게시판
- 댓글 시스템 -

댓글 시스템


댓글 달기 예제 입니다.
작성자: sking/ 조회수: 178/ 수정일: 2018-10-22
댓글 달기 예제 입니다.
댓글을 달아보세요.
[댓글 달기] 작성자 : sking
비밀번호 :
등록
취소
admin
test1
작성일 : 2018-10-22
sking
test
작성일 : 2018-10-22
댓글은 계층 구조를 가짐
- 트리 구조 -
sking
삭제된 글입니다.
작성일 : 2018-10-22
sking
test
작성일 : 2018-10-22

댓글 시스템

❖ 계층형 쿼리 : START WITH … CONNECT BY

- 트리 구조의 내용 처리

```
SELECT expr1, expr2, ...
FROM 테이블
WHERE 조건
START WITH[최상위 조건]
CONNECT BY [NOCYCLE][PRIOR 계층형 구조 조건];
ORDER SIBLINGS BY [정렬 필드]
```

- START WITH 조건
  - 최상위 계층의 로우를 식별하는 조건
- CONNECT BY 조건
  - 계층형 구조가 어떤 식으로 연결되는지 기술
  - 부모에 대한 참조 필드와 연결되는 필드명 앞에 PRIOR 키워드 부여
  - connect by manager_id = prior employee_id
- level이라는 의사 컬럼 생성됨
  - 계층 구조에서 레벨(깊이)을 의미
- ORDER SIBLING BY
  - 같은 level의 행들의 정렬 기준 설정

댓글 시스템

❖ HR 계정의 EMPLOYEES 테이블

- MANAGER_ID
  - 자신의 상사 직원 ID
  - 조직 계층도 구성

```
select lpad(' ', (level-1)*4) || last_name
from hr.employees
connect by manager_id = prior employee_id
start with manager_id is null
order siblings by hire_date;
```

댓글 시스템

❖ 댓글 테이블 정의

```
CREATE TABLE REPLIES(
   REPLY_ID       NUMBER PRIMARY KEY,
   BOARD_ID       NUMBER,        -- 연관 게시글 번호
   PARENT         NUMBER,        -- 상위 댓글 번호
   WRITER         VARCHAR2(20) NOT NULL,
   CONTENT        VARCHAR2(1024),
   DELETED        NUMBER(1),     -- 삭제 여부, 실제 삭제하지 않음
   REG_DATE       DATE,
   UPDATE_DATE    DATE,
   CONSTRAINT F_REPLIES_BOARD FOREIGN KEY(BOARD_ID)
      REFERENCES BOARDS(BOARD_ID),
   CONSTRAINT F_REPLIES_MEMBER FOREIGN KEY(WRITER)
      REFERENCES MEMBERS(USER_ID)
);
```

```
CREATE SEQUENCE REPLIES_SEQ;
```

댓글 시스템

❖ 모델 : Rely

```
@Data
public class Reply {
   private int    replyId;    // 댓글 번호
   private int    boardId;    // 글 그룹(게시글 ID)
   private int    level;      // 댓글 수준
   private  int   parent;     // 상위 글 번호
   private String writer;     // 작성자 ID
   private String content;    // 내용
   private int    deleted;    // 삭제여부
   private Date   regDate;
   private Date   updateDate;
}
```

❖ ReplyDao : 매퍼 인터페이스

```
public interface ReplyDao {

   @Select("select count(*) from reply where board_id=#{boardId}")
   int count(long boardId) throws Exception;

   @Select("select * from reply where reply_id=#{replyId} ")
   Reply findById(long replyId) throws Exception;

   @Select({
      "select board_id, reply_id, level, parent, writer, content,
deleted, update_date",
      "from reply",
      "where board_id = #{boardId}",
      "start with parent = 0",
      "connect by parent = prior reply_id ",
      "order siblings by reply_id desc"
   })
   List<Reply> getList(long boardId) throws Exception;
```

❖ ReplyDao : 매퍼 인터페이스

```
   @Insert({
      "insert into reply(reply_id, board_id, password, parent, writer,
content)",
      "values(reply_seq.nextval, #{boardId}, #{password}, #{parent},
#{writer}, #{content})"
   })
   @Options(useGeneratedKeys=true,
            keyColumn="reply_id", keyProperty = "replyId")
   int insert(Reply reply) throws Exception;

   @Update({
      "update reply set",
      "  content = #{content}, deleted = 1, update_date = sysdate",
      "where reply_id=#{replyId}  and password = #{password}"
   })
   int update(Reply reply) throws Exception;
```

❖ ReplyDao : 매퍼 인터페이스

```
   @Delete({
      "update reply set",
      "  content = '삭제된 글', deleted = 1, update_date = sysdate",
      "where reply_id=#{replyId}  and password = #{password}"
   })
   int delete(Reply reply) throws Exception;

}
```

❖ root-context.xml

```
<bean id="sqlSessionFactory"
      class="org.mybatis.spring.SqlSessionFactoryBean">
   <property name="dataSource" ref="dataSource" />
   <property name="typeAliasesPackage"
           value="edu.iot.common.model" />
   <property name="configLocation"
           value="classpath:config/mybatis-config.xml"/>
   <property name="mapperLocations">
      <list>
         <value>edu.iot.**.dao</value>
         <value>classpath:mapper/**/*-mapper.xml</value>
      </list>
   </property>
</bean>
```

❖ 서비스 인터페이스 : ReplyService

```
public interface ReplyService {

   List<Reply> getList(long boardId) throws Exception;

   Reply findById(long replyId) throws Exception;

   Reply create(Reply reply) throws Exception;

   Reply update(Reply reply) throws Exception;

   void delete(Reply reply) throws Exception;

}
```

❖ 서비스 인터페이스 구현 : ReplyServiceImpl

```
@Service
public class ReplyServiceImpl implements ReplyService {
   @Autowired
   ReplyDao dao;

   @Override
   public List<Reply> getList(long boardId) throws Exception {
      return dao.getList(boardId);
   }

   @Override
   public Reply findById(long replyId) throws Exception {
      return dao.findById(replyId);
   }

   @Override
   public Reply create(Reply reply) throws Exception {
      dao.insert(reply);
      return dao.findById(reply.getReplyId());
   }
```

❖ 서비스 인터페이스 구현 : ReplyServiceImpl

```
   @Override
   public Reply update(Reply reply) throws Exception {
      dao.update(reply);
      if(dao.update(reply)==0)
         throw new PasswordMissmatchException();
      return dao.findById(reply.getReplyId());
   }

   @Override
   public void delete(Reply reply) throws Exception {
      if(dao.delete(reply)==0)
         throw new PasswordMissmatchException();
   }
}
```

댓글 시스템

❖ BoardController

```
@Controller
@RequestMapping("/board")
public class BoardController {
   @Autowired
   BoardService service;

   @Autowired
   ReplyService replyService;
      :
   @RequestMapping(value="/view/{boardId}", method=RequestMethod.GET)
   public String view(@PathVariable int boardId,
                      Model model)
                      throws Exception {
      Board board = service.findById(boardId);
      model.addAttribute("board", board);
      model.addAttribute("replies", replyService.getList(boardId));

      return "board/view";
   }
   :
```

댓글 시스템

❖ reply REST API

- ○ url 설계

URI	METHOD	의미
/board/reply/{boardId}	GET	boardId 게시글의 댓글 목록
/board/reply/{boardId}/{replyId}	GET	replyId의 댓글
/board/reply/{boardId}	POST	댓글 생성
/board/reply/{boardId}/{replyId}	PUT	replyId의 댓글 수정
/board/reply/{boardId}/{replyId} ?password=	DELETE	replyId의 댓글 삭제

❖ REST 컨트롤러 : ReplyApiController

```
@RestController
@RequestMapping("/board/reply/{boardId}")
public class ReplyController {
   @Autowired
   ReplyService service;

   // 정상 처리 응답
   public <T> ResponseEntity<T> getResult(T t) {
      final HttpHeaders headers = new HttpHeaders();
      headers.add("Content-Type", "application/json;charset=UTF-8");
      return new ResponseEntity<T>(t, headers, HttpStatus.OK);
   }
   // 에러 처리 응답
   public <T> ResponseEntity<T> handleError(Exception e) {
      e.printStackTrace();
      final HttpHeaders headers = new HttpHeaders();
       headers.add("Content-Type", "application/json;charset=UTF-8");
      return new ResponseEntity<T>(null, headers,
                                   HttpStatus.INTERNAL_SERVER_ERROR);
   }
```

❖ REST 컨트롤러 : ReplyApiController

```
@RequestMapping(method=RequestMethod.GET)
public ResponseEntity<List<Reply>> list(
                                @PathVariable long boardId){
   try {
      List<Reply> list = service.getList(boardId);
      return getResult(list);
    } catch (Exception e) {return handleError(e);}  }

@RequestMapping(value="/{replyId}", method=RequestMethod.GET)
public ResponseEntity<Reply> replyId(
                            @PathVariable long replyId){
   try {
      Reply reply= service.findById(replyId);
      return getResult(reply);
    } catch (Exception e) {return handleError(e);}
}
```

❖ REST 컨트롤러 : ReplyApiController

```
@RequestMapping(method=RequestMethod.POST)
public ResponseEntity<Reply> create(@RequestBody Reply reply){
   // 데이터가 json으로 전송되므로 @RequestBody 사용
   try {
      Reply r= service.create(reply);
      return getResult(r);
    } catch (Exception e) {return handleError(e);}
}

@RequestMapping(value="/{replyId}", method=RequestMethod.PUT)
public ResponseEntity<Reply> update(@RequestBody Reply reply){
   // 데이터가 json으로 전송되므로 @RequestBody 사용
   try {
      System.out.println(reply);
      Reply r= service.update(reply);
      return getResult(r);
    } catch (Exception e) {return handleError(e);}
}
```

❖ REST 컨트롤러 : ReplyApiController

```
   @RequestMapping(value="/{replyId}", method=RequestMethod.DELETE)
   public ResponseEntity<Reply> delete(Reply reply){
      // password가 쿼리 파라미터로 전송됨
      // @RequestBody 사용하지 않음
      try {
         service.delete(reply);
         return getResult(reply);
      } catch (Exception e) {return handleError(e);}
   }
}
```

댓글 시스템


댓글 달기 예제 입니다.
작성자: sking/ 조회수: 178/ 수정일: 2018-10-22
댓글 달기 예제 입니다.
댓글을 달아보세요.
[댓글 달기] 작성자 : sking
비밀번호 :
등록
취소
admin
test1
작성일 : 2018-10-22
sking
test
작성일 : 2018-10-22
sking
삭제된 글입니다.
작성일 : 2018-10-22
sking
test
작성일 : 2018-10-22

댓글 시스템

❖ 자바스크립트 준비

- resoruces/js/reply.js
  - 댓글 처리용 자바스크립트 파일 생성

```
// 최상위 댓글 달기 jQuery 플러그인
$.fn.replyForm = function(listPanel) {

}

// 댓글 목록 관리 jQuery 플러그인
$.fn.replyList = function(boardId) {

}
```

❖ 뷰 기본 골격 : view.jsp

```
<script src="${context}/resources/js/reply.js"></script>

<script>
// 전역 변수
var context = '${context}';  // 자바스크립트에서 사용할 컨텍스트 경로명
var user = '${USER.userId}'; // 현재 사용자 ID

$(function(){
   // 기존 코드
      :
   $('#reply-form').replyForm($('#reply-list'));
   $('#reply-list').replyList(${board.boardId});
});
</script>

   :
```

❖ 뷰 기본 골격 : view.jsp

```
          :
<div>
   ${board.content}
</div>

<c:if test="${not empty USER}">
   <jsp:include page="reply-form.jsp"/>
</c:if>

<div id="reply-list" class="mt-5">
   <jsp:include page="reply-list.jsp"/>
</div>

   :
```

댓글 쓰기 영역 :
로그인 사용자만 활성화

댓글 쓰기 영역 :
로그인 사용자만 활성화

댓글 시스템

❖ reply-form.jsp

댓글 시스템

❖ reply-form.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<div class="card card-body mp-3">
   <form id="reply-form"
      action="${context}/board/reply/${boardId}" method="post">
      <input type="hidden" name="boardId" value="${boardId}">
      <input type="hidden" name="writer" value="${USER.userId}">
      <div class="mb-2">
         <strong>[댓글 달기] </strong>
         작성자 : ${USER.userId}
      </div>
      <textarea name="content" required
         style="width:100%"></textarea>
      <div class="text-right">
         비밀번호 : <input type="password" name="password" required>
         <button type="submit">등록</button>
         <button type="reset">취소</button>
      </div>
   </form>
</div>
```

댓글 시스템

❖ reply-list.jsp

- media 클래스


```
<div class="media my-3">
   <img src="…" class="rounded-circle avata-md">

   <div class="media-body ml-3">
      <h5>타이틀</h5>
      <div>
         내용
      </div>
   </div>
</div>
```

❖ reply-list.jsp

댓글 시스템

❖ reply-list.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>

<c:forEach var="reply" items="${replies}">
   <div class="media my-3"
      style="margin-left:${50*(reply.level-1)}px"
      data-reply-id="${reply.replyId}"
      data-board-id="${reply.boardId}"
      data-parent="${reply.replyId}"
      data-level="${reply.level}">

      <img src="${context}/member/avata/${reply.writer}"
         class="rounded-circle avata-md">
```

댓글 시스템

❖ reply-list.jsp

```
<div class="media-body ml-3">
   <div class="button-group float-right">
      작성일 :
      <span class="update-date">
         <fmt:formatDate value="${reply.updateDate}"
                         pattern="yyyy-MM-dd"/>
      </span>
      <c:if test="${reply.deleted != 1}">
         <c:if test="${not empty USER}">
            <button class="reply-add-show">
               <i class="fa fa-reply"></i></button>
            <c:if test="${reply.writer == USER.userId}">
               <button class="reply-edit-show">
                  <i class="fa fa-edit"></i></button>
               <button class="reply-delete-show">
                  <i class="fa fa-trash"></i></button>
            </c:if>
         </c:if>
      </c:if>
   </div>
```

삭제글이 아닌 경우에 만

자신이 작성한 댓글인 경우에 만

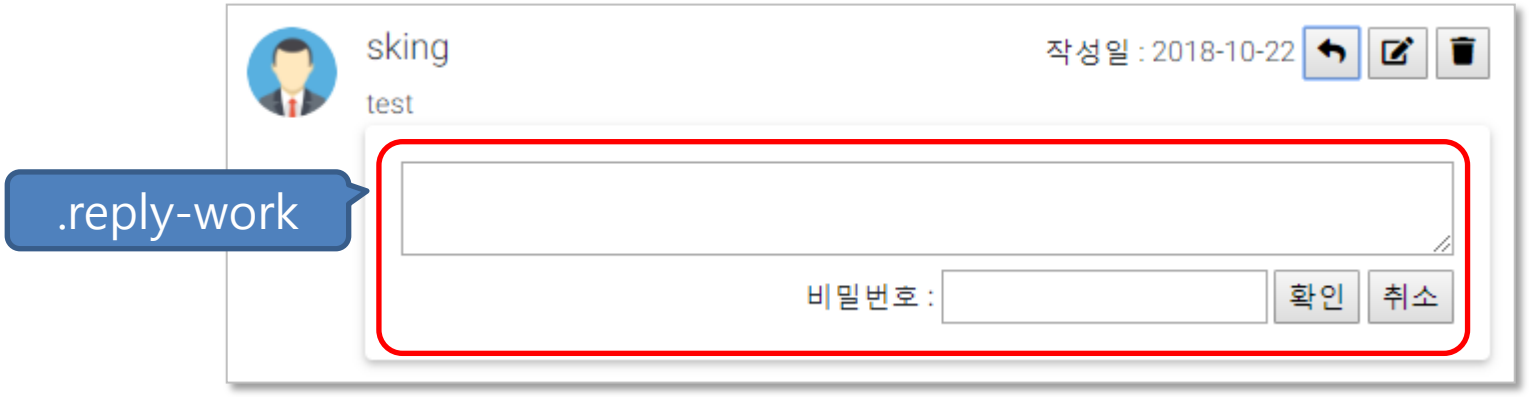
댓글 시스템

❖ reply-list.jsp

```
            <h5>${reply.writer}</h5>

            <div class="reply-content">${reply.content}</div>

            <div class="reply-work"></div>
        </div>
    </div>
</c:forEach>
```



- .reply-work 영역은 추가/수정/삭제 버튼 클릭 시 내용 추가
- 처리 성공 및 취소 시 제거

게시판
- 최상위 댓글 달기 -

❖ reply.js : 최상위 댓글 추가

- replyForm 플러그인 처리
  - $(선택자).replyForm(댓글목록 영역 jQuery 객체);

```
$('#reply-form').replyForm($('#reply-list'));
```

❖ reply.js : 최상위 댓글 추가 / 공통 기능 함수

```
// 날짜 출력
Date.prototype.formatDate= function() {
   var year = this.getFullYear();
   var month = this.getMonth()+1;
   var date = this.getDate();

   month = (month < 10) ? '0' + month : month;
   date = (date < 10) ? '0' + date : date;

   return `${year}-${month}-${date}`;
}

// 폼 데이터를 JS 객체로 변환
function formToObject(form) {
   var arr = $(form).serializeArray();
   var obj = {}
   for(entry of arr) {
      obj[entry.name] = entry.value;
   }
   return obj;
}
```

❖ reply.js : 최상위 댓글 추가

```
// ajax 에러 처리 핸들러
function error(xhr){
   alert(`요청 처리 실패(${xhr.status}) : ${xhr.responseText}`);
}

// ajax 호출 메서드
function replyAjax(obj) {
   $.ajax({
      url : obj.url,              // ajax 요청 url
      type : obj.type,            // ajax 요청 HTTP 메서드
      contentType: 'application/json',  // 전송 컨텐츠 인코딩 타입
      data : obj.data,            // 전송 데이터
      dataType : 'json',          // 결과 포맷
      success : obj.success,      // 성공 시 콜백
      error : error               // 에러 시 콜백
   });
}
```

❖ reply.js : 최상위 댓글 추가

```
// 댓글 템플릿 생성 함수
function addReply(reply) {
   return `
   <div class="media my-3"
      style="margin-left:${50*(reply.level-1)}px"
      data-reply-id="${reply.replyId}"
      data-board-id="${reply.boardId}"
      data-writer="${user}"
      data-parent="${reply.replyId}"
      data-level="${reply.level}">

      <img src="${context}/member/avata/${reply.writer}"
         class="rounded-circle avata-md">
```

❖ reply.js : 최상위 댓글 추가

```
      <div class="media-body ml-3">
         <div class="button-group float-right">
            작성일 :
            <span class="update-date">
               ${reply.updateDate.formatDate()}
            </span>
            <button class="reply-add-show">
               <i class="fa fa-reply"></i></button>
            <button class="reply-edit-show">
               <i class="fa fa-edit"></i></button>
            <button class="reply-delete-show">
               <i class="fa fa-trash"></i></button>
         </div>
         <h5>${reply.writer}</h5>
         <div class="reply-content">
            ${reply.content}
         </div>
         <div class="reply-work"></div>
      </div>
   </div>`;
}
```

❖ reply.js : 최상위 댓글 추가

```
$.fn.replyForm = function(listPanel) {
   var self = this;
   self.submit(function(e){
      e.preventDefault();                  // submit 기본 행동 막기
      var reply = formToObject(this); // 폼 데이터를 JS 객체로 변환

      replyAjax({
         url : $(this).attr("action"),
         type : 'post',
         data : JSON.stringify(reply),
         success : function(result){     // 처리 성공
            result.updateDate = new Date(result.updateDate);
            result.level = 1;                  // 최상위 댓글 수준
            listPanel.prepend(addReply(result)); // 댓글을 맨 앞에 추가
            self[0].reset();                   // 폼 내용 지우기
         }
      });
   });
   return self;
}
```

게시판
- 댓글 관리 UI -

❖ reply.js : 댓글 관리

- replyList 플러그인 처리
  - $(선택자).replyList(게시글 번호);

```
$('#reply-list').replyList(${board.boardId});
```

  - 추가/수정/삭제 버튼 이벤트 핸들러 등록
    - 해당 작업 인터페이스 추가
  - 작업 인터페이스의 완료/취소 버튼 이벤트 핸들러 등록

→ on() 메서드로 delegation 처리

❖ reply.js : 댓글 관리

```
$.fn.replyList = function(boardId) {
   var self = this; // 이벤트 핸들러에서 사용할 클로저 변수
   // ajax 처리 기본 url
   var baseurl = context + "/board/reply/" + boardId;

   // 댓글 추가, 수정, 삭제 버튼 이벤트 핸들러 설정
   // 각각의 작업 영역 출력
   self.on('click', '.reply-add-show', null);
   self.on('click', '.reply-edit-show', null);
   self.on('click', '.reply-delete-show', null);

   // 작업 영역의 취소 버튼 이벤트 핸들러 설정
   self.on('click', '.reply-cancel', null);

   // 작업 영역의 확인(추가, 수정, 삭제) 버튼 이벤트 핸들러 설정
   self.on('click', '.reply-post', null);
   self.on('click', '.reply-put', null);
   self.on('click', '.reply-delete', null);
}
```

댓글 시스템

❖ reply.js : 댓글 관리

❖ reply.js : 댓글 관리

```
$.fn.replyList = function(boardId) {
   var self = this; // 이벤트 핸들러에서 사용할 클로저 변수
   // ajax 처리 기본 url
   var baseurl = context + "/board/reply/" + boardId;

   // 댓글 추가, 수정, 삭제 버튼 이벤트 핸들러 설정
   // 각각의 작업 영역 출력
   self.on('click', '.reply-add-show', null);
   self.on('click', '.reply-edit-show', null);
   self.on('click', '.reply-delete-show', null);

   // 작업 영역의 취소 버튼 이벤트 핸들러 설정
   self.on('click', '.reply-cancel', null);

   // 작업 영역의 확인(추가, 수정, 삭제) 버튼 이벤트 핸들러 설정
   self.on('click', '.reply-post', null);
   self.on('click', '.reply-put', null);
   self.on('click', '.reply-delete', null);
}
```

❖ reply.js : 댓글 관리

```
// 댓글 추가/수정/삭제 작업 영역 템플릿 생성
function replyForm(method, content) {
   var template = `
      <div class="card card-body mp-3">`;

   if(method != 'delete') {  // 추가/수정 작업 시 textarea 추가
      template += `
         <textarea class="reply-edit-content" required
               style="width:100%">${content?content:''}</textarea>`;
   }

   template += `
         <div class="text-right mt-2">
            비밀번호 :
            <input type="password" name="password" required>
            <button type="button" class="reply-${method}">
                  확인</button>
            <button type="button" class="reply-cancel">취소</button>
         </div>
      </div>`;
   return template;
}
```

.reply-post
.reply-put
.reply-delete

❖ reply.js : 댓글 관리

```
// 댓글 작업 UI 생성 및 화면에 보이기
function setReplyWork(method, content) {
   // 댓글 추가 UI 생성
   var form = replyForm(method, content);

   // 댓글 수정 UI를 .reply-work에 추가 하기
   $(this).closest('.media-body') // 작업 영역 부모 찾기
         .find('.reply-work')// .reply-work 찾기
         .append(form);      // 댓글 수정 UI를 .reply-work에 추가
}

// 현재 댓글 내용 얻기 - 댓글 수정 시 사용
function getContent(target) {
   return $(target).closest('.media-body')
         .find('.reply-content')
         .hide()
         .text().trim();
}
```

❖ reply.js : 댓글 관리

```
$.fn.replyList = function(boardId) {
   var self = this; // 이벤트 핸들러에서 사용할 클로저 변수
   // ajax 처리 기본 url
   var baseurl = context + "/board/reply/" + boardId;

   // 댓글 추가, 수정, 삭제 버튼 이벤트 핸들러 설정
   // 각각의 작업 영역 출력
   self.on('click', '.reply-add-show',
         e=>setReplyWork.call(e.target, 'post'));
   self.on('click', '.reply-edit-show',
         e=>setReplyWork.call(e.target, 'put',
                        getContent(e.target)));
   self.on('click', '.reply-delete-show',
         e=>setReplyWork.call(e.target, 'delete'));
   :
}
```

게시판
- 댓글 관리 : 취소 -

❖ reply.js : 댓글 관리 - 취소 처리

```
$.fn.replyList = function(boardId) {
   :

   // 취소 처리
   self.on('click', '.reply-cancel', function(){
      // 취소 버튼이 속한 .media-body 찾기
      var body = $(this).closest('.media-body')
      body.find('.reply-work').empty();    // 작업 영역 제거
      body.find('.reply-content').show();  // 컨텐츠 영역 복원
   });

   :

}
```

게시판
– 댓글 관리 : 하위 댓글 처리 –

❖ reply.js 댓글 관리 : 공통 기능

```
// 댓글의 정보 추출
function getReply(target) {
   // 정보 추출을 위해 대상 .media 선택
   var media = $(target).closest('.media');

   return {
      replyId : media.data('reply-id'),  // replyId 추출(수정,삭제 시)
      boardId : media.data('board-id'),  // boardId 추출
      writer :  user,                    // user 전역변수로 작성자 추출
      parent :  media.data('parent'),    // 상위 댓글 번호 추출
      level :  media.data('level'),      // 댓글 수준 (추가 시)
      password : media.find(':password').val(),  // 비밀번호 추출
      content : media.find('textarea').val()     // 댓글 내용 추출
   }
}
```

❖ reply.js 댓글 관리 : 공통 기능

```
// 댓글의 유효성 검사
// 비밀번호와 내용 입력 여부 검사
// 정상인 경우 reply 객체 리턴
// 아닌 경우 undefiend 리턴
function checkReply(target, contentCheck) {

   var reply = getReply(target);   // reply 객체 추출

   if(reply.password =='') {
      return alert('비밀번호를 입력하세요');
   }

   if(contentCheck && reply.content =='') {
      return alert('내용을 입력하세요');
   }
   return reply;
}
```

❖ reply.js 댓글 관리 : 하위 댓글 추가

```
$.fn.replyList = function(boardId) {
   :
   // 댓글 추가
   self.on('click', '.reply-post', function () {
      var reply = checkReply(this, true);
      if(!reply) return;

      var media = $(this).closest('.media');
      var body = $(this).closest('.media-body');
      console.log(reply)
      replyAjax({
         url : baseurl,
         type : 'post',
         data : JSON.stringify(reply),
         success : function(result) {
            body.find('.reply-work').empty();  // 작업영역 제거
            result.updateDate = new Date(result.updateDate);
            result.level = reply.level + 1;    // 댓글 수준 증가
            media.after(addReply(result));
         }
      });
   });
```

❖ reply.js 댓글 관리 : 댓글 수정

```
$.fn.replyList = function(boardId) {
   :
   // 댓글 수정
   self.on('click', '.reply-put', function () {
      var reply = checkReply(this, true);
      if(!reply) return;

      var body = $(this).closest('.media-body');
      replyAjax({
         url : baseurl + "/" + reply.replyId,
         type : 'put',
         data : JSON.stringify(reply),
         success : function(result) {
            body.find('.reply-work').empty(); // 작업영역 제거
            result.updateDate = new Date(result.updateDate);
            body.find('.update-date')          // 수정일 갱신
                  .text(result.updateDate.formatDate());
            body.find('.reply-content').text(result.content);  // 내용
            body.find('.reply-content').show();
         }
      });
   });
```

❖ reply.js 댓글 관리 : 댓글 삭제

```
$.fn.replyList = function(boardId) {
   :
   // 댓글 삭제
   self.on('click', '.reply-delete', function replyDelete() {
      var reply = checkReply(this);
      if(!reply) return;
      if(!confirm('해당 댓글을 삭제할까요?')) return;

      var body = $(this).closest('.media-body');
      var url = baseurl + "/" + reply.replyId
                  + "?password=" + reply.password;
      replyAjax({
         url : url,
         type : 'delete',
         success : function(result) {
            body.find('.reply-content').text('삭제된 글입니다.');
            body.find('.reply-work').empty();
            body.find('.button-group').empty();
            body.find('.reply-content').show();
         }
      });
   });
```