

16장. 댓글 구현



reply



댓글 구현

제목



Search

번호	제목	작성자	등록일	조회수
24	토성 [1]	cloud	2022-09-19 04:48:04	3
23	목성 [1]	cloud	2022-09-19 04:47:56	3
22	화성 [0]	cloud	2022-09-19 04:47:49	0
21	지구 [2]	cloud	2022-09-19 04:47:39	4
20	금성 [0]	cloud	2022-09-19 04:47:31	0
19	수성 [0]	cloud	2022-09-19 04:47:24	1
18	보라 [1]	sky123	2022-09-19 04:46:49	2
17	남색 [0]	sky123	2022-09-19 04:46:42	0
16	파랑 [0]	sky123	2022-09-19 04:46:31	0
15	초록 [0]	sky123	2022-09-19 04:46:20	0

1 2 3

글쓰기



댓글 구현

등록일	2022-09-19 04:42:18
조회수	<input type="text" value="5"/>
<input type="button" value="목록"/>	

댓글

작성자: admin (작성일: 2022-09-19 04:50:14)
여름이 가고... 가을이 깊어갑니다.

작성자: sky123 (작성일: 2022-09-19 04:51:45)
가을옷을 준비해야겠어요

작성자



댓글 구현

- DB 설계 - 댓글 테이블 만들기

```
-- 댓글 테이블
CREATE TABLE tbl_reply(
    rno number(5),
    bno number(5) not null,
    reply VARCHAR2(1000) NOT NULL,
    replyer VARCHAR2(50) NOT NULL,
    replydate DATE DEFAULT SYSDATE,
    update date DATE DEFAULT SYSDATE
);
-- 자동 순번
CREATE SEQUENCE seq_reply;
-- 기본키 설정
ALTER TABLE tbl_reply ADD CONSTRAINT pk_reply PRIMARY KEY(rno);
-- 외래키 설정
ALTER TABLE tbl_reply ADD CONSTRAINT fk_reply_board
FOREIGN key(bno) REFERENCES tbl_board(bno);
```

댓글은 게시물이 있어야
작성할수 있으므로
bno로 관계를 맺음



댓글 구현

- DB 설계 - 댓글 테이블 만들기

```
-- 더미 데이터 (댓글) 입력
INSERT INTO tbl_reply(rno, bno, reply, replyer)
VALUES (seq_reply.NEXTVAL, 2, '태풍이 자주 오네요..', 'admin');
```

```
SELECT * FROM tbl_reply
WHERE bno = 2;
```

RNO	BNO	REPLY	REPLYER	REPLYDATE	UPDATEDATE
4	2	여름 가고...	sky123	22/09/19	22/09/19



댓글 구현

- ReplyVO 클래스

```
@Data
public class ReplyVO {

    private int rno;    //댓글 번호
    private int bno;    //글 번호

    private String reply;    //댓글 내용
    private String replyer;  //댓글 작성자
    private Date replyDate;  //댓글 작성일
    private Date updateDate; //댓글 수정일
}
```



댓글 구현

- ReplyMapper 설계

```
public interface ReplyMapper {  
  
    public int register(ReplyVO vo); //댓글 추가  
  
    public List<ReplyVO> getReplyList(int bno); //댓글 목록 조회  
  
}
```

```
<mapper namespace="com.cloud.mapper.ReplyMapper">  
    <insert id="register">  
        INSERT INTO tbl_reply (rno, bno, reply, replyer)  
        VALUES (seq_reply.nextval, #{bno}, #{reply}, #{replyer})  
    </insert>  
  
    <select id="getReplyList" resultType="com.cloud.domain.ReplyVO">  
        SELECT * FROM tbl_reply WHERE bno = #{bno}  
    </select>  
</mapper>
```



댓글 구현

- Service 설계

```
public interface ReplyService {  
    public int register(ReplyVO vo); //댓글 추가  
    public List<ReplyVO> getReplyList(int bno); //댓글 목록 조회  
}
```



댓글 구현

- Service 설계

```
@Service
public class ReplyServiceImpl implements ReplyService{

    @Autowired
    private ReplyMapper mapper;

    //댓글 등록
    @Override
    public int register(ReplyVO vo) {
        return mapper.register(vo);
    }

    //댓글 목록
    @Override
    public List<ReplyVO> getReplyList(int bno) {
        return mapper.getReplyList(bno);
    }
}
```



댓글 구현

- Controller 설계 – BoardController.java

```
@Log4j
@RequestMapping("/board/*")
@Controller
public class BoardController {

    @Autowired
    private BoardService service;

    @Autowired
    private ReplyService replyService;
```

```
//게시글 상세 보기, 댓글 목록
@PreAuthorize("isAuthenticated()")
@RequestMapping("/boardView")
public String getBoard(int bno, @ModelAttribute("cri") Criteria cri, Model model) {
    service.updateCount(bno); //조회수 증가
    BoardVO board = service.getBoard(bno); //상세 보기 처리
    List<ReplyVO> replyList = replyService.getReplyList(bno);

    model.addAttribute("board", board); //model-"board"
    model.addAttribute("replyList", replyList);
    return "/board/boardView";
}
```



- Controller 설계 – BoardController.java

```
//댓글 등록
@PostMapping("/reply")
public String reply(ReplyVO vo, RedirectAttributes rttr) {
    Log.info("댓글 작성");
    replyService.register(vo);

    rttr.addAttribute("bno", vo.getBno());

    return "redirect:/board/boardView";
}
```



댓글 구현

- view 설계 – boardView.jsp

```
</table>
</form>
<!-- 댓글 영역 -->
<div class="comment">
    <h4>댓글</h4>
    <ol class="replyList">
        <c:forEach items="${replyList}" var="list">
            <li>
                <p>작성자: <c:out value="${list.replyer}" />&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
                    (작성일: <fmt:formatDate value="${list.replyDate}"
                        pattern="yyyy-MM-dd hh:mm:ss"/>)
                </p>
                <p><c:out value="${list.reply}" /></p>
            </li>
        </c:forEach>
    </ol>
<!-- 댓글 등록 폼 -->
```

댓글 구현

- view 설계 – boardView.jsp

```
<!-- 댓글 등록 폼 -->
<form method="post" id="replyForm" class="replyForm">
  <input type="hidden" name="bno" value="${board.bno}">
  <ul>
    <li>
      <label>작성자</label>
      <input type="text" name="replyer" id="replyer"
        value='<security:authentication property="principal.username"/>'>
    </li>
    <li>
      <textarea rows="4" cols="60" name="reply" id="reply"></textarea>
      <button type="button" class="replyBtn">댓글 등록</button>
    </li>
  </ul>
</form>
</div>
</section>
<!-- 페이지 및 검색 전송 폼 -->
<form action="/board/boardList" method="get" id="actionForm">
```



- view 설계 – boardView.jsp

댓글

작성자: sky123 (작성일: 2022-09-19 04:52:45)

여름 가고...

작성자: admin (작성일: 2022-09-19 05:09:01)

태풍이 자주 오네요..

작성자

가을이 오고 있어요...
가을 옷을 입어야 할 거 같네요..

댓글 등록

멋글 구현

- view 설계 – css 스타일

```
/* comment 스타일 */
#view .comment{margin: 40px auto; width: 580px; text-align: left;
    border-top: 2px dashed #aaa;}
#view h4{padding: 15px 5px 5px;}
#view .replyList p{padding-left: 10px; margin: 10px;}
#view ul li{display: block; margin: 10px; line-height: 2em;}
#view textarea{float: left; margin-right: 20px;}
#view button{font-size: 0.9em; padding: 3px;}
#view .replyForm{margin: 20px auto; width: 580px; text-align: left;
    border-top: 2px dashed #aaa;}
```



댓글 구현

- view 설계 – 댓글 등록 폼 이벤트 처리

```
//댓글 등록
let replyForm = $("#replyForm");
$(".replyBtn").click(function(e){
    e.preventDefault();
    //console.log("click....");

    replyForm.attr("action", "/board/reply");
    replyForm.submit();
});
</script>
```



댓글 구현

- 댓글 개수 설계 – tbl_board에 replycnt 추가

```
CREATE TABLE tbl_board(  
    bno NUMBER(5),  
    title VARCHAR2(200) NOT NULL,  
    writer VARCHAR2(20) NOT NULL,  
    content VARCHAR2(2000) NOT NULL,  
    regdate DATE DEFAULT SYSDATE,  
    update date DATE DEFAULT SYSDATE,  
    cnt NUMBER(5) DEFAULT 0,  
    replycnt NUMBER DEFAULT 0  
);
```

-- 댓글 개수 칼럼 추가

```
ALTER TABLE tbl_board ADD replycnt NUMBER DEFAULT 0;
```



댓글 구현

- 댓글 개수 설계

```
-- 댓글 수 업데이트 (이전에 작성된 댓글 개수)  
UPDATE tbl_board  
SET replycnt =  
    (  
        SELECT COUNT(rno) FROM tbl_reply  
        WHERE tbl_reply.bno = tbl_board.bno  
    );
```



댓글 구현

- BoardVO 클래스 수정

```
@Setter
@Getter
public class BoardVO {
    private int bno;
    private String title;
    private String writer;
    private String content;

    @DateTimeFormat(pattern="yyyy-MM-dd")
    private Date regDate;
    private Date updateDate;
    private int cnt;
    private int replyCnt; //댓글 개수

    private MultipartFile uploadFile; //파일 업로드
}
```



댓글 구현

- BoardMapper 설계

```
//댓글 개수 - MyBatis는 2개의 파라미터를 사용할 수 없음. @Param 사용
public void updateReplyCnt(
    @Param("bno") int bno,
    @Param("amount") int amount
);
```

```
<!-- 댓글 개수 -->
<update id="updateReplyCnt">
    UPDATE tbl_board
    SET replycnt = replycnt + #{amount}
    WHERE bno = #{bno}
</update>
```

☞ param을 사용하지 않은 경우 오류 발생

오류가 발생했습니다.

nested exception is org.apache.ibatis.binding.BindingException: Parameter 'amount' not found. Available parameters are [arg1, arg0, param1, param2]



- BoardMapper.xml 의 목록 보기에서 replycnt 추가.

```
<!-- 목록 보기(페이징, 검색) -->
<select id="getListWithPage"
    resultType="com.cloud.domain.BoardVO">
    <![CDATA[
    SELECT * FROM
        (SELECT /*+ INDEX_DESC(tbl_board pk_board) */
            ROWNUM rn, bno, title, content, writer, regdate, updatedate, cnt, replycnt
        FROM tbl_board
        WHERE
    ]]>
    <include refid="criteria"></include>

    <![CDATA[
        ROWNUM <= #{pageNum} * #{amount})
    WHERE rn > (#{pageNum} - 1) * #{amount}
    ]]>
</select>
```



댓글 구현

- ReplyServiceImpl 트랜잭션 처리

```
@Service
public class ReplyServiceImpl implements ReplyService{

    @Autowired
    private ReplyMapper mapper;

    @Autowired
    private BoardMapper boardMapper;

    @Transactional //boardMapper와 연동함
    @Override
    public int register(ReplyVO vo) {
        //댓글 개수
        boardMapper.updateReplyCnt(vo.getBno(), 1); //1 - amount(개수)
        return mapper.register(vo);
    }

    @Override
    public List<ReplyVO> getReplyList(int bno) {
        return mapper.getReplyList(bno);
    }
}
```



댓글 구현

- 댓글 개수 출력 – boardList.jsp

```
<c:forEach var="board" items="${boardList}">
<tr>
  <td><c:out value="${board.bno}" /></td>
  <td>
    <a class="move" href='<c:out value="${board.bno}" />'>
      <c:out value="${board.title}" />
      <!-- 댓글 개수 -->
      <b>[<c:out value="${board.replyCnt}" />]</b>
    </a>
  </td>
  <td><c:out value="${board.writer}" /></td>
  <td><fmt:formatDate value="${board.regDate}"
    pattern="yyyy-MM-dd hh:mm:ss" /></td>
  <td><c:out value="${board.cnt}" /></td>
</tr>
</c:forEach>
```



댓글 구현

- 댓글 개수 출력 – boardList.jsp

<div>제목 <input type="text"/> <input type="button" value="Search"/></div>				
번호	제목	작성자	등록일	조회수
24	토성 [1]	cloud	2022-09-19 04:48:04	3
23	목성 [1]	cloud	2022-09-19 04:47:56	3
22	화성 [1]	cloud	2022-09-19 04:47:49	3
21	지구 [2]	cloud	2022-09-19 04:47:39	5



- 댓글 수정 및 삭제

댓글

작성자: admin (작성일: 2022-09-20 06:59:06)

대한민국 우주 강국

수정

삭제

작성자: sky123 (작성일: 2022-09-19 04:55:00)

지구가 최고예요

작성자: admin (작성일: 2022-09-19 04:58:27)

우주 여행 시대 가격이 너무 비싸요.

수정

삭제

작성자

댓글 등록



댓글 구현

- 댓글 수정 및 삭제

댓글 삭제

해당 댓글을 삭제하시겠습니까?

예

아니오

댓글 수정

댓글 내용

대한민국 우주 강국

저장

취소



댓글 구현

- 댓글 수정 및 삭제 버튼 만들기

```
<h4>댓글</h4>
<ol class="replyList">
  <c:forEach items="${replyList}" var="list">
    <li>
      <p class="replyer">작성자: <c:out value="${list.replyer}" />&nbsp;&nbsp;&nbsp;
        (작성일: <fmt:formatDate value="${list.updateDate}"
          pattern="yyyy-MM-dd hh:mm:ss"/>)
      </p>
      <p><c:out value="${list.reply}" /></p>
      <c:if test="${pinfo.username eq list.replyer}">
        <p>
          <button type="button" class="replyUpdateBtn" data-rno="${list.rno}">수정</button>
          <button type="button" class="replyDeleteBtn" data-rno="${list.rno}">삭제</button>
        </p>
      </c:if>
    </li>
  </c:forEach>
</ol>
```



댓글 구현

- 특정 댓글 조회, 댓글 수정 및 삭제

```
public interface ReplyMapper {  
  
    public void register(ReplyVO vo); //댓글 추가  
  
    public List<ReplyVO> getReplyList(int bno); //댓글 목록  
  
    public ReplyVO getReply(int rno); //댓글 1개 조회  
  
    public void delete(ReplyVO vo); //댓글 삭제  
  
    public void update(ReplyVO vo); //댓글 수정  
  
}
```



댓글 구현

- 특정 댓글 조회, 댓글 수정 및 삭제

```
<!-- 특정 댓글 조회 -->
<select id="getReply" resultType="com.cloud.domain.ReplyVO">
    SELECT * FROM tbl_reply WHERE rno = #{rno}
</select>

<!-- 댓글 삭제 -->
<delete id="delete">
    DELETE FROM tbl_reply WHERE rno = #{rno}
</delete>

<!-- 댓글 수정 -->
<update id="update">
    UPDATE tbl_reply
    SET reply = #{reply}, updatedate = SYSDATE
    WHERE rno = #{rno}
</update>
```



댓글 구현

- 특정 댓글 조회, 댓글 수정 및 삭제 – Service 계층

```
public interface ReplyService {  
  
    public void register(ReplyVO vo); //댓글 추가  
  
    public List<ReplyVO> getReplyList(int bno); //댓글 목록  
  
    public ReplyVO getReply(int rno); //댓글 1개 조회  
  
    public void delete(ReplyVO vo); //댓글 삭제  
  
    public void update(ReplyVO vo); //댓글 수정  
}
```



댓글 구현

- 특정 댓글 조회, 댓글 수정 및 삭제 – Service 계층

```
//특정 댓글 조회
@Override
public ReplyVO getReply(int rno) {
    return replyMapper.getReply(rno);
}

//댓글 삭제
@Override
public void delete(ReplyVO vo) {
    replyMapper.delete(vo);
}

//댓글 수정
@Override
public void update(ReplyVO vo) {
    replyMapper.update(vo);
}
```



댓글 구현

- 특정 댓글 조회, 댓글 수정 및 삭제 – BoardController 계층

```
//댓글 삭제 페이지 요청, 삭제할 대상 댓글 가져오기
@GetMapping("/replyDelete")
public String replyDeleteView(ReplyVO vo, Model model,
    RedirectAttributes rttr) {
    ReplyVO selectReply = replyService.getReply(vo.getRno());

    model.addAttribute("selectReply", selectReply);

    return "/board/replyDelete";
}

//댓글 삭제
@PostMapping("/replyDelete")
public String replyDelete(ReplyVO vo,
    RedirectAttributes rttr) {
    replyService.delete(vo);

    rttr.addAttribute("bno", vo.getBno());

    return "redirect:/board/boardView";
}
```



댓글 구현

- 특정 댓글 조회, 댓글 수정 및 삭제 – BoardController 계층

```
//댓글 수정 페이지 요청, 수정할 대상 댓글 가져오기
@GetMapping("/replyUpdate")
public String replyUpdateView(ReplyVO vo, Model model,
    RedirectAttributes rttr) {
    ReplyVO selectReply = replyService.getReply(vo.getRno());

    model.addAttribute("selectReply", selectReply);

    return "/board/replyUpdate";
}

//댓글 수정 처리
@PostMapping("/replyUpdate")
public String replyUpdate(ReplyVO vo, RedirectAttributes rttr) {
    replyService.update(vo);

    rttr.addAttribute("bno", vo.getBno());

    return "redirect:/board/boardView";
}
```



댓글 구현

- 댓글 수정 및 삭제 페이지 요청 – boardView.jsp

```
//댓글 삭제 페이지 요청
$(".replyDeleteBtn").click(function(e){
    e.preventDefault();
    console.log("click....");
    let rno = $(this).attr("data-rno");

    location.href = "/board/replyDelete?bno=${board.bno}"
                  + "&rno=" + rno;
});

//댓글 수정 페이지 요청
$(".replyUpdateBtn").click(function(e){
    e.preventDefault();
    console.log("click....");
    let rno = $(this).attr("data-rno");

    location.href = "/board/replyUpdate?bno=${board.bno}"
                  + "&rno=" + rno;
});
```



댓글 구현

- CSS 스타일 – style.css

```
/* comment 스타일 */
#view .comment{margin: 40px auto; width: 580px; text-align: left;
    border-top: 2px dashed #aaa;}
.comment h4{padding: 15px 5px 5px;}
.comment .replyList p{padding-left: 10px; margin: 5px; font-size: 0.9em}
.comment .replyList p.replyer{border-top: 1px dotted #aaa; padding: 10px;}
.comment .replyUpdateBtn, .replyDeleteBtn{font-size: 0.8em;}
.comment .replyForm{margin: 15px auto; width: 580px; text-align: left;
    border-top: 2px dashed #aaa;}
.comment ul li{display: block; margin: 10px; line-height: 2em;}
.comment textarea{float: left; margin-right: 20px;}
.comment .replyBtn{font-size: 0.9em; padding: 3px;}

/* replyDelete 스타일 */
#reply_del button{margin-top: 20px; font-size: 0.9em;
    width: 60px; height: 30px;}

/* replyUpdate 스타일 */
#reply_update .updateForm{margin: 20px auto; width: 580px;}
.updateForm ul li{display: block; margin: 10px;}
.updateForm label{float: left; padding-left: 60px;}
/* .updateForm textarea{margin-left: 10px} */
.updateForm button{font-size: 0.9em; padding: 5px;}
```



댓글 구현

- 댓글 삭제 페이지 – replyDelete.jsp

```
<div id="container">
  <section id="reply_del">
    <div class="title">
      <h2>댓글 삭제</h2>
    </div>
    <form action="/board/replyDelete" method="post" id="deleteForm">
      <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}" />
      <input type="hidden" name="bno" value="${selectReply.bno}">
      <input type="hidden" name="rno" value="${selectReply.rno}">

      <div>
        <h3>해당 댓글을 삭제하시겠습니까?</h3>
        <button type="submit" class="deleteBtn">예</button>
        <button type="button" class="cancelBtn">아니오</button>
      </div>
    </form>
  </section>
</div>
```



댓글 구현

- 댓글 수정 페이지 – replyUpdate.jsp

```
<section id="reply_update">
  <div class="title">
    <h2>댓글 수정</h2>
  </div>
  <form action="/board/replyUpdate" method="post" id="updateForm" class="updateForm">
    <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}" />
    <input type="hidden" name="bno" value="${selectReply.bno}">
    <input type="hidden" name="rno" value="${selectReply.rno}">
    <ul>
      <li>
        <label>댓글 내용</label>
        <textarea rows="4" cols="50"
          name="reply"><c:out value="${selectReply.reply}" /></textarea>
      </li>
    </ul>
    <div>
      <button type="submit" class="updateBtn">저장</button>
      <button type="button" class="cancelBtn">취소</button>
    </div>
  </form>
</section>
```



댓글 구현

- 댓글 삭제 처리 이벤트 – replyDelete.jsp

```
<script type="text/javascript">
    $(document).ready(function(){
        let deleteForm = $("#deleteForm");

        $(".cancelBtn").click(function(e){
            e.preventDefault();
            console.log("click....");

            deleteForm.attr("action", "/board/boardView?bno=${selectReply.bno}");
            deleteForm.submit();
        });
    });
</script>
```



댓글 구현

- 댓글 삭제 처리 이벤트 – replyUpdate.jsp

```
<script type="text/javascript">
    $(document).ready(function(){
        let deleteForm = $("#updateForm");

        $(".cancelBtn").click(function(e){
            e.preventDefault();
            console.log("click....");

            deleteForm.attr("action", "/board/boardView?bno=${selectReply.bno}");
            deleteForm.submit();
        });
    });
</script>
```



Rest 방식으로 전환

- REST(Representational State Transfer)란?

REST는 하나의 URI는 하나의 고유한 리소스(Resource)를 대표하도록 설계된다는 개념에 전송방식을 결합해서 원하는 작업을 지정한다.

예를 들어 `"/boards/123"`은 게시물 중에서 123번이라는 고유한 의미를 가지도록 설계하고, 이에 대한 처리는 GET, POST 방식과 같이 추가적인 정보를 통해서 결정한다.

스프링은 `@RequestBody` 또는 `@ResponseBody`와 같이 REST 방식의 데이터 처리를 위한 여러 종류의 어노테이션을 제공한다.



Rest 방식으로 전환

@RestController 사용하기

JSP같은 View를 별도로 만들지 않는 대신에 컨트롤러 메서드가 리턴한 데이터 자체를 클라이언트로 전달한다.

클라이언트에 전달되는 데이터는 대부분 문자열이거나 VO(Value Object)나 컬렉션 형태의 자바 객체인데, 자바 객체가 전달되는 경우에는 자동으로 JSON으로 변환하여 처리하게 됨

```
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
  <version>2.10.3</version>
</dependency>
<dependency>
  <groupId>com.fasterxml.jackson.dataformat</groupId>
  <artifactId>jackson-dataformat-xml</artifactId>
  <version>2.10.3</version>
</dependency>
```



Rest 방식으로 전환

- 문자열 리턴하기

```
@Log4j
@RestController
@RequestMapping("/sample")
public class SampleController {

    @GetMapping(value="/getText", produces="text/plain; charset=utf-8")
    public String getText() {
        Log.info("MIME TYPE: " + MediaType.TEXT_PLAIN_VALUE);
        return "안녕~ REST"; // "Hello~ REST" , 한글인 경우 ???로 출력
    }
}
```

← → ↻ localhost:8080/sample/getText

안녕~ REST

INFO: 이름이 []인 컨텍스트를 다시 로드하는 것을 완료했습니다.

INFO : com.cloud.controller.SampleController - MIME TYPE: text/plain



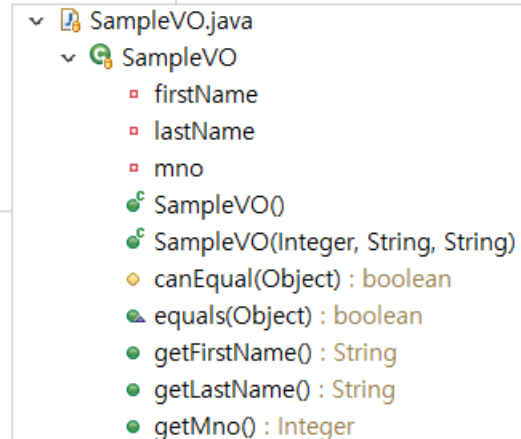
Rest 방식으로 전환

- VO 객체 리턴하기

컨트롤러 메서드가 단순히 문자열을 리턴하지 않고 VO(Value Object) 객체를 리턴하는 경우에는 VO 객체를 JSON 데이터로 변환하여 응답 프로토콜 바디에 출력한다.

```
@NoArgsConstructor    //매개변수 없는 기본 생성자
@AllArgsConstructor    //매개변수 있는 생성자
@Data
public class SampleVO {

    private Integer mno;
    private String firstName;
    private String lastName;
}
```



The image shows the IntelliJ IDEA interface. On the left, the 'Project Structure' pane displays the file 'SampleVO.java' and the class 'SampleVO'. On the right, the 'Class Hierarchy' pane shows the class 'SampleVO' with its attributes and methods. The attributes are 'firstName', 'lastName', and 'mno'. The methods are 'SampleVO()', 'SampleVO(Integer, String, String)', 'canEqual(Object) : boolean', 'equals(Object) : boolean', 'getFirstName() : String', 'getLastName() : String', and 'getMno() : Integer'.

- SampleVO.java
 - SampleVO
 - firstName
 - lastName
 - mno
 - SampleVO()
 - SampleVO(Integer, String, String)
 - canEqual(Object) : boolean
 - equals(Object) : boolean
 - getFirstName() : String
 - getLastName() : String
 - getMno() : Integer



Rest 방식으로 전환

- VO 객체 리턴하기

```
//VO 객체 리턴
@GetMapping(value="/getSample",
            produces= {MediaType.APPLICATION_JSON_VALUE,
                      MediaType.APPLICATION_XML_VALUE})
public SampleVO getSample() {
    return new SampleVO(112, "잡스", "스티브");
}
```

← → ↻ ⓘ localhost:8080/sample/getSample

This XML file does not appear to have any style

▼ <SampleVO>
 <mno>112</mno>
 <firstName>잡스</firstName>
 <lastName>스티브</lastName>
</SampleVO>

<http://localhost:8080/sample/getSample.json>
뒤에 .json을 붙이면 json 데이터로 변환됨

← → ↻ ⓘ localhost:8080/sample/getSample.json

```
{"mno":112,"firstName":"잡스","lastName":"스티브"}
```



Rest 방식으로 전환

- Collection(리스트) 객체 리턴하기

```
//컬렉션(리스트) 리턴
@GetMapping("/getList")
public List<SampleVO> getList(){
    return IntStream.range(1, 10)
        .mapToObj(i -> new SampleVO(i, i+"First", i+"Last"))
        .collect(Collectors.toList());
}
```

This XML file does not appear to have

```
▼<List>
  ▼<item>
    <mno>1</mno>
    <firstName>1First</firstName>
    <lastName>1Last</lastName>
  </item>
  ▼<item>
    <mno>2</mno>
    <firstName>2First</firstName>
    <lastName>2Last</lastName>
  </item>
  ▼<item>
    <mno>3</mno>
    <firstName>3First</firstName>
    <lastName>3Last</lastName>
  </item>
```

← → ↺ ⓘ localhost:8080/sample/getList.json

```
[{"mno":1,"firstName":"1First","lastName":"1Last"},
{"mno":4,"firstName":"4First","lastName":"4Last"},{
{"mno":7,"firstName":"7First","lastName":"7Last"},{
```



Rest 방식으로 전환

- Collection(Map) 객체 리턴하기

```
//컬렉션(Map) 리턴
@GetMapping("/getMap")
public Map<String, SampleVO> getMap(){
    Map<String, SampleVO> map = new HashMap<>();
    map.put("First", new SampleVO(112, "Bill", "Gates"));
    return map;
}
```

← → ↻ ⓘ localhost:8080/sample/getMap

This XML file does not appear to have

```
▼<Map>
  ▼<First>
    <mno>112</mno>
    <firstName>Bill</firstName>
    <lastName>Gates</lastName>
  </First>
</Map>
```

← → ↻ ⓘ localhost:8080/sample/getMap.json

```
{"First":{"mno":112,"firstName":"Bill","lastName":"Gates"}}
```



Rest 방식으로 전환

@RestController의 파라미터

@PathVariable

URL 뒤에 { }로 처리된 부분은 컨트롤러의 메서드에서 변수로 처리가 가능하다.

```
//배열로 리턴 - URL 경로 일부를 파라미터로 사용
@GetMapping("/product/{cat}/{pid}")
public String[] getpath(
    @PathVariable("cat") String cat,
    @PathVariable("pid") String pid) {
    return new String[] {"category: " + cat, "productid: " + pid};
}
```

← → ↻ ⓘ localhost:8080/sample/product/cat/1234

This XML file does not appear to have any

▼ <Strings>
 <item>category: cat</item>
 <item>productid: 1234</item>
</Strings>

← → ↻ ⓘ localhost:8080/sample/product/cat/1234.json

["category: cat", "productid: 1234"]



Rest 방식으로 전환

@RestController의 파라미터

@RequestBody

전달된 요청(request)의 내용(body)을 이용해서 해당 파라미터의 타입으로 변환을 요구한다.

대부분의 경우에는 JSON 데이터를 서버에 보내서 원하는 타입의 객체로 변환하는 용도로 사용됨

```
//@RequestBody는 ticket 객체 타입으로 변환을 요구함 - PostMapping 주의!  
@PostMapping("/ticket")  
public Ticket convert(@RequestBody Ticket ticket) {  
    Log.info("convert.....ticket" + ticket);  
    return ticket;  
}
```



Rest 방식으로 전환

Ticket 클래스 생성

```
@Data
public class Ticket {

    private int tno;           //번호
    private String owner;      //소유주
    private String grade;      //등급
}
```

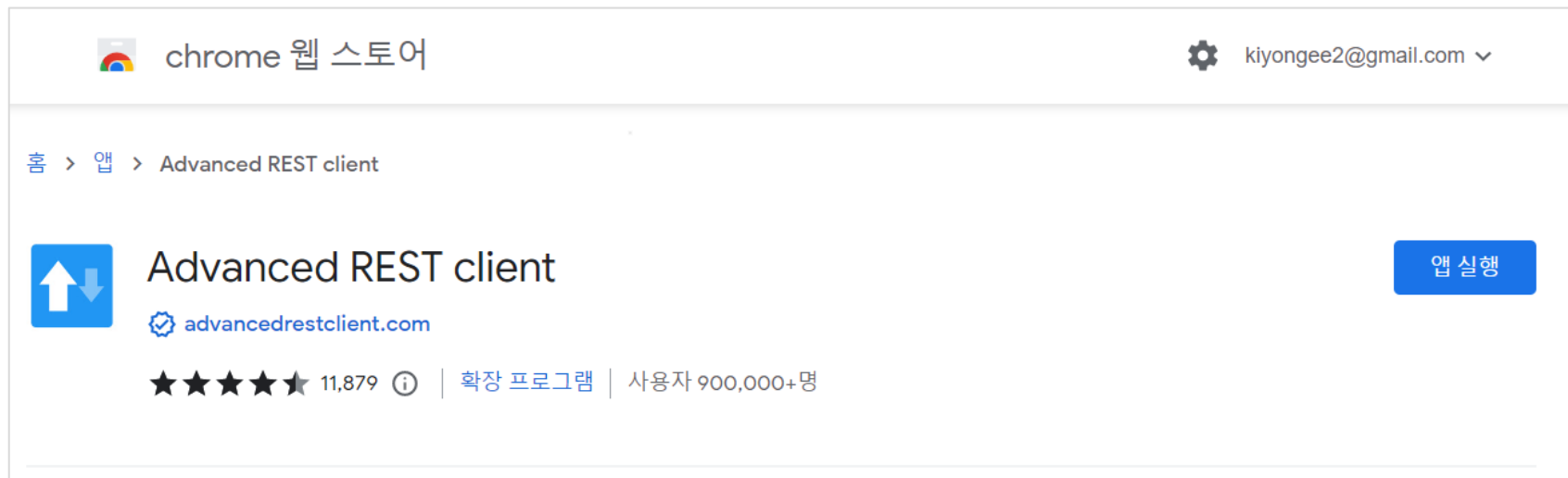


Rest 방식으로 전환

@Rest 방식의 테스트

Advanced REST client 검색

Chrome 확장도구로 추가 > chrome 웹 스토어 > 앱 실행으로 바뀜



Rest 방식으로 전환

@Rest 방식의 테스트 – Post로 보냄

Method	Request URL	
POST	http://localhost:8080/sample/ticket	<div>SEND</div>
Parameters ^		
Headers		
Body		
Variables		
Body content type	Editor view	
application/json	Raw input	
FORMAT JSON MINIFY JSON		
<pre>{"tno":123, "owner":"user1", "grade":"AAA"}</pre>		

200 OK 171.00 ms

```
.HomeController - Welcome home! The client locale is ko_KR.  
.SampleController - convert.....ticketTicket(tno=123, owner=user1, grade=AAA)
```

