게시판 CRUD
학원 홈페이지 프로젝트

박민정

아래 링크를 통해 직접 확인해보실 수 있습니다.
http://dolgodo888.cafe24.com/academy/

목차

1. 개요

2. 설계 구조

3. 주요 로직 설명

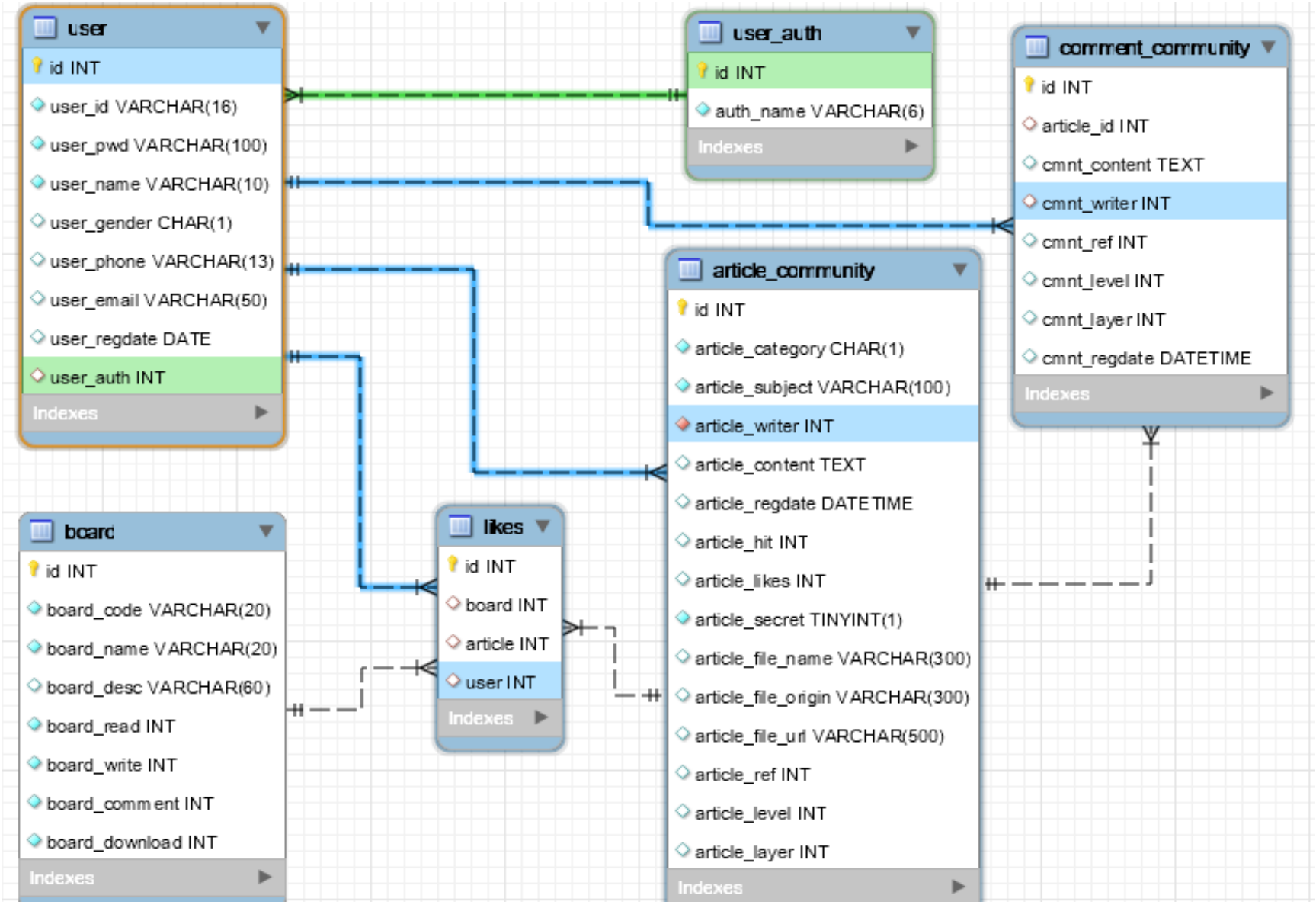
4. 후기

1. 개요

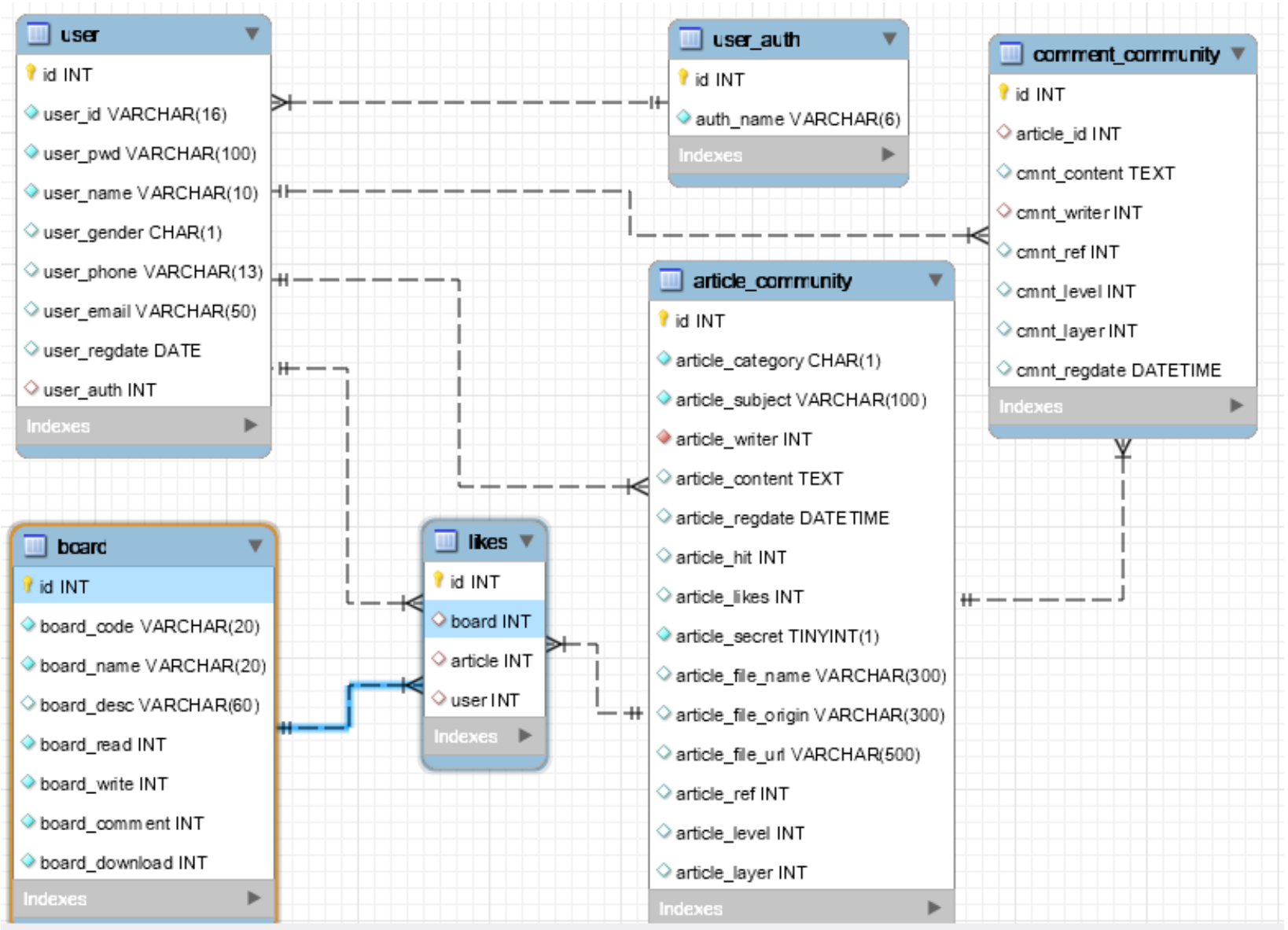
아래 주소를 통해 직접 확인하실 수 있습니다.
dolgodo888.cafe24.com/academy

- 제작 기간 : 02.07 - 03.14 (약 5주)
- 제작 계기 : 앞선 팀 프로젝트 이후 혼자 프로젝트를 진행하며 spring을 전체적으로 이해하고자 하였습니다. 게시판 CRUD 기능을 위주로 웹페이지를 제작하였으며 이전 프로젝트에서 조금 더 개선시켜 개발하였습니다. 암호화된 비밀번호, 쿠키를 이용한 아이디 저장, 로그인 인터셉터, 추천 기능 등을 구현하였습니다.
- 개발 환경
  - Front : HTML, CSS, JavaScript(Jquery)
  - Back : Spring, JAVA, MySQL(DB), Mybatis, Maven, Tomcat
- 소스코드 : https://github.com/pmj-jung/portfolio/tree/main/Academy_Project

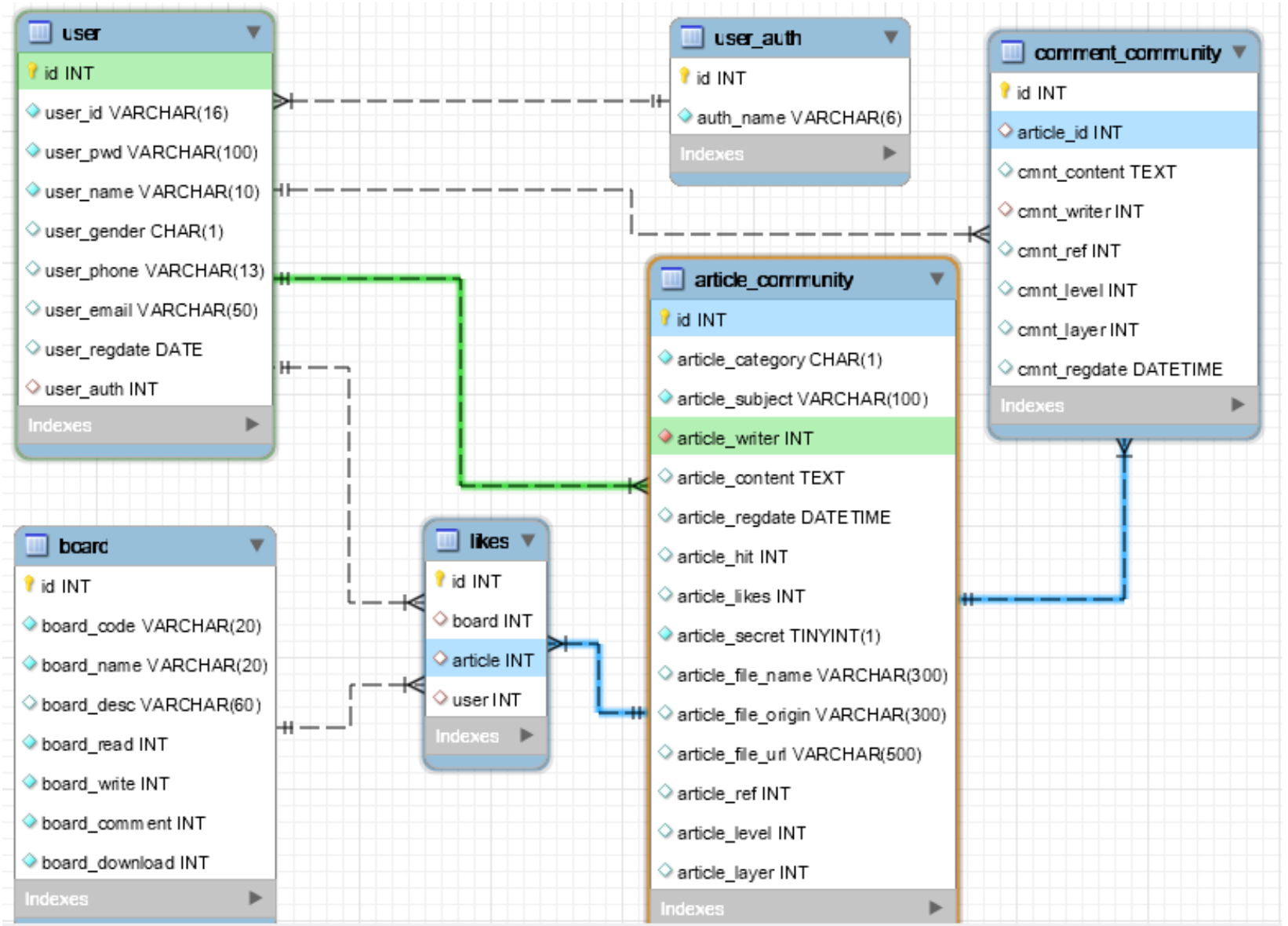
2. 설계 구조 데이터베이스 모델링



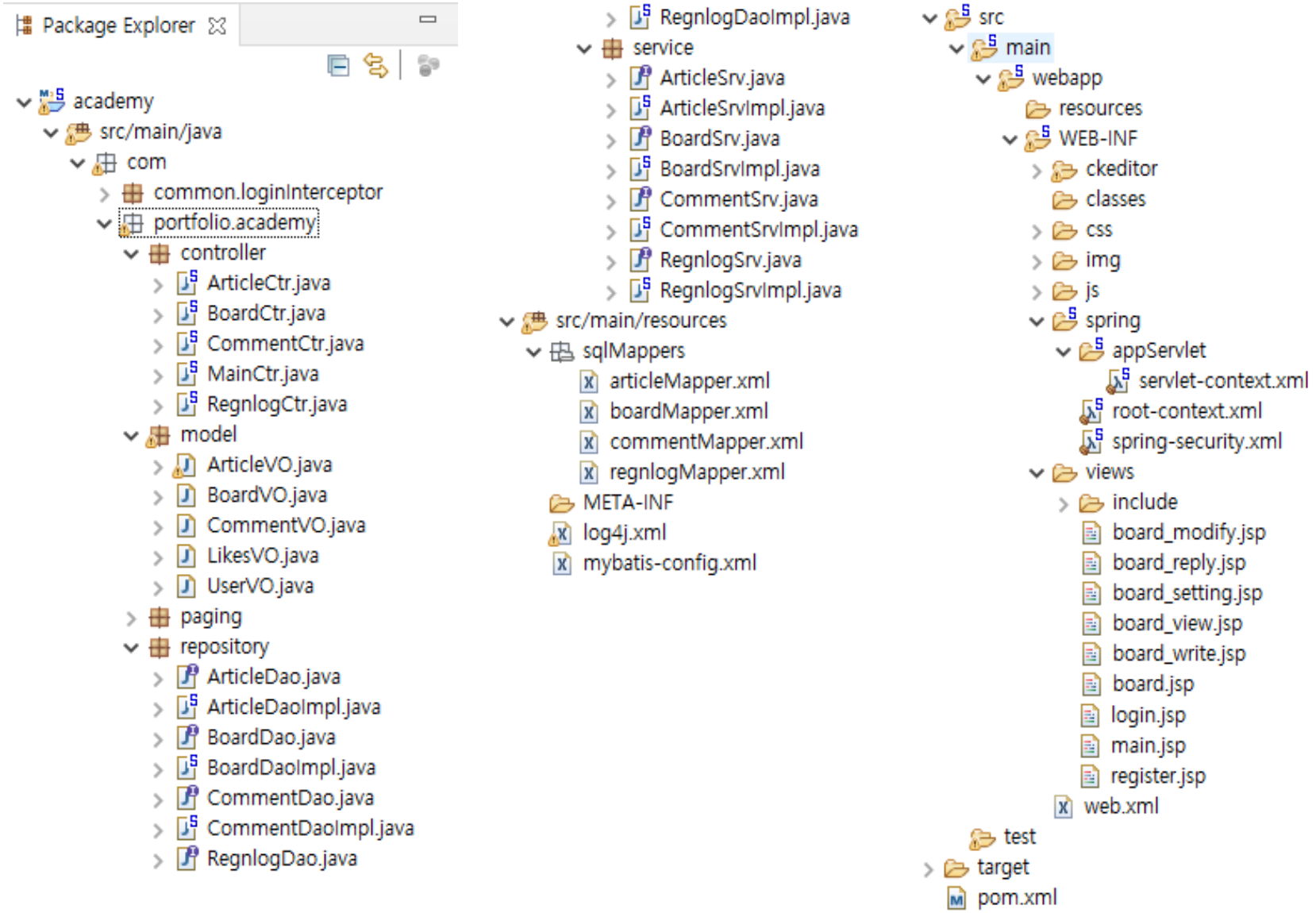
2. 설계 구조 데이터베이스 모델링



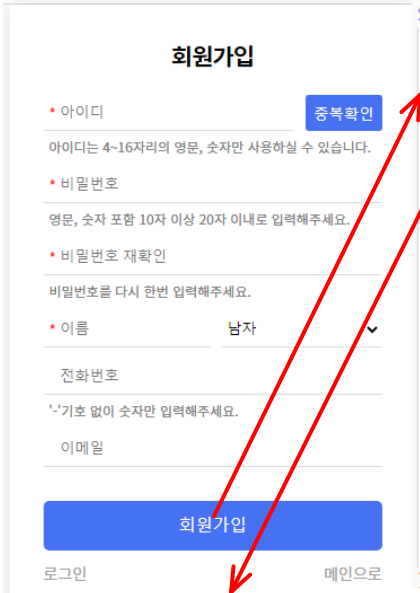
2. 설계 구조 데이터베이스 모델링



2. 설계 구조 패키지 구조


Package Explorer
academy
src/main/java
com
common.loginInterceptor
portfolio.academy
controller
ArticleCtr.java
BoardCtr.java
CommentCtr.java
MainCtr.java
RegnlogCtr.java
model
ArticleVO.java
BoardVO.java
CommentVO.java
LikesVO.java
UserVO.java
paging
repository
ArticleDao.java
ArticleDaoImpl.java
BoardDao.java
BoardDaoImpl.java
CommentDao.java
CommentDaoImpl.java
RegnlogDao.java
RegnlogDaoImpl.java
service
ArticleSrv.java
ArticleSrvImpl.java
BoardSrv.java
BoardSrvImpl.java
CommentSrv.java
CommentSrvImpl.java
RegnlogSrv.java
RegnlogSrvImpl.java
src/main/resources
sqlMappers
articleMapper.xml
boardMapper.xml
commentMapper.xml
regnlogMapper.xml
META-INF
log4j.xml
mybatis-config.xml
src
main
webapp
resources
WEB-INF
ckeditor
classes
css
img
js
spring
appServlet
servlet-context.xml
root-context.xml
spring-security.xml
views
include
board_modify.jsp
board_reply.jsp
board_setting.jsp
board_view.jsp
board_write.jsp
board.jsp
login.jsp
main.jsp
register.jsp
web.xml
test
target
pom.xml

3. 주요 로직 설명 회원가입



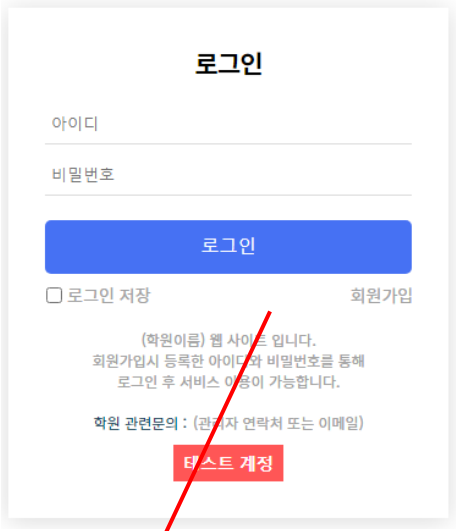

```javascript
$("#regBtn").click(function () {
    if (checkId() && checkPwd() && checkRePwd() && checkName() && checkPhone() && checkEmail()) {
        $.ajax({
            url: "/academy/setRegister",
            type: "POST",
            data: $("#regFrm").serialize(),
            success: function (resData) {
                if (resData == "success") {
                    alert("회원가입이 완료되었습니다.\n로그인 후 이용해주세요.");
                    window.location.href = "/academy/login.do";
                } else {
                    alert("이미 존재하는 아이디입니다.\n아이디를 변경해주세요.");
                    $("#userId").val("");
                    $("#userId").focus();
                }
            },
            error: function () {
                alert("회원가입 에러 발생\n관리자에게 문의하세요.");
                return false;
            }
        });
    }
});
```

유저가 회원가입 시 JS에서 입력된 정보들이 올바르게 입력되었는지 확인 후, 모든 값이 TRUE가 반환이 되면 입력된 데이터를 가지고 URL 요청을 보냅니다. URL이 일치하는 컨트롤러에서 **아이디 중복확인 및 비밀번호 암호화를 처리**합니다. 이때 아이디가 중복된다면 JS로 failure를 반환하고, 아이디가 유일하다면 DB에 회원정보를 저장하고 JS로 success를 반환합니다. JS로 success가 넘어갔다면 로그인 화면으로 이동합니다.

```java
@RequestMapping(value = "/setRegister", method = RequestMethod.POST)
@ResponseBody
public String setRegister(@ModelAttribute UserVO uvo) {
    int count = regnlogSrv.checkSameId(uvo.getUserId()); // 아이디중복확인

    // 비밀번호 암호화
    String pwd = pwdEncoder.encode(uvo.getUserPwd());
    uvo.setUserPwd(pwd);

    if(count > 0) {
        // 아이디가 중복
        return "failure";
    }else {
        // 아이디 유일 -> 회원가입 가능
        regnlogSrv.setRegister(uvo);
        return "success";
    }
}
```

```xml
<select id="checkSameId" parameterType="String" resultType="int">
  SELECT count(id) FROM user
  WHERE user_id = #{userId}
</select>
<insert id="setRegister" parameterType="uservo">
  INSERT INTO user SET
  user_id = #{userId},
  user_pwd = #{userPwd},
  user_name = #{userName},
  user_gender = #{userGender},
  user_phone = #{userPhone},
  user_email = #{userEmail},
  user_regdate = now(),
  user_auth = 1
</insert>
```

3. 주요 로직 설명 로그인/로그인 인터셉터




```java
@Override
public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
        throws Exception {
    HttpSession session = request.getSession();
    response.setContentType("text/html; charset=UTF-8");

    String path = request.getRequestURI();
    String query = request.getQueryString();
    if( query != null ) {
        path = path + "?" + query;
    }

    logger.info("[LoginInterceptor] path = " + path);

    if( session.getAttribute("userName") == null ) {
        PrintWriter out = response.getWriter();
        out.append("<script>alert('로그인이 필요한 서비스입니다');");
        out.append("location.href='login';");
        out.append("</script>").flush();
        out.close();
        return false;
    }

    return true;
```

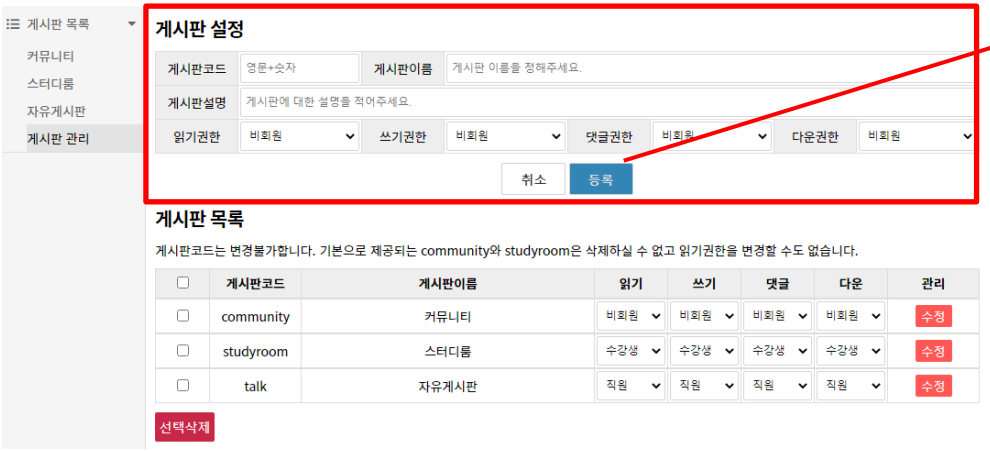
```xml
<interceptors>
    <interceptor>
      <mapping path="/articleView" />
      <mapping path="/articleWrite" />
      <mapping path="/articleDownload" />
      <mapping path="/boardSetting" />
      <beans:ref bean="loginInter" />
    </interceptor>
</interceptors>
```

```java
@RequestMapping(value = "/login", )
public ModelAndView loginCheck(@ModelAttribute UserVO uvo,HttpSession session) {
    ModelAndView mav = new ModelAndView();

    String pwd = regnlogSrv.getPassword(uvo);
    boolean pwdMatch = pwdEncoder.matches(uvo.getUserPwd(), pwd);

    if( pwd != null && pwdMatch == true  ) {
        uvo.setUserPwd(pwd);
        regnlogSrv.loginCheck(uvo, session);
        mav.setViewName("redirect:/main");
    }else {
        mav.addObject("msg", "아이디/비밀번호를 확인하세요.");
        mav.setViewName("/login");
    }

    return mav;
}
```

유저가 로그인 요청 시 **form-submit** 방식으로 form에 입력된 데이터를 컨트롤러로 보냅니다. 로그인 성공할 경우, 고객정보를 세션에 저장한 후 메인페이지로 이동시킵니다. 로그인 실패 시에는 화면에 메시지를 띄워줍니다.

만약 유저가 **로그인 하지 않고** 글보기, 글쓰기, 다운로드, 게시판관리의 기능을 이용하려고 URL 요청을 보낼 때, 요청 URL을 가로채 **LoginInterceptor** 클래스에 보내어 로그인 페이지를 반환합니다.

3. 주요 로직 설명 게시판 관리(관리자)_1



관리자 권한을 가진 유저만 게시판 관리 메뉴를 이용할 수 있습니다. 게시판 설정의 테이블을 모두 채운 후 등록 버튼을 누르면 데이터베이스에 테이블 두 개가 추가됩니다.

예를 들어 게시판 코드가 'talk'인 게시판을 등록할 경우, 'article_talk' (게시판), 'comment_talk' (댓글) 테이블이 생성되며 'board' 테이블의 행에 게시판 설정 정보값이 추가됩니다.)

```
$("#btn_submit").click(function () {

    // 1. 게시판코드 유효성검사
    var codeReg = /^[A-Za-z0-9+]{1,12}$/; // 영문대소문자+숫자 1~12자

    if (!codeReg.test(document.getElementById('boardCode').value)) {
        alert("게시판코드는 영문과 숫자를 사용하여 1~12자 사이여야 합니다.");
        $("#boardCode").val('');
        $("#boardCode").focus();
        return false;
    }

    // 2. 공백제거 후, 아무것도 쓰여있지않으면 ajax로 넘기지않음 ->
    if ($.trim($("#boardCode").val()) == '') {
        $("#boardCode").val();
        $("#boardCode").focus();
        return false;
    }

    if ($.trim($("#boardName").val()) == '') {
        $("#boardName").val();
        $("#boardName").focus();
        return false;
    }

    if ($.trim($("#boardDesc").val()) == '') {
        $("#boardDesc").val();
        $("#boardDesc").focus();
        return false;
    }
```

3. 주요 로직 설명 게시판 관리(관리자)_2

```javascript
$.ajax({
  url: "/academy/setBoard",
  type: "POST",
  data: $("#board_frm").serialize(),
  success: function (resData) {
    if (resData == 'failure') {
      alert($("#boardCode").val() + " 게시판코드가 중복되었습니다.\n게시판코드를 변경해주세요.");
      $("#boardCode").val('');
      $("#boardCode").focus();
      return false;
    } else {
      alert($("#boardCode").val() + "게시판 & 댓글게시판이 생성되었습니다.");
      window.location.href = "/academy/boardSetting";
    }
  },
  error: function () {
    alert("ajax error!");
  }
});
```

```java
@RequestMapping(value = "/setBoard", method = RequestMethod.POST)
@ResponseBody
public String setBoard(@ModelAttribute BoardVO bvo) {
    // 1. 게시판코드,게시판이름,게시판설명의 앞뒤공백제거 후 저장
    bvo.setBoardCode(bvo.getBoardCode().trim());
    bvo.setBoardName(bvo.getBoardName().trim());
    bvo.setBoardDesc(bvo.getBoardDesc().trim());

    // 2. 게시판코드가 중복이 아니라면..
    // 게시판 테이블(board)에 게시판 설정 저장 후, Repository에서
    // 해당 게시판코드의 게시글테이블 생성(테이블명 : article_boardCode),
    // 해당 게시판코드의 댓글테이블 생성(테이블명 : comment_boardCode)
    String result = boardSrv.setBoard(bvo);

    // failure : 게시판코드 중복
    // success : 게시판설정저장+게시글테이블생성+댓글테이블생성 완료
    return result;
}
```

```java
@Override
public String setBoard(BoardVO bvo) {

    // boardCode 중복 확인( 결과가 0이 아니면 중복 )
    int result = sqlSession.selectOne("board.checkCode", bvo.getBoardCode());

    if( result > 0 ) {
        return "failure";
    }else {
        // 1. 게시판 설정 저장
        sqlSession.insert("board.setBoard", bvo);

        // 2. 게시글게시판 생성
        String articleStr = "CREATE TABLE article_" + bvo.getBoardCode();
        articleStr += "(id int not null auto_increment primary key,";
        articleStr += "article_category char(1) not null,";
        articleStr += "article_subject varchar(100) not null,";
        articleStr += "article_writer int not null,";
        articleStr += "article_content text,";
        articleStr += "article_regdate datetime,";
        articleStr += "article_hit int default 0,";
        articleStr += "article_likes int default 0,";
        articleStr += "article_secret boolean not null,";
        articleStr += "article_file_name varchar(300),";
        articleStr += "article_file_origin varchar(300),";
        articleStr += "article_file_url varchar(500),";
        articleStr += "article_ref int,";
        articleStr += "article_level int,";
        articleStr += "article_layer int);";

        Map<String, String> map = new HashMap<String, String>();
        map.put("articleStr", articleStr);
        sqlSession.insert("board.setArticleTbl", map);
```

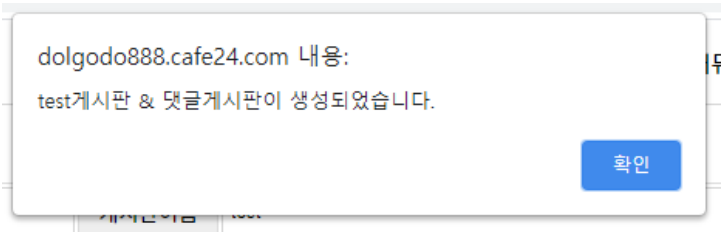
게시판 코드가 중복인지 확인 후, 중복이 아니라면 게시판 생성을 시작합니다. 첫째, 게시판 테이블(board)에 해당 게시판 설정 데이터를 추가합니다. 다음으로는 해당 게시판의 게시글들을 저장해줄 테이블(article_**)을 생성합니다.

3. 주요 로직 설명 게시판 관리(관리자)_3

```
// 3. 댓글게시판 생성
String cmntStr = "CREATE TABLE comment_" + bvo.getBoardCode();
cmntStr += "(id int not null auto_increment primary key,";
cmntStr += "article_id int,";
cmntStr += "cmnt_content text,";
cmntStr += "cmnt_writer int,";
cmntStr += "cmnt_ref int,";
cmntStr += "cmnt_level int,";
cmntStr += "cmnt_layer int,";
cmntStr += "cmnt_regdate datetime);";

map.put("cmntStr", cmntStr);
sqlSession.insert("board.setCommentTbl", map);

return "success";
```

댓글 테이블(comment_**)을 생성합니다.
게시판 설정 추가, 게시글 테이블 생성, 댓글 테이블 생성이 끝나면 컨트롤러는 success를 반환합니다. 만일 게시판 코드가 중복이 된다면 failure를 반환합니다.
이후 JS는 컨트롤러로부터 전달받은 값에 따라 적절한 alert 창을 띄웁니다.

```
<insert id="setBoard" parameterType="boardvo">
  INSERT board SET
    board_code = #{boardCode},
    board_name = #{boardName},
    board_desc = #{boardDesc},
    board_read = #{boardRead},
    board_write = #{boardWrite},
    board_comment = #{boardComment},
    board_download = #{boardDownload}
</insert>

<insert id="setArticleTbl" parameterType="Hashmap">
  ${articleStr}
</insert>

<insert id="setCommentTbl" parameterType="Hashmap">
  ${cmntStr}
</insert>
```



3. 주요 로직 설명 댓글 작성


```
$("#comment_btn").click(function(){

  if( $("#userId").val() == '0' ){
    alert("비회원은 댓글을 작성할 수 없습니다.");
    $("#txt_comment").val('');
    return false;
  }

  if( $.trim($("#txt_comment").val()) == '' ){
    alert("댓글 내용을 작성해주세요.");
    $("#txt_comment").val('');
    $("#txt_comment").focus();
    return false;
  }

  if( $.trim($("#txt_comment").val()) == 'deleted' ){
    alert("잘못된 댓글 내용입니다.");
    $("#txt_comment").val('');
    $("#txt_comment").focus();
    return false;
  }

  var formData = $("#cmntForm").serialize();
  $.ajax({
    url : "/academy/setComment",
    type : "POST",
    data : formData,
    success : function(resData) {
      commentList();
      $("#txt_comment").val('');
    },
    error : function(){
      alert("댓글 입력 오류발생");
    }
  });
});
```

```
<insert id="setComment" parameterType="commentvo">
  <selectKey resultType="int" keyProperty="cmntRef" order="BEFORE" >
    SELECT IFNULL(MAX(id), 0) + 1 FROM comment_${boardCode}
  </selectKey>
  INSERT comment_${boardCode} SET
    article_id = #{articleId},
    cmnt_content = #{cmntContent},
    cmnt_writer = #{cmntWriter},
    cmnt_ref = #{cmntRef},
    cmnt_level = 0,
    cmnt_layer = 0,
    cmnt_regdate = now()
</insert>
```

댓글을 등록하면 AJAX를 이용해 입력된 댓글의 내용과 유저정보를 가지고 URL 요청을 보냅니다. URL이 일치하는 컨트롤러를 거쳐서 DB에 댓글의 정보를 저장한 후 JS로 success를 반환합니다. 이후 JS는 댓글의 목록만을 다시 불러오고 댓글 입력창을 비워줍니다.

3. 주요 로직 설명 댓글 목록_1

```
function commentList(){
  var formData = $("#cmntForm").serialize();

  $.ajax({
    url : "/academy/getCommentList.do",
    type : "POST",
    data : formData,
    success : function(resData) {
      var a = '';
      a += '<p>댓글 <span class="count">'+ resData.count + '</span></p>';

      var uid = $("#userId").val();
      var userAuth = $("#userAuth").val();

      var b = '';
      $.each(resData.list, function(key,value) {

        var writer = String(value.cmntWriter);
        if( userAuth >= 3 && uid != writer ) {

          b += '<div class="comment_section p_tb10">';
            b += '<div class="flex justify_c_between">';

              b += '<div class="white_s_no">';
                for(var i=0;i < value.cmntLayer ;i++ ) {
                  b += '<img src="/academy/img/icon_reply.gif" style="display:inline-block;"/>';
                }
              b += '</div>';

              b += '<div class="w_100">';
                b += '<div class="cmt_head m_b5">';
                  b += '<span class="p_r2">'+ value.cmntWriterName +'</span>';
                  b += '<span class="txt_regdate">'+ value.cmntRegdate +'</span>';
                  b += '<span class="a_wrap">';
                    b += '<a class="p_r5" href="javascript:void(0);" onClick="commentReply('+ value.cid +');">답글</a>';
                    b += '<a href="javascript:void(0);" onClick="commentDelete('+ value.cid +');">삭제</a>';
                  b += '</span>';
                b += '</div>';
```

게시글 화면을 불러올 때 JS로 해당 게시글의 댓글 목록을 불러옵니다. 댓글폼(cmntForm)에 숨겨진 유저와 게시글의 정보를 토대로 해당 게시글의 댓글 목록을 AJAX 방식으로 가져옵니다.

만약 유저의 권한이 3(직원)보다 상위라면 댓글을 삭제할 수 있으나 수정은 불가합니다. 직원은 아니지만 해당 댓글의 작성자라면 삭제와 수정 둘다 가능합니다. 이도 아니라면 답글만 달 수 있습니다.

또한 댓글이 삭제되면 없어지는 것이 아니라 '삭제된 댓글입니다' 라고 표시됩니다. 대댓글에 대한 혼동을 피하기 위함입니다.

3. 주요 로직 설명 댓글 목록_2

```
} else if( uid != writer ){

    b += '<div class="comment_section p_tb10">';
     b += '<div class="flex justify_c_between">';
      b += '<div class="white_s_no">';
       for(var i=0;i < value.cmntLayer ;i++ ) {
        b += '<img src="/academy/img/icon_reply.gif" style="display:inline-block;"/>';
       }
      b += '</div>';
      b += '<div class="w_100">';
       b += '<div class="cmt_head m_b5">';
        b += '<span class="p_r2">'+ value.cmntWriterName +'</span>';
        b += '<span class="txt_regdate">'+ value.cmntRegdate +'</span>';
        b += '<span class="a_wrap">';
         b += '<a class="p_r5" href="javascript:void(0);" onClick="commentReply('+ value.cid +');">답글</a>';
        b += '</span>';
       b += '</div>';
       if( value.cmntContent == 'deleted' ) {
        b += '<p>삭제된 댓글입니다.</p>';
       }else {
        b += '<p>'+ value.cmntContent +'</p>';
       }
      b += '</div>';
     b += '</div>';
     b += '<div class="commentMod'+ value.cid +'"></div>';
    b += '</div>';
```

세션의 유저가 해당 **댓글의 작성자가 아니라면** 수정/삭제 불가하며 답글(=대댓글)작성만 가능합니다.

3. 주요 로직 설명 댓글 목록_3

```
} else {

    b += '<div class="comment_section p_tb10">';
      b += '<div class="flex justify_c_between">';
        b += '<div class="white_s_no">';
        for(var i=0;i < value.cmntLayer ;i++ ) {
          b += '<img src="/academy/img/icon_reply.gif" style="display:inline-block;"/>';
        }
        b += '</div>';
        b += '<div class="w_100">';
          b += '<div class="cmt_head m_b5">';
            b += '<span class="p_r2">'+ value.cmntWriterName +'</span>';
            b += '<span class="txt_regdate">'+ value.cmntRegdate +'</span>';
            b += '<span class="a_wrap">';
              b += '<a class="p_r5" href="javascript:void(0);" onClick="commentReply('+ value.cid +');">답글</a>';
              b += '<a class="p_r5" href="javascript:void(0);" onClick="commentModify('+ value.cid +',\''+ value.cmntContent +'\')">수정</a>';
              b += '<a href="javascript:void(0);" onClick="commentDelete('+ value.cid +');">삭제</a>';
            b += '</span>';
          b += '</div>';
          if( value.cmntContent == 'deleted' ) {
            b += '<p>삭제된 댓글입니다.</p>';
          }else {
            b += '<p>'+ value.cmntContent +'</p>';
          }
        b += '</div>';
      b += '</div>';
      b += '<div class="commentMod'+ value.cid +'"></div>';
    b += '</div>';
  }

});
```

세션의 유저가 해당 **댓글의 작성자라면** 답글/수정/삭제 모두 가능합니다.
For문이 끝나고 댓글 목록과 댓글 개수를 담은 변수 a,b,c를 화면에 보여줍니다.

```
  var c = '';
  c += '댓글<span class="m_l3 count">'+ resData.count +'</span>';
  $("#comment_list").html(a+b);
  $("#commentCount").html(c);
```

3. 주요 로직 설명 댓글의 답글(대댓글)_1

```
function commentReply(cid){
 var a = '';

 a += '<form class="comment_post m_t20" autocomplete="off">';
    a += '<textarea name="commentReply_'+ cid +'" placeholder="내용을 입력해주세요."></textarea>';
    a += '<div class="right">';
       a += '<button type="button" class="comment_btn" onClick="commentReplyProc('+ cid +')">등록</button>';
    a += '</div>';
 a += '</form>';
 $('.commentMod' + cid).html(a);

}
```

답글을 달고자하는 댓글의 '답글'버튼을 누르면 답글을 입력할 수 있는 textarea가 나타납니다.

답글을 입력하면 해당 답글의 form 데이터가 요청URL에 맞는 컨트롤러로 전송됩니다.

```
function commentReplyProc(cid) {
 var replyContent = $('[name=commentReply_' + cid + ']').val();

 if( $.trim(replyContent) == '' ) {
  alert('내용을 입력해주세요.');
  $('[name=commentReply_' + cid + ']').val('');
  $('[name=commentReply_' + cid + ']').focus();
 }

 var formData = {
     cid : cid,
     articleId : $("#articleId").val(),
     cmntWriter : $("#userId").val(),
     boardCode : $("#boardCode").val(),
     cmntContent : replyContent
 };

 $.ajax({
  url : "/academy/commentReply.do",
  type : "POST",
  data : formData,
  success : function(resData) {
   commentList();
  },
  error : function(){
   alert("답글 등록 중 오류가 발생했습니다.\n관리자에게 문의하세요.");
  }
 });

}
```

3. 주요 로직 설명 댓글의 답글(대댓글)_2

```java
@RequestMapping("/commentReply")
@ResponseBody
public String setCommentReply(@ModelAttribute CommentVO cvo) {
    commentSrv.setCommentReply(cvo);
    return "success";
}

@Override
public void setCommentReply(CommentVO cvo) {
    commentDao.setCommentReply(cvo);
}

@Override
public void setCommentReply(CommentVO cvo) {
    CommentVO vo = sqlSession.selectOne("comment.getReplyInfo", cvo);
    cvo.setCmntRef(vo.getCmntRef());
    cvo.setCmntLevel(vo.getCmntLevel());
    cvo.setCmntLayer(vo.getCmntLayer());

    sqlSession.update("comment.setCommentLevel", cvo);
    sqlSession.insert("comment.setCommentReply", cvo);
}
```

```xml
<select id="getReplyInfo" parameterType="commentvo" resultType="commentvo">
  SELECT
    cmnt_ref AS cmntRef,
    cmnt_level AS cmntLevel,
    cmnt_layer AS cmntLayer
  FROM comment_${boardCode}
  WHERE id = #{cid}
</select>

<update id="setCommentLevel" parameterType="commentvo">
  UPDATE comment_${boardCode} SET
    cmnt_level = cmnt_level + 1
  WHERE cmnt_ref = #{cmntRef} AND cmnt_level > #{cmntLevel}
</update>

<insert id="setCommentReply" parameterType="commentvo">
  INSERT comment_${boardCode} SET
    article_id = #{articleId},
    cmnt_content = #{cmntContent},
    cmnt_writer = #{cmntWriter},
    cmnt_ref = #{cmntRef},
    cmnt_level = #{cmntLevel} + 1,
    cmnt_layer = #{cmntLayer} + 1,
    cmnt_regdate = now()
</insert>
```

DAO단에서 답글을 달고자하는 원댓글의 정보를 가져옵니다. (getReplyInfo)
여기서 cmnt_ref 는 참고하는 댓글의 기본키(id), cmnt_level은 댓글 그룹의 순서, cmnt_layer는 댓글의 계층을 나타냅니다.
새로운 답글일수록 상단에 위치하기 때문에 UPDATE문으로 댓글 그룹의 순서를 업데이트 해줍니다.
댓글의 답글 등록이 완료되면 JS에서는 댓글의 목록을 다시 불러와 업데이트 합니다.

4. 보완점 및 후기

처음부터 끝까지 혼자서 고민하고 해결해보고 싶어 시작하게 된 프로젝트였습니다. 그렇기 때문에 이전 팀 프로젝트에서 하지 못했던 것들을 적용시켜보고 싶었습니다. 팀 프로젝트 당시 게시판을 다른 팀원분이 담당해서 개인 프로젝트에서는 직접 검색, 페이징, 게시글 및 댓글 작성을 만들었습니다. 또한 쿠키를 이용한 로그인정보 저장, 로그인 인터셉터, 암호화된 비밀번호, 추천기능, 한번에 볼 수 있는 게시글의 수 지정 등이 있었습니다.

게시판기능 구현에만 초점을 두고 완성시켰기 때문에 미완성된 부분이 많이 있습니다. 회원관리기능은 팀에서의 경험으로 쉽게 구현할 수 있을 것이라 생각해서 이번 프로젝트에서는 후순위로 미뤄두고 진행했습니다.

이전에는 DB에 유저의 비밀번호가 그대로 저장되었기 때문에 보안에 좋지 않다고 생각되어 스프링 시큐리티를 이용해 암호화하여 저장하였지만 보다 더 나은 방법이 있을 것 같아 추후에 여유가 된다면 이것을 보완하고 싶습니다. 그리고 댓글을 AJAX로 구현하였지만 추천 기능은 AJAX로 구현하지 않아 페이지 이동이 발생하여 조회수가 증가하는 문제가 있어 이를 AJAX로 변경하여야겠다고 생각하고 있습니다.

개인 프로젝트를 진행하면서 스스로 문제를 해결하는 능력을 기를 수 있었습니다.
앞으로도 능동적으로 일하는 개발자가 되기 위해 노력하겠습니다. 감사합니다.

GitHub : https://github.com/pmj-jung/portfolio/tree/main/Academy_Project
실제 사이트 : http://dolgodo888.cafe24.com/academy/