

REST API 게시판과 댓글

Youtube 홍팩님의 '스프링부트 입문' 강의를 따라
라서 만든 게시판과 댓글 기능입니다.

학습용으로 한 작업으로 포트폴리오이기 보다는
제가 국비지원교육을 수료하고 나서 지속적으로
자바를 공부하였음을 증명하기 위한 자료로
제출합니다.

개발환경

개발도구 : IntelliJ IDEA Community

프론트엔드 : HTML5, CSS, Javascript, fetch, Mustache

백엔드 : Java8, Spring Boot, Gradle, JPA

데이터베이스 : H2, Postgres

서버 : Apache Tomcat 9

시작 페이지

Navbar

Home

Link

Dropdown ▾

Disabled

Search

Search

euipyo님, 반갑습니다!

```
package com.example.firstproject.controller;

import ...

@Controller
public class FirstController {

    @GetMapping("/hi")
    public String niceToMeetYou(Model model) {

        model.addAttribute(attributeName: "username", attributeValue: "euipyo");

        return "greetings"; // templates/greetings.mustache -> 브라우저로 전송!
    }

    @GetMapping("/bye")
    public String seeYouNext(Model model) {

        model.addAttribute(attributeName: "nickname", attributeValue: "euipyo");

        return "goodbye";
    }
}
```

```
{{>layouts/header}}
```

```
<!-- content -->
```

```
<div class="bg-dark text-white p-5">
```

```
<h1>{{username}}님, 반갑습니다!</h1>
```

```
</div>
```

```
{{>layouts/footer}}
```

게시판

Navbar Home Link Dropdown ▼ Disabled

ID	TITLE	CONTENT
1	가	1
2	나	2
3	다	3
4	당신의 인생 영화는?	댓글 ㄱ
5	당신의 서울 푸드는?	댓글 ㄱㄱ
6	당신의 취미는?	댓글 ㄱㄱㄱ

[New Article](#)

```
package com.example.firstproject.controller;

import ...

@Controller
@Slf4j // 로깅을 위한 골뱅이(어노테이션)
public class ArticleController {

    @Autowired // 스프링 부트가 미리 생성해놓은 객체를 가져다가 자동 연결!
    private ArticleRepository articleRepository;

    @Autowired private CommentService commentService;

    @GetMapping("/articles/new")
    public String newArticleForm() { return "articles/new"; }

    @PostMapping("/articles/create")
    public String createArticle(ArticleForm form) {
        log.info(form.toString());
        // System.out.println(form.toString()); -> 로깅 기능으로 대체!

        // 1. DTO를 변환! Entity!
        Article article = form.toEntity();
        log.info(article.toString());
        // System.out.println(article.toString());

        // 2. Repository에게 Entity를 DB안에 저장하게 함!
        Article saved = articleRepository.save(article);
        log.info(saved.toString());
        // System.out.println(saved.toString());
    }
}
```



```
package com.example.firstproject.api;

import ...

@RestController // RestAPI 용 컨트롤러! 데이터(JSON)를 반환
@Slf4j
public class ArticleApiController {

    @Autowired // DI, 생성 객체를 가져와 연결!
    private ArticleService articleService;

    // GET
    @GetMapping("/api/articles")
    public List<Article> index() { return articleService.index(); }

    @GetMapping("/api/articles/{id}")
    public Article show(@PathVariable Long id) { return articleService.show(id); }

    // POST
    @PostMapping("/api/articles")
    public ResponseEntity<Article> create(@RequestBody ArticleForm dto) {
        Article created = articleService.create(dto);
        return (created != null) ?
            ResponseEntity.status(HttpStatus.OK).body(created):
            ResponseEntity.status(HttpStatus.BAD_REQUEST).build();
    }
}
```

```
package com.example.firstproject.service;

import ...

@Slf4j
@Service // 서비스 선언! (서비스 객체를 스프링 부트에 생성)
public class ArticleService {

    @Autowired // DI
    private ArticleRepository articleRepository;

    public List<Article> index() { return articleRepository.findAll(); }

    public Article show(Long id) { return articleRepository.findById(id).orElse( other: null); }

    public Article create(ArticleForm dto) {
        Article article = dto.toEntity();

        if(article.getId() != null) {
            return null;
        }

        return articleRepository.save(article);
    }

    public Article update(Long id, ArticleForm dto) {
        // 1: 수정용 엔티티 생성!
        Article article = dto.toEntity();
        log.info(article.toString());
    }
}
```

```
package com.example.firstproject.repository;
```

```
import ...
```

```
public interface ArticleRepository extends CrudRepository<Article, Long> {  
    @Override  
    ArrayList<Article> findAll();  
}
```

```
{{>layouts/header}}
```

```
<table class="table">
```

```
  <thead>
```

```
    <tr>
```

```
      <th scope="col">ID</th>
```


```
      <th scope="col">TITLE</th>
```

```
      <th scope="col">CONTENT</th>
```

```
    </tr>
```

```
  </thead>
```

```
  <tbody>
```

```
     {{#articleList}}
```

```
      <tr>
```

```
        <th>{{id}}</th>
```

```
        <td><a href="/articles/{{id}}">{{title}}</a></td>
```

```
        <td>{{content}}</td>
```

```
      </tr>
```

```
    {{/articleList}}
```

```
  </tbody>
```

```
</table>
```

```
<a href="/articles/new">New Article</a>
```

```
{{>layouts/footer}}
```

게시글 내용 및 댓글

Navbar

HomeLinkDropdown ▼Disabled

Search

Search

ID	TITLE	CONTENT
4	당신의 인생 영화는?	댓글 7

EditDeleteGo to Article List

Park수정삭제

곧 월 헌팅

Kim수정삭제

아이 엠 샘

닉네임

댓글 내용

댓글 작성

```
package com.example.firstproject.api;

import ...

@RestController
public class CommentApiController {

    @Autowired private CommentService commentService;

    // 댓글 목록 조회
    @GetMapping("/api/articles/{articleId}/comments")
    public ResponseEntity<List<CommentDto>> comments(@PathVariable Long articleId) {
        // 서비스에게 위임
        List<CommentDto> dtos = commentService.comments(articleId);

        // 결과 응답
        return ResponseEntity.status(HttpStatus.OK).body(dtos);
    }

    // 댓글 생성
    @PostMapping("/api/articles/{articleId}/comments")
    public ResponseEntity<CommentDto> create(@PathVariable Long articleId,
                                             @RequestBody CommentDto dto) {

        // 서비스에게 위임
        CommentDto createdDto = commentService.create(articleId, dto);

        // 반환
        return ResponseEntity.status(HttpStatus.OK).body(createdDto);
    }
}
```

```
@Transactional
```

```
public CommentDto create(Long articleId, CommentDto dto) {  
    // 게시글 조회 및 예외 발생  
    Article article = articleRepository.findById(articleId)  
        .orElseThrow(() -> new IllegalArgumentException("댓글 생성 실패! 대상 게시글이 없습니다"));  
  
    // 댓글 엔티티 생성  
    Comment comment = Comment.createComment(dto, article);  
  
    // 댓글 엔티티를 DB로 저장  
    Comment created = commentRepository.save(comment);  
  
    // DTO로 변경하여 반환  
    return CommentDto.createCommentDto(created);  
}
```

```
<div class="card m-2" id="comments-new">

  <div class="card-body">
    <!-- 댓글 작성 폼 -->
    <form>
      <!-- 닉네임 입력 -->
      <div class="mb-3">
        <label class="form-label">닉네임</label>
        <input class="form-control form-control-sm" id="new-comment-nickname">
      </div>
      <!-- 댓글 본문 입력 -->
      <div class="mb-3">
        <label class="form-label">댓글 내용</label>
        <textarea class="form-control form-control-sm" rows="3" id="new-comment-body"></textarea>
      </div>

      <!-- 히든 인풋 -->
      {{#article}}
        <input type="hidden" id="new-comment-article-id" value="{{id}}">
      {{/article}}

      <!-- 전송 버튼 -->
      <button type="button" class="btn btn-outline-primary btn-sm" id="comment-create-btn">댓글 작성</button>
    </form>
  </div>
</div>
```


댓글 수정

Navbar

HomeLinkDropdown ▼Disabled

SearchSearch

ID	TITLE	CONTENT
4	당신의 인생 영화는?	댓글 7
<div>EditDeleteGo to Article List</div>		
Park	<div>수정삭제</div>	
굳 윌 헌팅		
Kim	<div>수정삭제</div>	
아이 엠 샘		
<div>닉네임</div> <div></div> <div>댓글 내용</div> <div></div> <div>댓글 작성</div>		

댓글 수정

✕

닉네임

Kim

댓글 내용

아이 엠 샘

수정 완료

© CloudStudying · Privacy · Terms

```
// 댓글 수정
```

```
@PatchMapping("/api/comments/{id}")
```

```
public ResponseEntity<CommentDto> update(@PathVariable Long id,  
                                           @RequestBody CommentDto dto) {
```

```
    // 서비스에게 위임
```

```
    CommentDto updatedDto = commentService.update(id, dto);
```

```
    // 결과 응답
```

```
    return ResponseEntity.status(HttpStatus.OK).body(updatedDto);
```

```
}
```

```
// 댓글 삭제
```

```
@RunningTime
```

```
@DeleteMapping("/api/comments/{id}")
```

```
public ResponseEntity<CommentDto> delete(@PathVariable Long id) {
```

```
    // 서비스에게 위임
```

```
    CommentDto deletedDto = commentService.delete(id);
```

```
    // 결과 응답
```

```
    return ResponseEntity.status(HttpStatus.OK).body(deletedDto);
```

```
}
```

```
@Transactional
public CommentDto update(Long id, CommentDto dto) {
    // 댓글 조회 및 예외 발생
    Comment target = commentRepository.findById(id)
        .orElseThrow(() -> new IllegalArgumentException("댓글 수정 실패! 대상 댓글이 없습니다.));

    // 댓글 수정
    target.patch(dto);

    // DB로 갱신
    Comment updated = commentRepository.save(target);

    // 댓글 엔티티를 DTO로 변환 및 반환
    return CommentDto.createCommentDto(updated);
}
```

```
@Transactional
public CommentDto delete(Long id) {
    // 댓글 조회 및 예외 발생
    Comment target = commentRepository.findById(id)
        .orElseThrow(() -> new IllegalArgumentException("댓글 삭제 실패! 대상이 없습니다.));

    // 댓글 DB에서 삭제
    commentRepository.delete(target);

    // 삭제 댓글을 DTO로 반환
    return CommentDto.createCommentDto(target);
}
```

```
<script>
{
  // 댓글 생성 버튼 변수화(id가 comment-create-btn인 대상)
  const commentCreateBtn = document.querySelector("#comment-create-btn");

  // 버튼 클릭 이벤트를 감지!
  commentCreateBtn.addEventListener("click", function() {

    // 새 댓글 객체 생성
    const comment = {
      nickname : document.querySelector('#new-comment-nickname').value,
      body : document.querySelector('#new-comment-body').value,
      article_id : document.querySelector('#new-comment-article-id').value
    };

    // 댓글 객체 출력
    console.log(comment);

    // fetch() - Talend API 요청을 JavaScript로 보내준다!
    const url = "/api/articles/" + comment.article_id + "/comments";
    fetch(url, {
      method : "post",    // POST 요청
      body : JSON.stringify(comment),    // comment JS객체를 JSON으로 변경하여 보냄
      headers : {
        "Content-Type" : "application/json"
      }
    }).then(response => {
      // http 응답 코드에 따른 메시지 출력
      const msg = (response.ok) ? "댓글이 등록되었습니다" : "댓글 등록 실패...!";
      alert(msg);
    });
  });
}
```

AOP - 로깅

```
@Aspect // AOP 클래스 선언: 부가 기능을 주입하는 클래스
@Component // Ioc 컨테이너가 해당 객체를 생성 및 관리
@Slf4j
public class DebuggingAspect {

    // 대상 메소드 선택: CommentService#create()
    @Pointcut("execution(* com.example.firstproject.service.CommentService.*(..))")
    private void cut() {}

    // 실행 시점 설정: cut()의 대상이 수행되기 이전
    @Before("cut()")
    public void loggingArgs(JoinPoint joinPoint) { // cut()의 대상 메소드
        // 입력값을 가져오기
        Object[] args = joinPoint.getArgs();

        // 클래스명
        String className = joinPoint.getTarget().getClass().getSimpleName();

        // 메소드명
        String methodName = joinPoint.getSignature().getName();

        // 입력값 로깅하기
        // CommentService#create()의 입력값 => 5
        // CommentService#create()의 입력값 => CommentDto(id=null, ...)
        for (Object obj : args) { // foreach 문
            log.info("{}#{}의 입력값 => {}", className, methodName, obj);
        }
    }
}
```

```
// 실행 시점 설정: cut()에 지정된 대상 호출 성공 후!
@AfterReturning(value = "cut()", returning = "returnObj")
public void loggingReturnValue(JoinPoint joinPoint, // cut()의 대상 메소드
                               Object returnObj) { // 리턴값

    // 클래스명
    String className = joinPoint.getTarget() Object
        .getClass() Class<capture of ? extends Object>
        .getSimpleName();

    // 메소드명
    String methodName = joinPoint.getSignature()
        .getName();

    // 반환값 로깅
    // CommentService#create()의 반환값 => CommentDto(id=10, ....)
    log.info("{}#{ }의 반환값 => {}", className, methodName, returnObj);
}
```

AOP - 실행시간 측정

```
package com.example.firstproject.annotation;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

// 궁금하면 구글링!
@Target({ElementType.TYPE, ElementType.METHOD}) // 어노테이션 적용 대상
@Retention(RetentionPolicy.RUNTIME) // 어노테이션 유지 기간
public @interface RunningTime {
}
```

```
@Aspect
@Component
@Slf4j
public class PerformanceAspect {

    // 특정 어노테이션을 대상 지정
    @Pointcut("@annotation(com.example.firstproject.annotation.RunningTime)")
    private void enableRunningTime() {}

    // 기본 패키지의 모든 메소드
    @Pointcut("execution(* com.example.firstproject..*.*(..))")
    private void cut() {}

    // 실행 시점 설정: 두 조건을 모두 만족하는 대상을 전후로 부가 기능을 삽입
    @Around("cut() && enableRunningTime()")
    public void loggingRunningTime(ProceedingJoinPoint joinPoint) throws Throwable {
        // 메소드 수행 전, 측정 시작
        Stopwatch stopwatch = new Stopwatch();
        stopwatch.start();

        // 메소드를 수행
        Object returningObj = joinPoint.proceed();

        // 메소드 수행 후, 측정 종료 및 로깅
        stopwatch.stop();

        // 메소드명
        String methodName = joinPoint.getSignature()
            .getName();

        log.info("{}dml 총 수행 시간 => {} sec", methodName, stopwatch.getTotalTimeSeconds());
    }
}
```