코드로 배우는 스프링 웹 프로젝트

- PART 4

02 특정 게시물의 댓글 목록

```
@GetMapping(value = "/pages/{bno}/{page}",
      produces = {
          MediaType.APPLICATION_XML_VALUE,
          MediaType.APPLICATION_JSON_UTF8_VALUE })
  public ResponseEntity<List<ReplyVO>> getList(
      @PathVariable("page") int page,
      @PathVariable("bno") Long bno) {

    log.info("getList.................");
    Criteria cri = new Criteria(page,10);
    log.info(cri);

  return new ResponseEntity<>(service.getList(cri, bno), HttpStatus.OK);
  }
```

XML과 JSON타입으로 서비스

localhost:8080/replies/pages/3145745/1

This XML file does not appear to have any style information associated

```
<ReplyPageDTO>
  <replyCnt>8</replyCnt>
  <list>
    <list>
      <rno>5</rno>
      <bno>3145745</bno>
      <reply>댓글 테스트 5</reply>
      <replyer>replyer5</replyer>
      <replyDate>1534550815000</replyDate>
      <updateDate>1534550815000</updateDate>
    </list>
    <list>
      <rno>10</rno>
      <bno>3145745</bno>
      <reply>댓글 테스트 10</reply>
      <replyer>replyer10</replyer>
      <replyDate>1534550815000</replyDate>
      <updateDate>1534550815000</updateDate>
    </list>
    <list>
      <rno>11</rno>
      <bno>3145745</bno>
      <reply>Hello Reply</reply>
      <replyer>user00</replyer>
      <replyDate>1534552080000</replyDate>
      <updateDate>1534552080000</updateDate>
    </list>
    <list>
      <rno>12</rno>
      <bno>3145745</bno>
      <reply>Hello Reply</reply>
      <replyer>user00</replyer>
      <replyDate>1534552193000</replyDate>
      <updateDate>1534552193000</updateDate>
    </list>
```

02 댓글의 삭제/조회

```
@GetMapping(value = "/{rno}",
     produces = { MediaType.APPLICATION_XML_VALUE, MediaType.APPLICATION_JSON_UTF8_VALUE })
 public ResponseEntity<ReplyVO> get(@PathVariable("rno") Long rno) {


   log.info("get: " + rno);


   return new ResponseEntity<>(service.get(rno), HttpStatus.OK);
 }


 @DeleteMapping(value= "/{rno}" ,produces = { MediaType.TEXT_PLAIN_VALUE })
 public ResponseEntity<String> remove(@PathVariable("rno") Long rno) {


   log.info("remove: " + rno);


   return service.remove(rno) == 1 ? new ResponseEntity<>("success", HttpStatus.OK)
       : new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);


 }
```

02 댓글의 수정

```
@RequestMapping(method = { RequestMethod.PUT, RequestMethod.PATCH },
      value = "/{rno}",
      consumes = "application/json",
      produces = { MediaType.TEXT_PLAIN_VALUE })
 public ResponseEntity<String> modify(@RequestBody ReplyVO vo, @PathVariable("rno") Long rno) {


   vo.setRno(rno);


   log.info("rno: " + rno);
   log.info("modify: " + vo);


   return service.modify(vo) == 1
       : new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);


 }
```

- JS의 모듈 패턴
  - 여러 기능들을 모아서 하나의 모듈화
  - 클로저를 이용해서 상태 유지
  - 여러 함수들이 메서드화 되므로 객체지향 구조에 적합

reply.js

```
console.log("Reply Module.....");

var replyService = (function(){

  function add(reply, callback){
    console.log("reply.....");
  }

  return {add:add};
})();
```

02 reply.js 댓글 등록

```
var replyService = (function() {

  function add(reply, callback, error) {
    console.log("add reply..............");
    $.ajax({
      type : 'post',
      url : '/replies/new',
      data : JSON.stringify(reply),
      contentType : "application/json; charset=utf-8",
      success : function(result, status, xhr) {
        if (callback) { callback(result); }
      },
      error : function(xhr, status, er) {
        if (error) { error(er); }
      }
    })
  }
   return {
    add : add
  };
})();
```

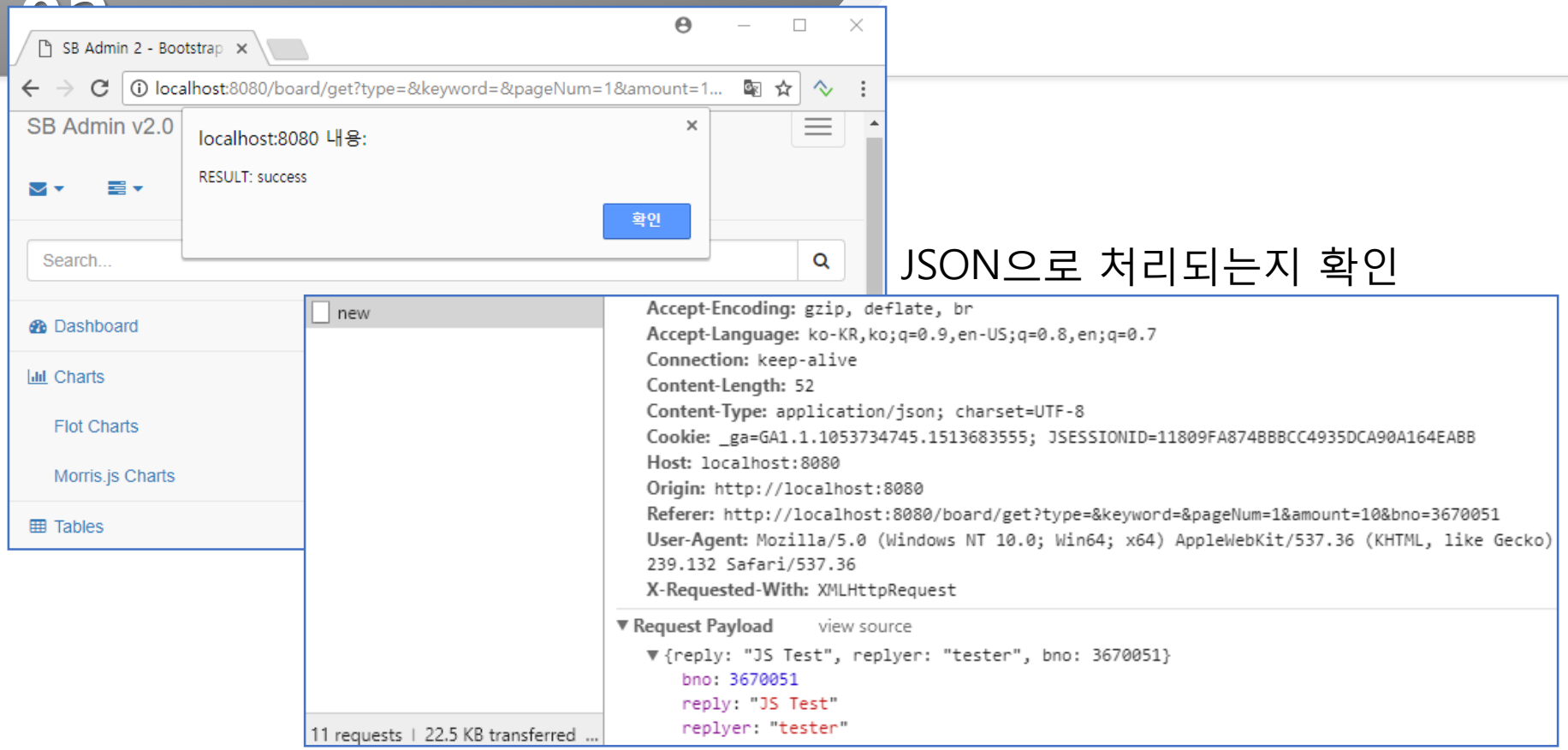
Ajax처리후 동작해야 하는 함수

Ajax로 ReplyController호출

모듈 패턴으로 외부에 노출하는 정보

02 조회 화면에서 호출

```
script type="text/javascript"  src="/resources/js/reply.js"></script>


<script>


console.log("===============");

console.log("JS TEST");


var bnoValue = '<c:out value="${board.bno}"/>';


//for replyService add test

replyService.add(

    {reply:"JS Test", replyer:"tester", bno:bnoValue} ,

    function(result){

      alert("RESULT: " + result);

    }

);


</script>
```

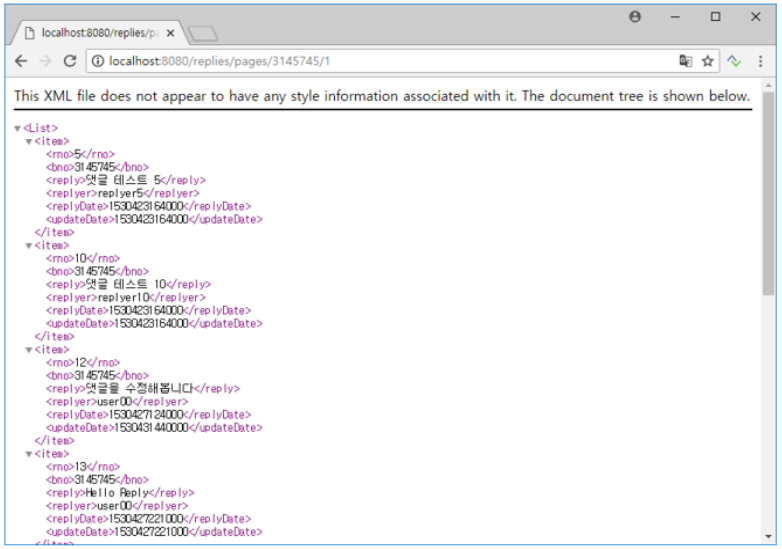


JSON으로 처리되는지 확인

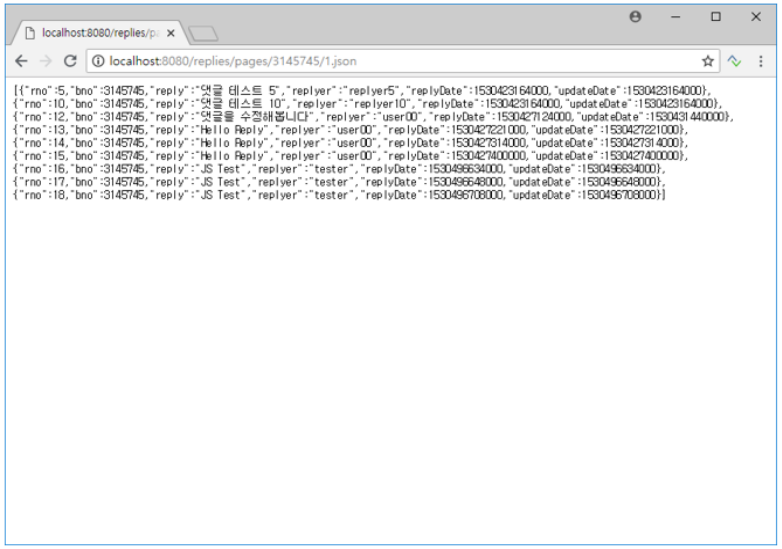
02 댓글의 목록 처리

댓글의 처리 전에 우선적으로 확인한 후에 진행

http://localhost:8080/replies/pages/3145745/1



http://localhost:8080/replies/pages/3145745/1.json



02 getJSON( )처리 –reply.js

```
function getList(param, callback, error) {


    var bno = param.bno;
    var page = param.page || 1;


    $.getJSON("/replies/pages/" + bno + "/" + page + ".json",
        function(data) {
          if (callback) {
            callback(data);
          }
        }).fail(function(xhr, status, err) {
      if (error) {
        error();
      }
    });
  }
```

Ajax로 ReplyController호출

02 화면에서의 호출 테스트

```
replyService.getList({bno:bnoValue, page:1}, function(list){

    for(var i = 0,  len = list.length||0; i < len; i++ ){
      console.log(list[i]);
    }
});
```

02 댓글의 삭제와 갱신

reply.js의 일부

```
function remove(rno, callback, error) {
   $.ajax({
     type : 'delete',
     url : '/replies/' + rno,
     success : function(deleteResult, status, xhr) {
       if (callback) {
         callback(deleteResult);
       }
     },
     error : function(xhr, status, er) {
       if (error) {
         error(er);
       }
     }
   });
 }
```

전송방식은 DELETE 방식 사용

02 댓글 삭제 테스트

- 존재하는 댓글의 번호를 이용해서 처리

```
//23번 댓글 삭제 테스트
replyService.remove(23, function(count) {

  console.log(count);

  if (count === "success") {
    alert("REMOVED");
  }
}, function(err) {
  alert('ERROR...');
});
```

삭제전

RN	RNO	REPLY	REPLYER	REPLYDATE	UPDATEDATE
1	5	댓글 테스트 5	replyer5	18/04/14	18/04/14
2	10	Update Reply	replyer10	18/04/14	18/04/14
3	21	JS Test	tester	18/04/14	18/04/14
4	22	JS Test	tester	18/04/14	18/04/14
5	23	JS Test	tester	18/04/14	18/04/14

삭제후

RN	RNO	REPLY	REPLYER	REPLYDATE	UPDATEDATE
1	5	댓글 테스트 5	replyer5	18/04/14	18/04/14
2	10	Update Reply	replyer10	18/04/14	18/04/14
3	21	JS Test	tester	18/04/14	18/04/14
4	22	JS Test	tester	18/04/14	18/04/14

02 댓글 수정/테스트

```
function update(reply, callback, error) {
     $.ajax({
      type : 'put',
      url : '/replies/' + reply.rno,
      data : JSON.stringify(reply),
      contentType : "application/json; charset=utf-8",
      success : function(result, status, xhr) {
        if (callback) {
          callback(result);
        }
      },
      error : function(xhr, status, er) {
        if (error) {
          error(er);
        }
      }
    });
  }
```

PUT방식으로 호출
전달하는 데이터는 JSON데이터

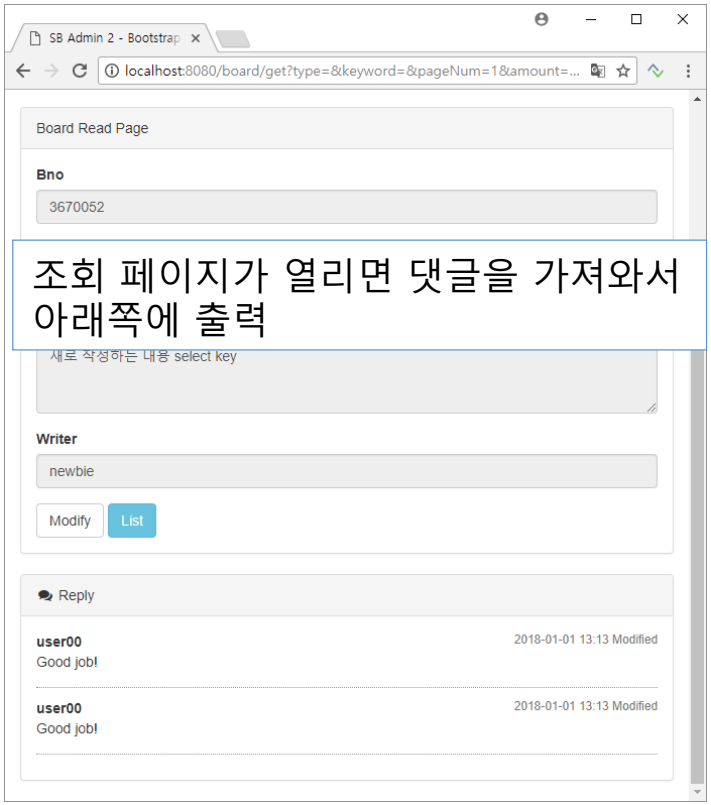
```
//22번 댓글 수정
replyService.update({
  rno : 22,
  bno : bnoValue,
  reply : "Modified Reply...."
}, function(result) {

  alert("수정 완료...");

});
```

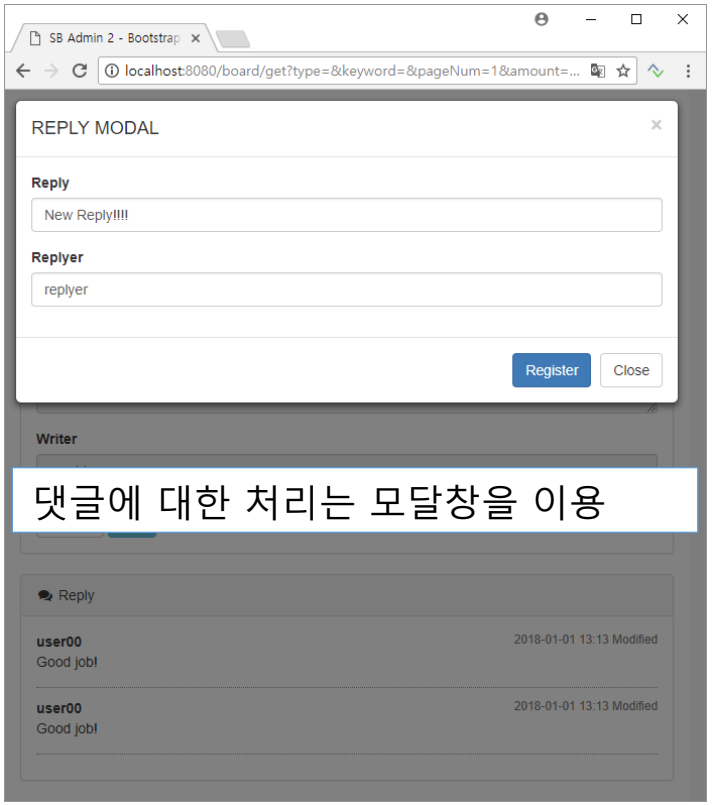
02 특정댓글조회/테스트

```
function get(rno, callback, error) {

    $.get("/replies/" + rno + ".json", function(result) {

      if (callback) {
        callback(result);
      }

    }).fail(function(xhr, status, err) {
      if (error) {
        error();
      }
    });
  }
```

```
replyService.get(10, function(data){
   console.log(data);
 });
```

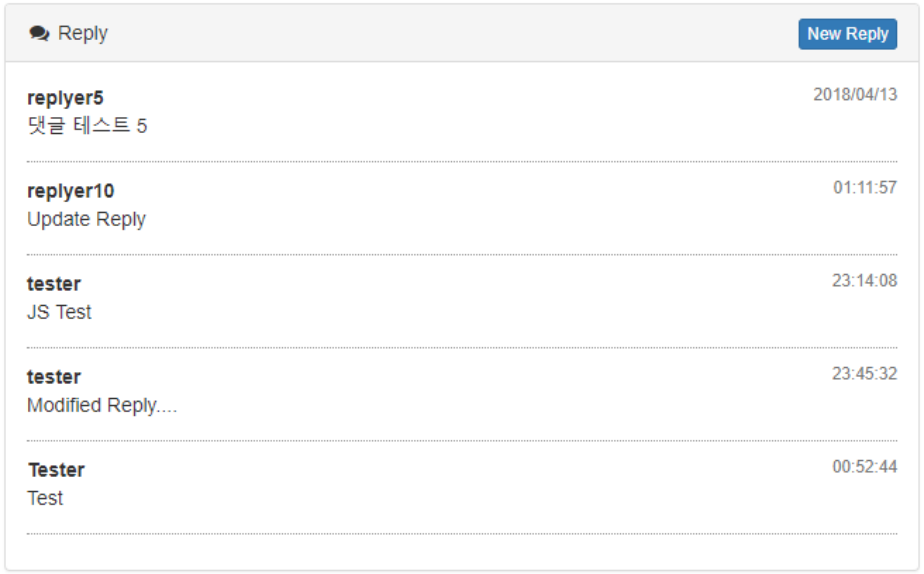
- Ajax호출 이벤트 처리와 후 처리





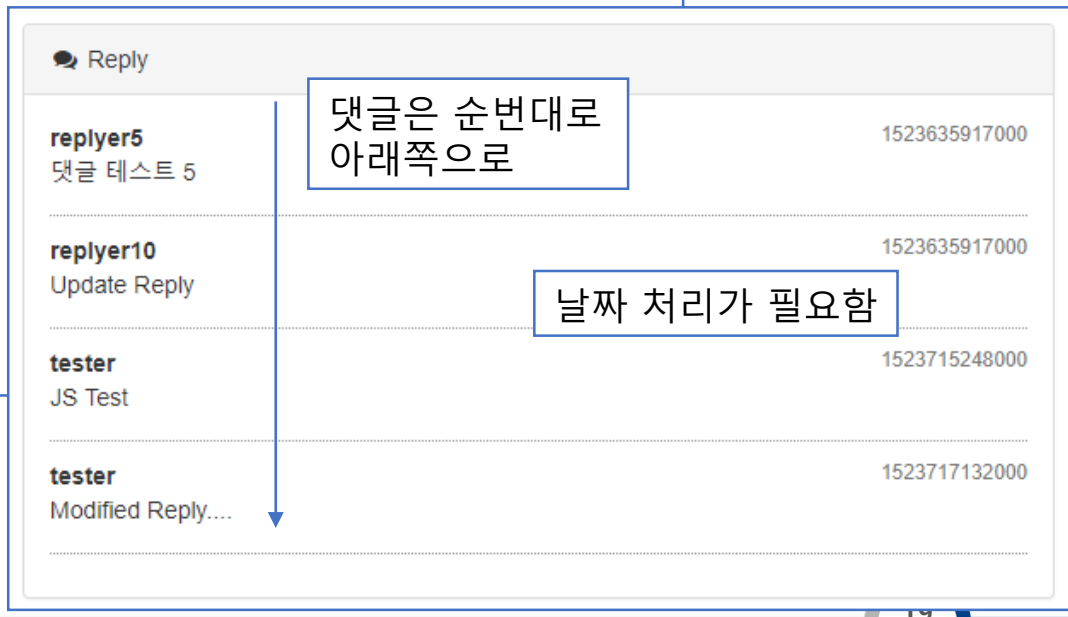
02 댓글 목록의 처리

- 댓글의 목록은 <ul> 태그 내에 <li> 태그를 이용해서 처리
- 태그는 하나의 댓글을 의미
- 수정이나 삭제 시에는 반드시 댓글의 번호(rno)가 필요하므로 'data-rno' 속성을 이용



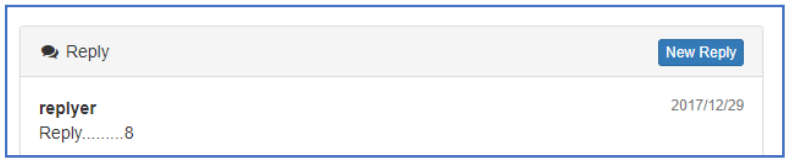
```
function showList(page){

    replyService.getList({bno:bnoValue,page: page|| 1 }, function(list) {

      var str="";
     if(list == null || list.length == 0){

      replyUL.html("");

      return;
    }
      for (var i = 0, len = list.length || 0; i < len; i++) {
        str +="<li class='left clearfix' data-rno='"+list[i].rno+"'>";
        str +="  <div><div class='header'><strong class='primary-font'>"+list[i].replyer+"</strong>";
        str +="    <small class='pull-right text-muted'>"+list[i].replyDate+"</small></div>";
        str +="    <p>"+list[i].reply+"</p></div></li>";
      }

   replyUL.html(str);

    });//end function

}//end showList
```

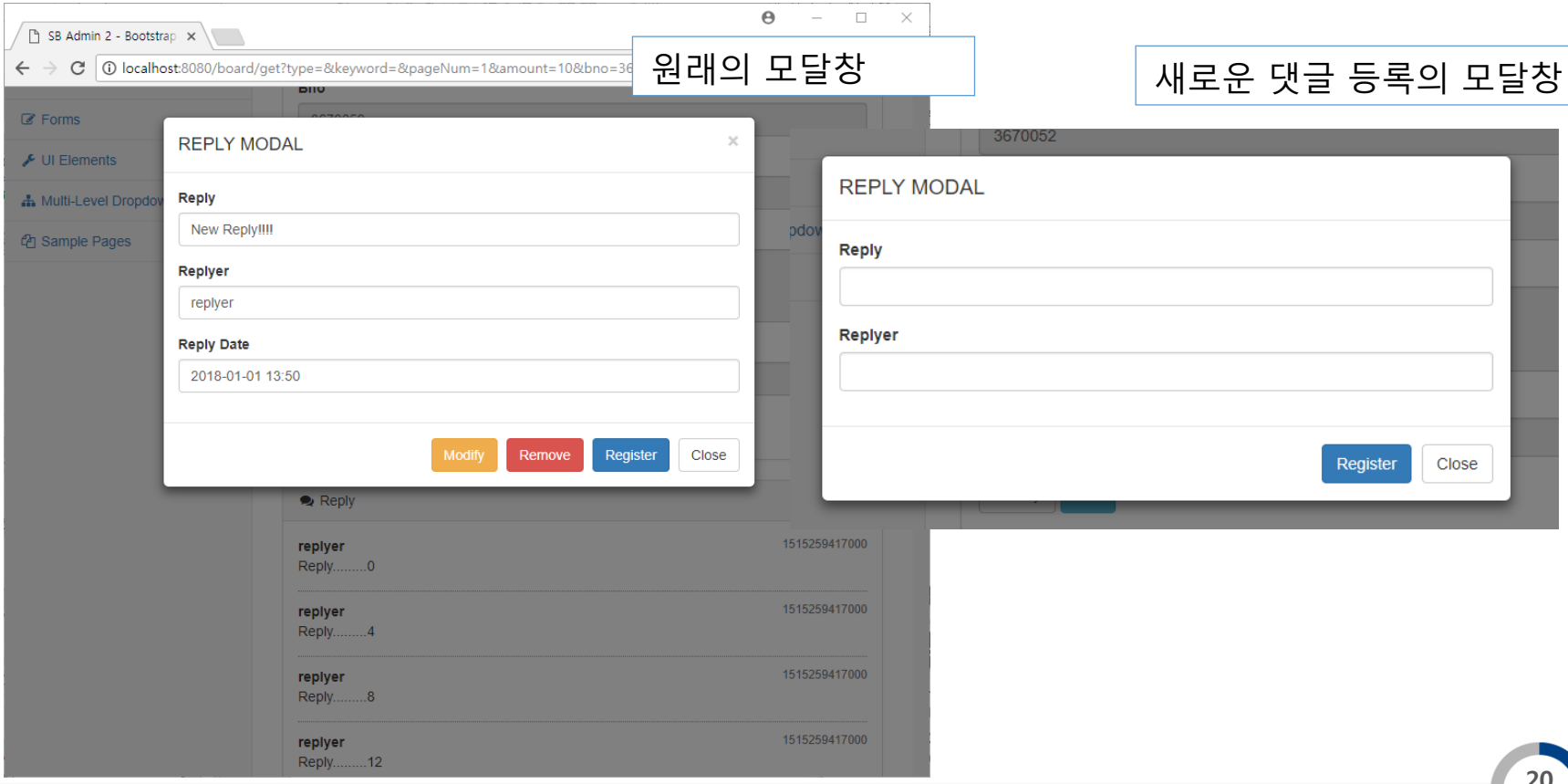
showList(페이지번호)는 해당 게시글의 댓글을 가져온 후 태그를 만들어서 화면에 보여준다.



02 새로운 댓글의 처리

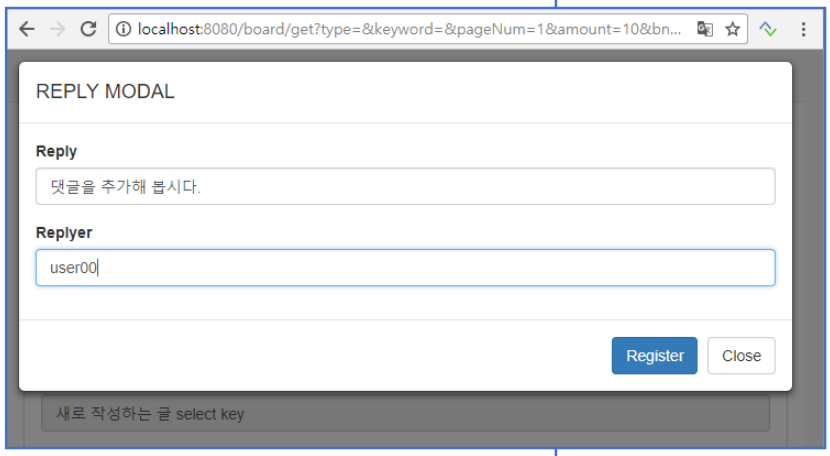
- 모달창을 이용해서 댓글 추가

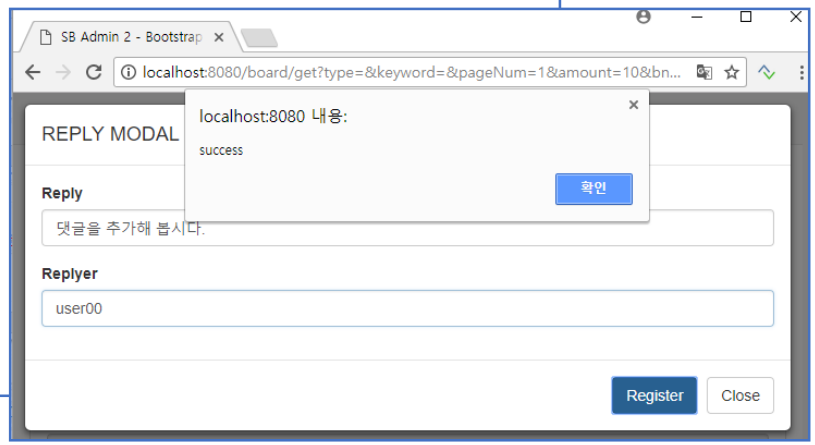




02 댓글 추가후 처리

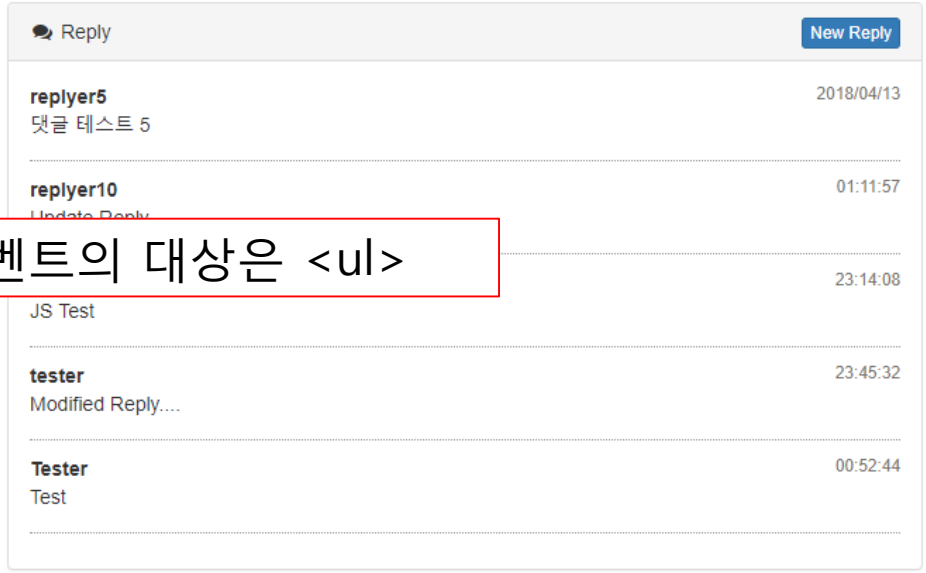
```
modalRegisterBtn.on("click",function(e){

      var reply = {
            reply: modalInputReply.val(),
            replyer:modalInputReplyer.val(),
            bno:bnoValue
          };
      replyService.add(reply, function(result){

        alert(result);

        modal.find("input").val("");
        modal.modal("hide");

      });

  });
```





02 특정 댓글의 클릭 이벤트

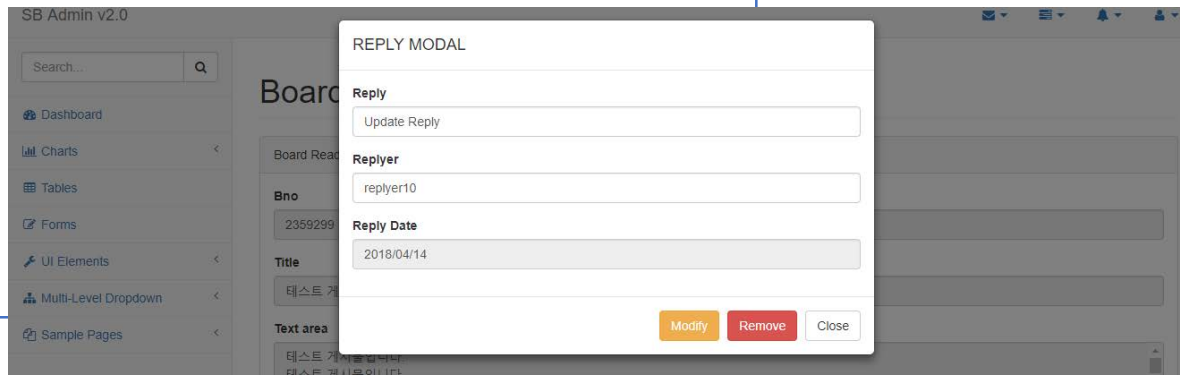
- 댓글은 Ajax로 가져온 결과를 DOM에 추가하는 형태이므로 이벤트 위임(delegation) 방식을 이용해서 처리



이벤트의 대상은

이벤트 위임으로 가 이벤트의 주인공

```
//댓글 조회 클릭 이벤트 처리
    $(".chat").on("click", "li", function(e){

      var rno = $(this).data("rno");

      replyService.get(rno, function(reply){

        modalInputReply.val(reply.reply);
        modalInputReplyer.val(reply.replyer);
        modalInputReplyDate.val(replyService.displayTime( reply.replyDate))
        .attr("readonly","readonly");
        modal.data("rno", reply.rno);

        modal.find("button[id !='modalCloseBtn']").hide();
        modalModBtn.show();
        modalRemoveBtn.show();

        $(".modal").modal("show");

      });
  });
```



02 댓글의 수정/삭제 처리 이벤트

- 수정/삭제 처리는 Ajax로 하고, 모달창 close
- 수정/삭제 후에는 다시 최신 댓글 목록 갱신

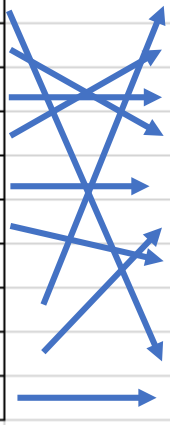
```
modalModBtn.on("click", function(e){

      var reply = {rno:modal.data("rno"), reply: modalInputReply.val()};

      replyService.update(reply, function(result){

        alert(result);
        modal.modal("hide");
        showList(1);

      });

  });
```

02 댓글의 페이징처리

- 데이터베이스의 인덱스 설계
  - 모든 댓글의 조회는 게시물 번호를 이용해서 처리하므로, 게시물 번호의 정렬된 구조가 필요

IDX_REPLY

bno	rno	ROWID
300	200	AAAXXK
200	3	AAAXXB
200	13	AAAXXD
200	22	AAAXXE
200	43	AAAXXF
200	89	AAAXXG
200	100	AAAXXH
100	1	AAAXXA
100	5	AAAXXC
100	123	AAAXXC

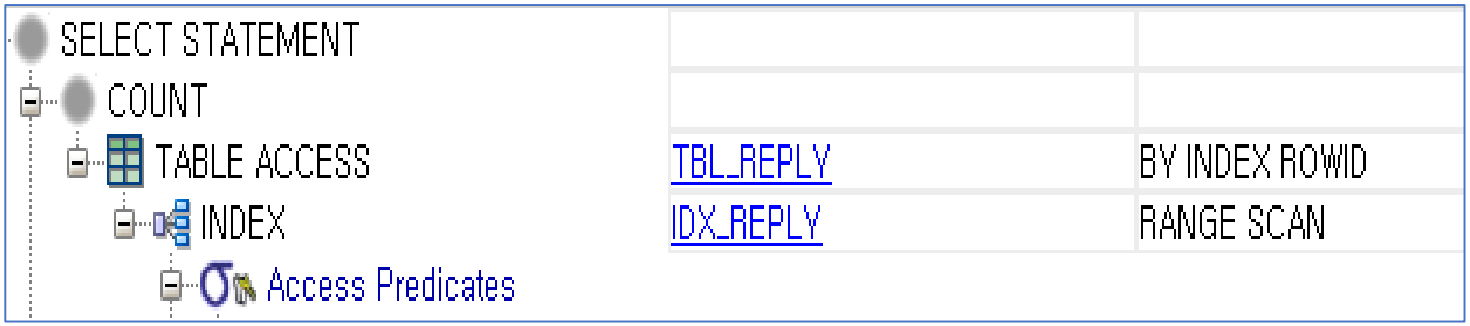


TBL_REPLY

ROWID	rno	bno	reply
AAAXXA	1	100	xxx
AAAXXE	22	200	xxx
AAAXXD	13	200	xxx
AAAXXB	3	200	xxx
AAAXXF	43	200	xxx
AAAXXC	5	100	xxx
AAAXXG	89	200	xxx
AAAXXH	100	200	xxx
AAAXXK	200	300	xxx
AAAXXI	123	100	xxx

02 인덱스를 이용하는 댓글 페이징 처리

```
select /*+INDEX(tbl_reply idx_reply) */
    rownum rn, bno, rno, reply, replyer, replyDate, updatedate
    from tbl_reply
    where bno = 3145745(게시물 번호)
    and rno > 0;
```



02 댓글의 수(카운트)

- 댓글 페이징을 위한 숫자 파악

```
<select id="getCountByBno" resultType="int">
<![CDATA[
select count(rno) from tbl_reply where bno = #{bno}
]]>
</select>
```

02 ReplyServiceImpl처리

- 댓글의 숫자와 댓글 목록을 처리하는 ReplyPageDTO 클래스

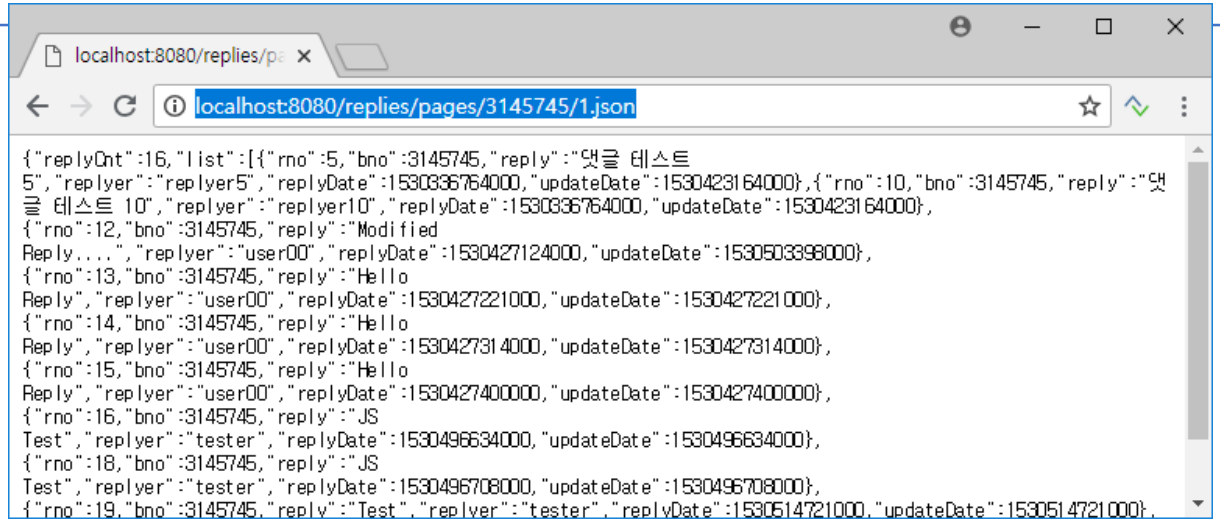
```
@Data
@AllArgsConstructor
@Getter
public class ReplyPageDTO {


  private int replyCnt;
  private List<ReplyVO> list;
}
```

```java
public interface ReplyService {

...생략...

    public ReplyPageDTO getListPage(Criteria cri, Long bno);
}
```

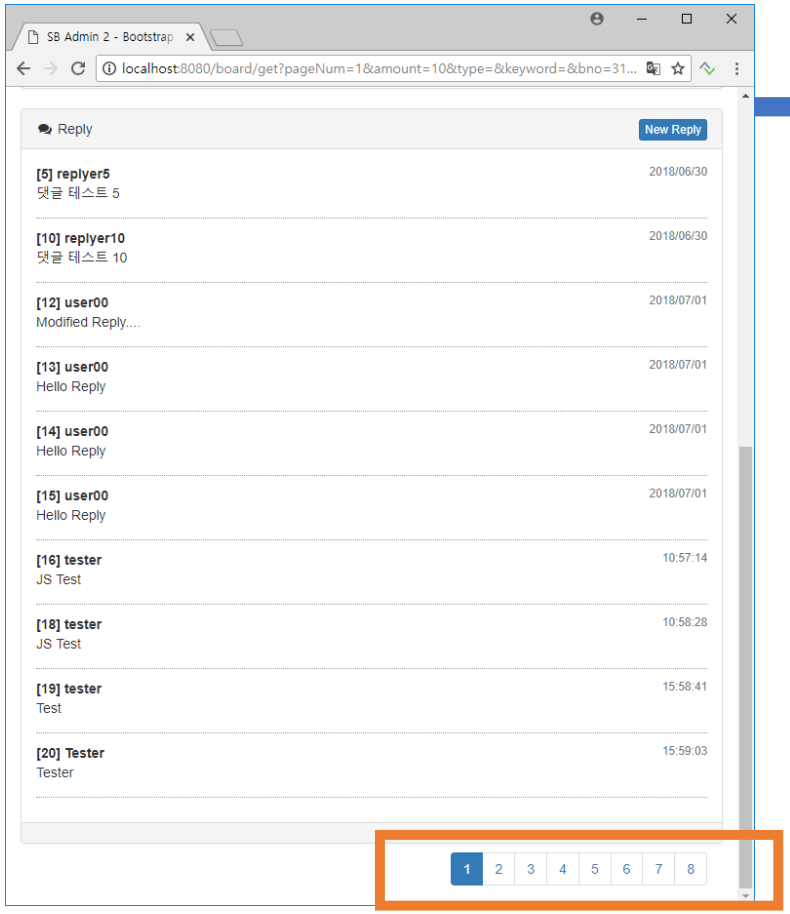
```java
public class ReplyServiceImpl implements ReplyService {

  @Setter(onMethod_ = @Autowired)
  private ReplyMapper mapper;

  ...생략...

  @Override
  public ReplyPageDTO getListPage(Criteria cri, Long bno)
{

    return new ReplyPageDTO(
        mapper.getCountByBno(bno),
        mapper.getListWithPaging(cri, bno));
  }

}
```

02 ReplyController의 수정

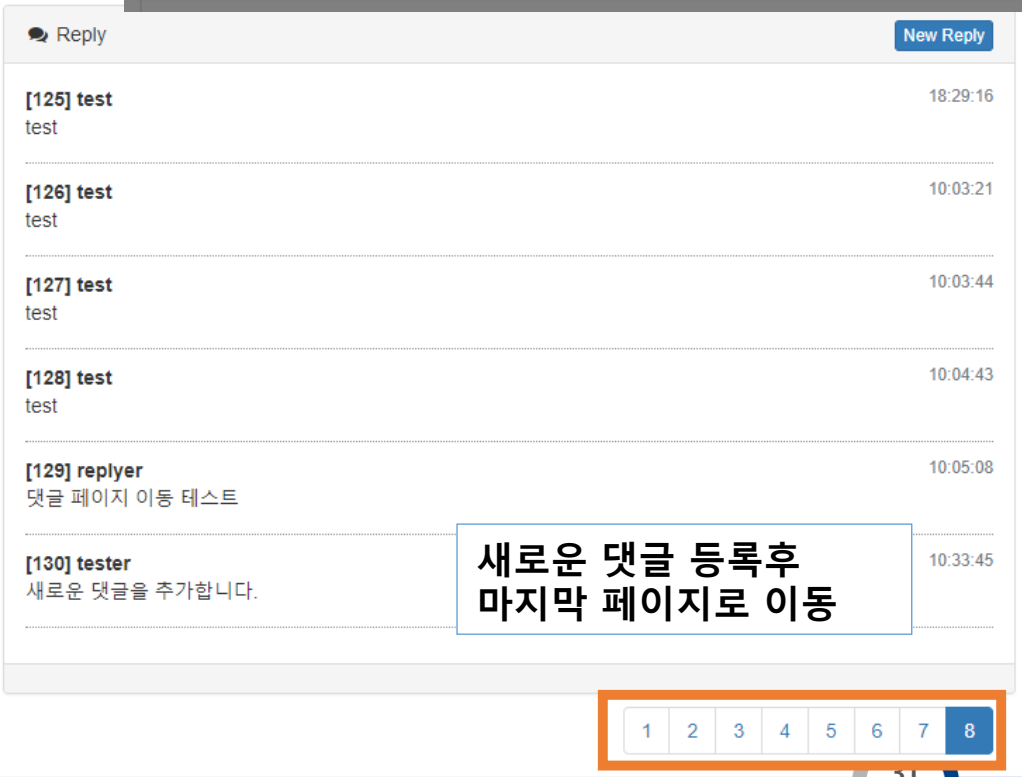
```
@GetMapping(value = "/pages/{bno}/{page}",
      produces = { MediaType.APPLICATION_XML_VALUE,
      MediaType.APPLICATION_JSON_UTF8_VALUE })
 public ResponseEntity<ReplyPageDTO> getList(@PathVariable("page") int page, @PathVariable("bno") Long bno) {


   Criteria cri = new Criteria(page, 10);


   log.info("get Reply List bno: " + bno);


   log.info("cri:" + cri);


   return new ResponseEntity<>(service.getListPage(cri, bno), HttpStatus.OK);
 }
```



02 댓글의 화면 처리


SB Admin 2 - Bootstrap
localhost:8080/board/get?pageNum=1&amount=10&type=&keyword=&bno=31...
Reply
New Reply
[5] replyer5
댓글 테스트 5
2018/06/30
[10] replyer10
댓글 테스트 10
2018/06/30
[12] user00
Modified Reply....
2018/07/01
[13] user00
Hello Reply
2018/07/01
[14] user00
Hello Reply
2018/07/01
[15] user00
Hello Reply
2018/07/01
[16] tester
JS Test
10:57:14
[18] tester
JS Test
10:58:28
[19] tester
Test
15:58:41
[20] Tester
Tester
15:59:03
1 2 3 4 5 6 7 8


REPLY MODAL
Reply
새로운 댓글을 추가합니다.
Replyer
tester
Register


Reply
New Reply
[125] test
test
18:29:16
[126] test
test
10:03:21
[127] test
test
10:03:44
[128] test
test
10:04:43
[129] replyer
댓글 페이지 이동 테스트
10:05:08
[130] tester
새로운 댓글을 추가합니다.
10:33:45
새로운 댓글 등록후
마지막 페이지로 이동
1 2 3 4 5 6 7 8

02 댓글의 페이지 계산과 출력

- reply.js 는 게시물의 번호와 페이지를 사용하도록 수정

```
function getList(param, callback, error) {


   var bno = param.bno;
   var page = param.page || 1;


   $.getJSON("/replies/pages/" + bno + "/" + page + ".json",
       function(data) {
         if (callback) {
           //callback(data); // 댓글 목록만 가져오는 경우
           callback(data.replyCnt, data.list); //댓글 숫자와 목록을 가져오는 경우
         }
       }).fail(function(xhr, status, err) {
     if (error) {
       error();
     }
   });
 }
```

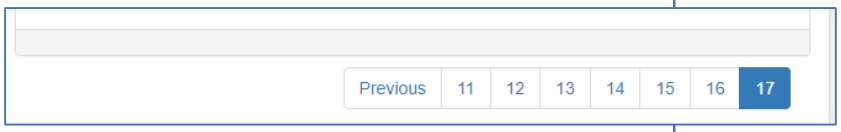
02 새로운 댓글 추가

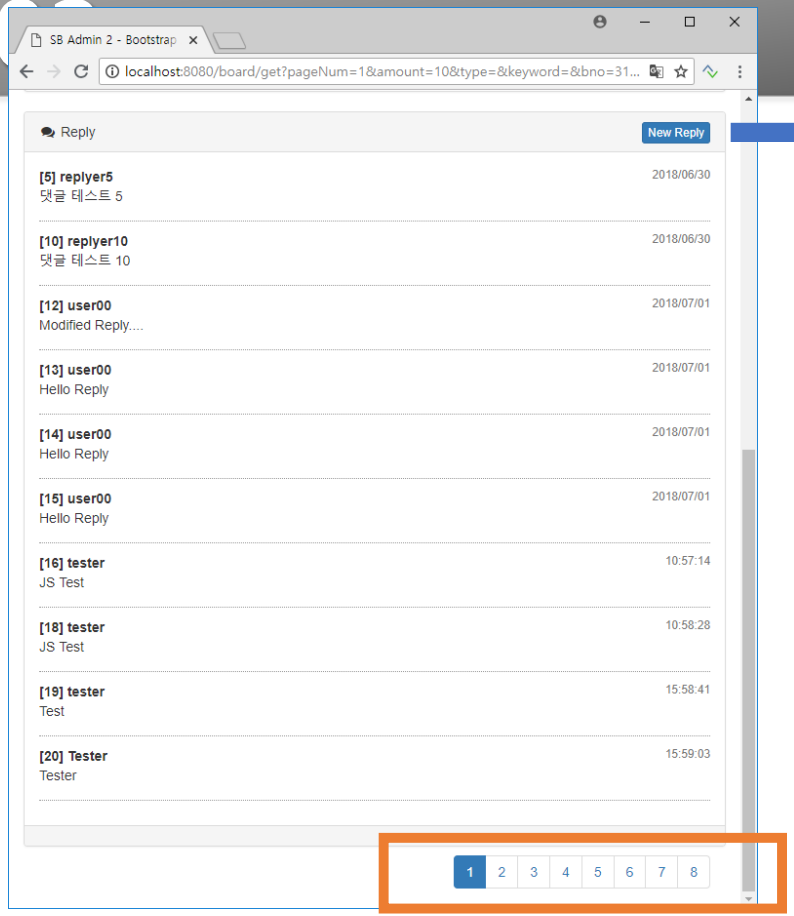
- 새로운 댓글을 추가하면 page값을 -1로 전송하고, 댓글의 전체 숫자를 파악한 후에 페이지 이동

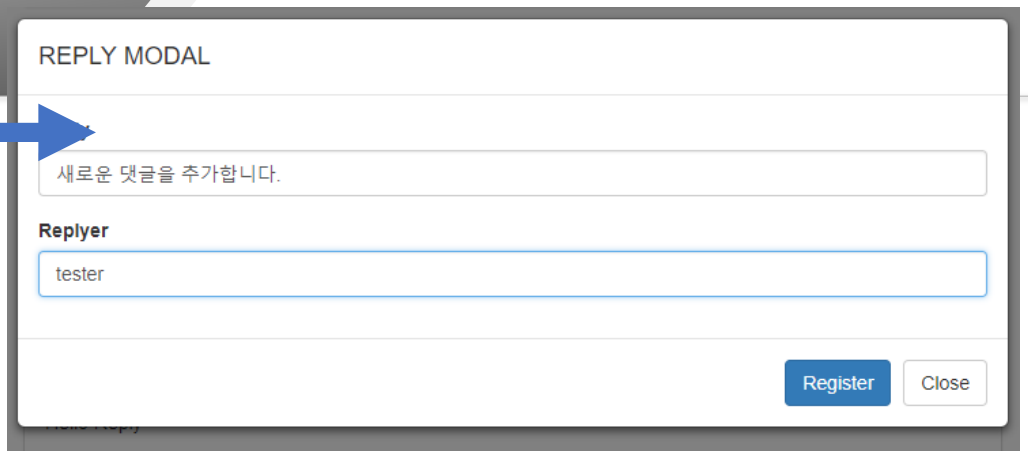
```
modalRegisterBtn.on("click",function(e){

      var reply = {
            reply: modalInputReply.val(),
            replyer:modalInputReplyer.val(),
            bno:bnoValue
          };
      replyService.add(reply, function(result){
        alert(result);
        modal.find("input").val("");
        modal.modal("hide");
        //showList(1);
        showList(-1);
     });
 });
```

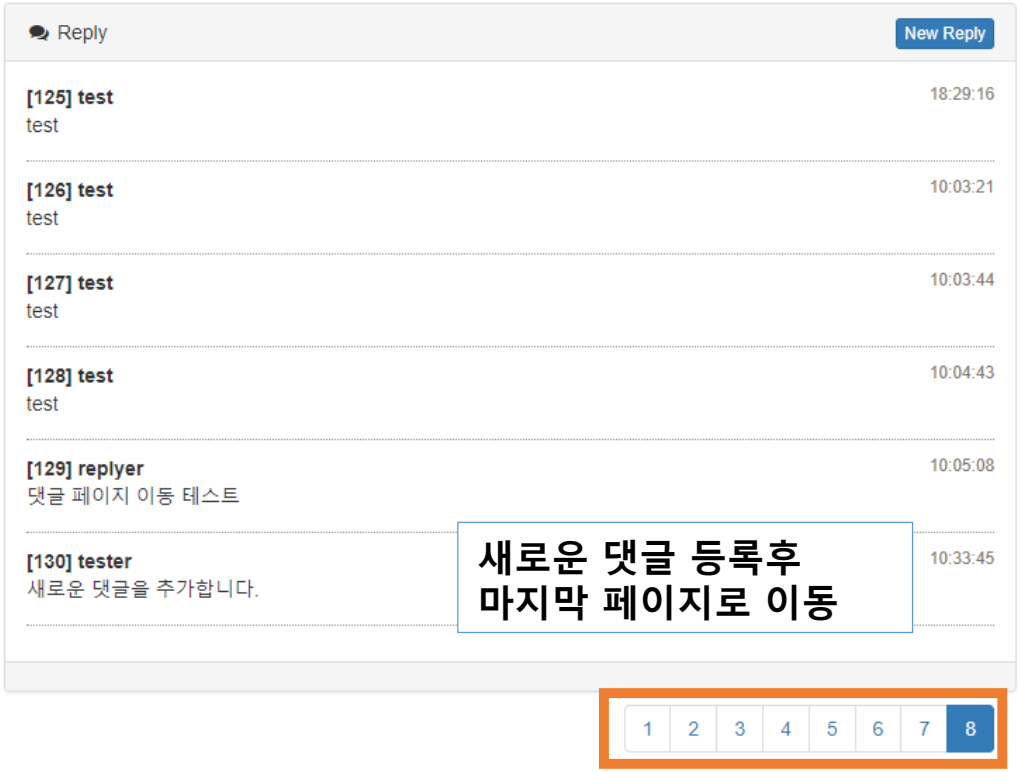
02 댓글의 페이지 번호 처리

```
<!-- /.panel-heading  기존에 존재하는 부분 -->
<div class="panel-body">

      <ul class="chat">
      </ul>

</div>
<!-- /.panel .chat-panel 추가-->

<div class="panel-footer">

</div>
```









새로운 댓글 등록후
마지막 페이지로 이동

Q & A

열공합시다!!