지원" />

            <table class="table table-sm" summary="게시글 쓰기">
                <colgroup>
                    <col width="20%">
                        <input type="text" value="제목" />
                    </col>
                    <col width="20%">
                        <input type="text" value="작성자" />
                    </col>
                    <col width="20%">
                        <input type="text" value="직급" />
                    </col>
                    <col width="40%">
                        <input type="text" value="내용" />
                    </col>
                </colgroup>
            </table>

            <input type="button" value="글쓰기" />
        </form>

        // 게시글 등록
        $("#btn1").on("click",function(e){
            e.preventDefault();
            //유효성체크
            if(valchk()){
                $("form").submit();
            }
        });
```

## 5. back-end -댓글 CRUD

작성자: 관리자/ROLE\_ADMIN/대리

전체 댓글 목록을 위해 운영원칙에 위배된 댓글은 삭제됩니다.

0/100

등록

작성자: 관리자/ROLE\_ADMIN/대리 2019-02-27 13:04:05

댓글 입력

댓글

관리자/ROLE\_ADMIN/대리 2019-02-26 00:00:00

@관리자/ROLE\_ADMIN/대리 11

```
@RestController
@PreAuthorize("hasAnyRole('HR','ADMIN')")
@RequestMapping("/hrrbbs")
public class HR_RbbsController {

    private static Logger logger = LoggerFactory.getLogger(HR_RbbsController.class);

    @Inject
    HR_RbbsSvcImplXML hr_rbbssvc;

    // 댓글 등록
    @RequestMapping(value = "/posts", method = RequestMethod.POST)
    public ResponseEntity<String> write(@RequestBody HR_RbbsDTO hrddto) {
        ResponseEntity<String> resCode = null;
        if (hrddto == null) {
            resCode = new ResponseEntity<String>("fail", HttpStatus.BAD_REQUEST);
            return resCode;
        }
        try {
            hr_rbbssvc.write(hrddto);
            resCode = new ResponseEntity<String>("success", HttpStatus.OK);
        } catch (Exception e) {
            resCode = new ResponseEntity<String>("fail", HttpStatus.BAD_REQUEST);
            e.printStackTrace();
        }
        return resCode;
    }
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="mappers.hrrbbs">

    <!-- 댓글등록 -->
    <insert id="write" parameterType="HR_RbbsDTO">
        INSERT INTO hr_reply (rnum,
            num, rid, loginid, rcontent, rgroup)
        VALUES (HR_REPLYNUM_SEQ.nextval, #{num}, #{rid}, #{loginid}, #{rcontent}, HR_REPLYNUM_seq.currval)
    </insert>

</mapper>
```

```
@Repository("hr_RbbsDAOImplXML")
public class HR_RbbsDAOImplXML implements HR_RbbsDAO {

    private static final Logger logger = LoggerFactory.getLogger(HR_RbbsDAOImplXML.class);

    @Inject
    SqlSession sql;

    // 댓글 등록
    @Override
    public int write(HR_RbbsDTO hr_rbbsDTO) throws Exception {
        return sql.insert("mappers.hrrbbs.write", hr_rbbsDTO);
    }
}
```

```
$.ajax({
    type: "POST", // post, get, put, delete
    url: "/hrrbbs/posts",
    headers: {"Content-Type": "application/json", "X-HTTP-Method-Override": "POST"},
    dataType: "text",
    data: JSON.stringify({
        num: num,
        rid: rid,
        loginid: loginid,
        rcontent: rcontent
    }),
    success: function(result){
        // 댓글 목록 호출
        //console.log(result);
        replyList(rereqPage);
    },
    error: function(xhr, status, err){
    }
});
```

## 5. back-end -페이징

	11128	20
	11127	20
	11126	20
	11125	20

1
2
3
4

```
private void init() {
    //1) endPage 계산 : 올림(요청페이지/한페이지에 보여줄 페이지수) * 한페이지에 보여줄 페이지수
    this.endPage = (int) Math.ceil(this.recordCriteria.getReqPage() /
        (double) this.pageNumberPerPage) * this.pageNumberPerPage;
    System.out.println(this.recordCriteria.getReqPage());
    System.out.println(this.pageNumberPerPage);
    System.out.println(endPage);
    //2) startPage 계산 : 종료페이지 - 한페이지에 보여줄 페이지수 + 1
    this.startPage = this.endPage - this.pageNumberPerPage + 1;

    //3) finalEndPage 계산 : 올림(전체 레코드수/한페이지에 보여줄 레코드수)
    this.finalEndPage = (int) Math.ceil(this.totalRec / (double) this.recordCriteria.getNumPerPage());
    //종료페이지가 최종페이지보다 크면 종료페이지를 최종페이지로!
    if(endPage > finalEndPage) {
        endPage = finalEndPage;
    }

    //4) prev 계산 : 현재페이지의 시작페이지가 1이 아니면 버튼 보이기
    this.prev = this.startPage != 1 ? true : false;

    //5) next 계산 : 현재페이지의 종료페이지 * 한페이지에 보여줄 레코드수가 전체 레코드수보다 작으면 다음버튼 보이기
    this.next = this.totalRec > (this.endPage * this.recordCriteria.getNumPerPage()) ? true : false;
}

// 파라미터 설정
public String makeURL() {
    StringBuffer str = new StringBuffer();

    if(recordCriteria.getReqPage() != 0) {
        str.append("reqPage="+recordCriteria.getReqPage());
    }
}
```

```
<!-- 글목록 요청페이지 -->
<select id="list" resultType="HR_bbsDTO">
    select t2.*
    from (select row_number() over (order by bgroup desc, step asc) as znum,t1.*
        from hr_board t1 ) t2
    where znum between #{startRec} and #{endRec}
```

```
@Override
public List<HR_bbsDTO> list(int startRec, int endRec) throws Exception {
    Map<String, Object> map = new HashMap<>();
    map.put("startRec", startRec);
    map.put("endRec", endRec);
    return sqlSession.selectList("mappers.hrbbs.list", map);
}
```

```
@Override
public void list(HttpServletRequest request, Model model) throws Exception {
    logger.info("void list(HttpServletRequest request, Model model) 호출됨!");

    int NUM_PER_PAGE = 10; //한페이지에 보여줄 레코드수
    int PAGE_NUM_PER_PAGE = 10; //한페이지에 보여줄 페이지수

    int reqPage = 0; //요청페이지
    int totalRec = 0; //전체레코드수

    String searchType = null; //검색타입
    String keyword = null; //검색어

    PageCriteria pc = null;
    RecordCriteria rc = null; //검색조건이 없는 경우의 레코드 시작, 종료페이지연산
    FindCriteria fc = null; //검색조건이 있는 경우의 레코드 시작, 종료페이지 연산 + 검색타입

    List<HR_bbsDTO> alist = null;

    // 요청페이지가 없는 경우 1페이지로 이동
    if(request.getParameter("reqPage") == null ||
        request.getParameter("reqPage") == "") {
        reqPage = 1;
    } else {
        reqPage = Integer.parseInt(request.getParameter("reqPage"));
    }
}
```



## 5. back-end

### -검색

```
<!-- 검색목록 -->
<select id="flist" resultType="HR_bbsDTO">
    select t2.*
    from (select row_number() over (order by bgroup desc, step asc) as znum, t1.*
    from hr_board t1
    where znum > 0
    <bind name="keyword" value="'%'+keyword+'%'" />
    <choose>
        <!-- 제목+내용 -->
        <when test="searchType == 'TC'.toString()">
            and title like #{keyword} or content like #{keyword}
        </when>
        <!-- 제목 -->
        <when test="searchType == 'T'.toString()">
            and title like #{keyword}
        </when>
        <!-- 내용 -->
        <when test="searchType == 'C'.toString()">
            and content like #{keyword}
        </when>
        <!-- 작성자 -->
        <when test="searchType == 'N'.toString()">
            and content like #{keyword}
        </when>
        <otherwise>
            and title like #{keyword} or content like #{keyword} and name like #{keyword}
        </otherwise>
    </choose>
    ) t2
    where znum between #{startRecord} and #{endRecord}
</select>
```

10086	[답글] [답글] 제목_9998sgsadgasg	테스터5	2019-01-18
10102	[답글] 제목_9954	테스터5	2019-01-21
10103	[답글] 제목_9944	테스터5	2019-01-21
10105	[답글] 제목_9937	테스터5	2019-01-21

1 2

검색어 제목 ↓ 답글 검색

```
// 검색목록
@Override
public List<HR_bbsDTO> list(int startRecord, int endRecord, String searchType, keyword) {
    List<HR_bbsDTO> list = null;
    list = hr_dao.list(startRecord, endRecord, searchType, keyword);
    return list;
}
```

```
// 검색목록
@Override
public List<HR_bbsDTO> list(int startRecord, int endRecord, String searchType, keyword) {
    Map<String, Object> map = new HashMap<>();
    map.put("startRecord", startRecord);
    map.put("endRecord", endRecord);
    map.put("searchType", searchType);
    logger.info("서치타입: " + searchType);
    map.put("keyword", keyword);

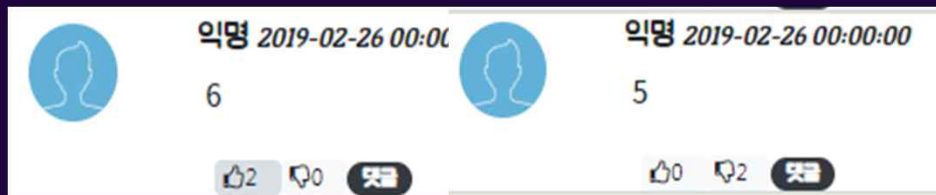
    return sqlSession.selectList("mappers.hrbbs.flist", map);
}
```

↳ JDBC로 구현할 때 보다  
Mybatis를 사용할 때 시각적으로  
좀 더 보기 편하고 SQL를 더 효율적으로  
구성할 수 있다는 것을 배웠습니다.



## 5. back-end

### -추천, 비추천



```
<update id="goodOrbad" parameterType="java.util.Map">
  <choose>
    <when test="goodOrbad == 'good'">
      update a_reply set rgood = rgood+1 where rnum = #{rnum}
    </when>
    <when test="goodOrbad = 'bad'">
      update a_reply set rbad = rbad+1 where rnum = #{rnum}
    </when>
  </choose>
</update>
```

```
@Override
public int goodOrbad(String rnum, String goodOrbad) {
    Map<String,String> map = new HashMap<>();
    map.put("rnum", rnum);
    logger.info(goodOrbad);
    map.put("goodOrbad", goodOrbad);

    return sql.update("mappers.arbbs.goodOrbad", map);
}
```

```
//호감,비호감
@RequestMapping(value="/posts/{goodOrbad}/{rnum}",method=RequestMethod.PUT)
public ResponseEntity<String> delete(
    @PathVariable("goodOrbad") String goodOrbad,
    @PathVariable("rnum") String rnum) {

    ResponseEntity<String> resCode = null;
    logger.info(goodOrbad);
    if(rnum == null || rnum.length() == 0) {
        resCode = new ResponseEntity<String>("fail",HttpStatus.BAD_REQUEST);
        return resCode;
    }
    try {
        logger.info(goodOrbad);
        logger.info(rnum);

        arbbsSvc.goodOrbad(rnum, goodOrbad);
        resCode = new ResponseEntity<String>("success",HttpStatus.OK);
    } catch (Exception e) {
        resCode = new ResponseEntity<String>("fail",HttpStatus.BAD_REQUEST);
        e.printStackTrace();
    }
    return resCode;
}
```

```
$('.goodBtn').on('click', function(){
    console.log('호감');
    var $li = $(this).parents('li');
    var rnum = $li.attr('data-rnum');
    console.log(rnum);

    //자바스크립트에서 종괄호는 객체 대괄호는 배열

    $.ajax({
        type:"PUT",
        url:"/arbbs/posts/good/"+rnum,
        dataType:"text",
        success:function(){
            replyList(rereqPage);
        },
        error:function(xhr,status,err){
            console.log("xhr"+xhr);
            console.log("status"+status);
            console.log("err"+err);
        }
    })

    $('.badBtn').on('click', function(){
        console.log('비호감');
        var $li = $(this).parents('li');
        var rnum = $li.attr('data-rnum');
        console.log(rnum);

        //자바스크립트에서 종괄호는 객체 대괄호는 배열

        $.ajax({
            type:"PUT",
            url:"/arbbs/posts/bad/"+rnum,
            dataType:"text",
            success:function(){
                replyList(rereqPage);
            },
            error:function(xhr,status,err){
                console.log("xhr"+xhr);
                console.log("status"+status);
                console.log("err"+err);
            }
        })
    });
```

## 6. 후기

우리가 일상 사용하는 게시판, 댓글 매소드를 구현하는데에  
생각보다 엄청난 노력과 시간이 든다는 것을 체감할 수 있었습니다.

프로젝트를 진행하면서 느낀 점은  
Mybatis의 효율성을 제대로 체감할 수 있었습니다.  
Mybatis덕분에 시간을 절약하고 그 시간 동안 다른 구현단계에 더 투자할 수 있었습니다.

더불어 Ajax필요성도 절실히 느낄 수 있었습니다.  
페이지가 다시 로딩되는 문제들을 Ajax기술을 조합하니  
간단하게 해결할 수 있어서 좋았고, 제가 원하는 View부분들을 깔끔하게 구현해낼 수 있어서  
원하는 목표를 이루어 만족감을 고취시킬 수 있었습니다.

프로젝트를 진행해보니 막히는 구간이 생기면  
스스로 정보를 찾아서 연구하고 문제를 해결하는 능력을 키우게 되는,  
저 자신을 업그레이드 시킬 수 있었던 소중한 시간을 경험할 수 있었습니다.

앞으로 더 문제가 닥치면 스스로의 능력을 키워  
해결할 수 있는 능동적인 개발자가 되도록 노력하겠습니다.

읽어주셔서 감사합니다!