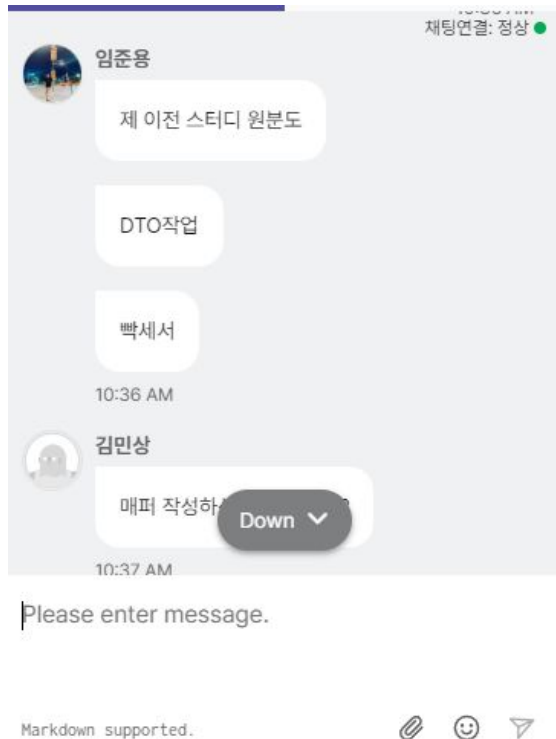
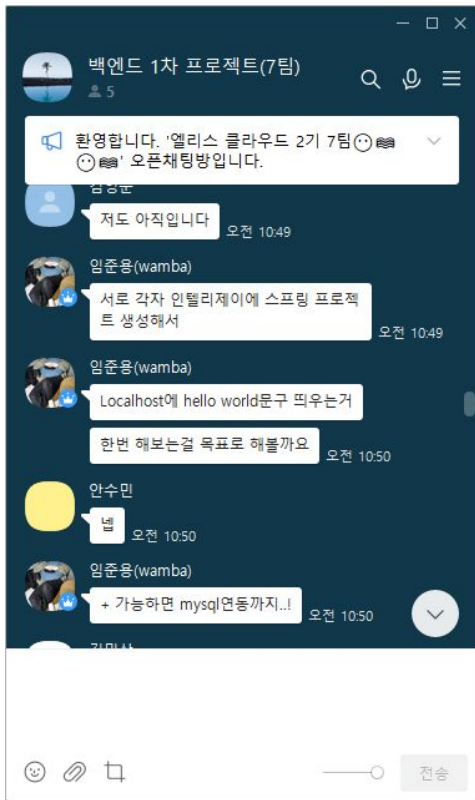

elice CRUD project

Team #07

2024-03-02

팀 활동

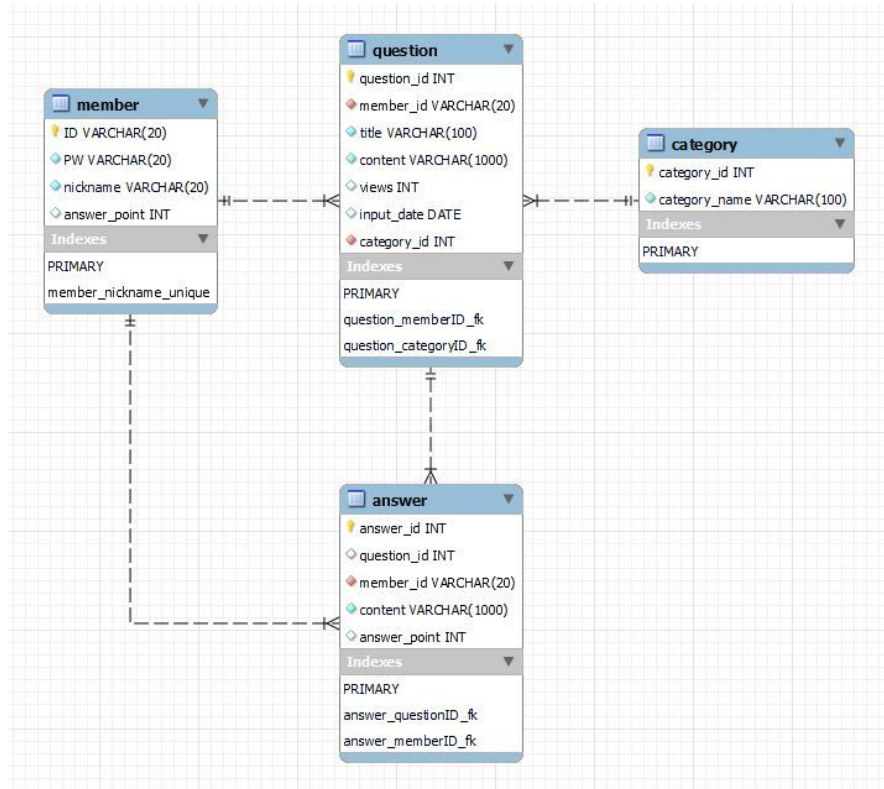
- 팀 활동 채널: 카카오톡 채팅, 엘리스 플랫폼 강의실



프로젝트 공통 순서

기능명세서 및 ERD 설계도를 작성한 후 코드 작성 진행

기능	설명
회원가입	ID,PW,닉네임을 입력받아 DB에 저장 ID 및 닉네임의 중복은 회원가입 화면의 버튼으로 확인 ID는 특수문자가 들어갈 수 없음
로그인	ID,PW를 입력받아 DB에 저장되어 있는 데이터와 비교하여 성공/실패 확인 실패시 메시지 : 아이디 또는 비밀번호가 틀렸습니다.
질문글 작성	질문등록 Button을 클릭 시 질문등록 페이지로 이동 - 비로그인 상태일 때 : alert(로그인이 필요합니다) 제목,내용을 받아 DB에 저장 - 제목 또는 내용이 입력되지 않았을 때 : alert(OO를 입력해주세요)
질문글 조회	초기화면은 한 페이지에 10개의 질문글이 출력 됨 각 게시글은 게시글 번호, 제목, 닉네임, 조회수, 작성시간(YYYY-MM-DD)이 표시 됨 게시글의 제목 클릭시 상세내용 출력 - 제목, 내용, 질문자 닉네임, 조회수, 답변, 답변자 닉네임
답변 작성	게시글 상세내용 페이지 하단의 text입력 form에 답변을 적고 답변하기 Button을 클릭 시 답변등록 됨 - form에 내용이 입력되지 않았을 때 : alert(내용을 입력해주세요) - 비로그인 상태일 때 : alert(로그인이 필요합니다) 등록 완료시 답변내용과 답변자의 닉네임이 출력됨
답변 선택	질문자는 도움이 된 답변중 하나를 선택할 수 있음 - 선택된 답변은 답변들 중 최상단에 노출되고 배경색이 채색됨 - 선택된 답변은 좋아요 Button이 활성화 되며 클릭된 수 만큼 점수를 얻음 - 좋아요 Button은 로그인된 유저만 1회 클릭가능
게시글 수정	답변이 등록된 질문글은 수정 불가 본인이 작성한 답변만 수정가능하며 수정 Button이 표시됨 수정 Button 클릭 시 text입력 form이 생기며 기존의 내용이 form에 기본값으로 입력됨
게시글 삭제	답변이 등록된 질문글은 삭제 불가 답변이 없는 질문글은 삭제 Button을 클릭 시 alert(삭제하시겠습니까? Y/N) 본인이 작성한 답변만 삭제가능하며 삭제 Button이 표시됨
점수표시	닉네임 좌측에 점수별 이미지가 표시됨



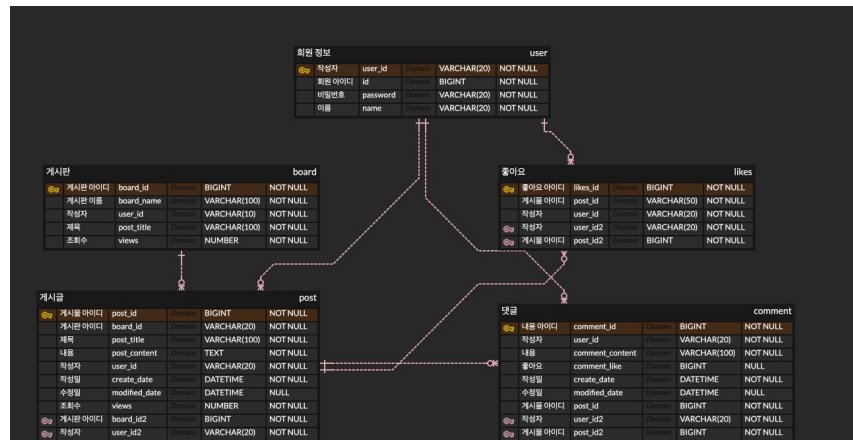
김민상

여행 정보 게시판

프로젝트 기획

5. 게시판 메인 페이지

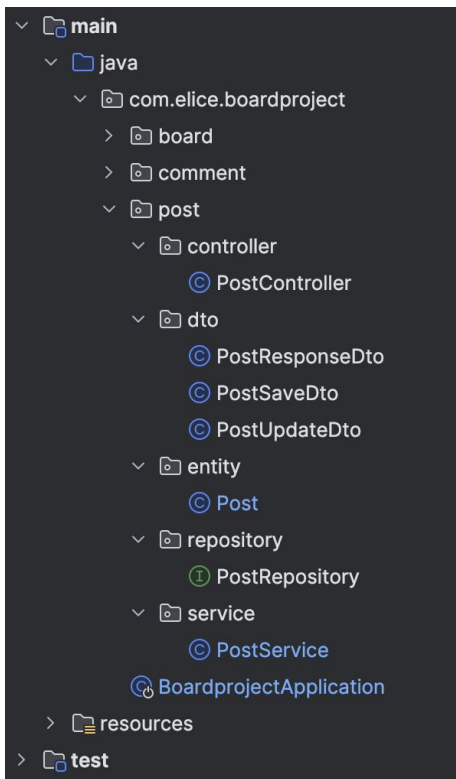
게시글 작성 및 수정	<ul style="list-style-type: none"> - 게시글 작성, 수정 시 제목과 내용은 공백 혹은 빈칸으로 작성하지 않도록 하기 - 내가 작성한 글만 수정 및 삭제 가능 - 최신순으로 정렬 - 게시글 제목 클릭 시 게시글 페이지로 이동 - 게시글 작성일 yyyy/mm/dd 23:59로 표기 - 생성 시간, 수정 시간 DB에 기록
코멘트	<ul style="list-style-type: none"> - 댓글은 로그인한 사용자만 작성 가능 - 댓글 작성 및 수정 시 빈칸 혹은 공백인 경우 "댓글을 작성해 주세요" 출력 - 댓글 수정 및 삭제는 댓글 작성자만 가능 - 게시글 삭제 시 코멘트도 같이 삭제
검색	<ul style="list-style-type: none"> - 제목 검색 : 제목에 해당 검색어가 포함된 게시글 모두 조회 - 내용 검색 : 내용에 해당 검색어가 포함된 게시글 모두 조회 - 작성자 검색 : 작성자 닉네임이 검색어와 정확히 일치할 경우 조회 - 제목 + 내용 검색 : 제목 또는 내용에 검색어가 포함된 게시글 모두 조회
유효성 검사	<ul style="list-style-type: none"> - 제목, 내용이 빈칸일 경우 "공백 또는 입력하지 않은 부분이 있습니다" 출력 - 수정 시에도 동일하게 적용



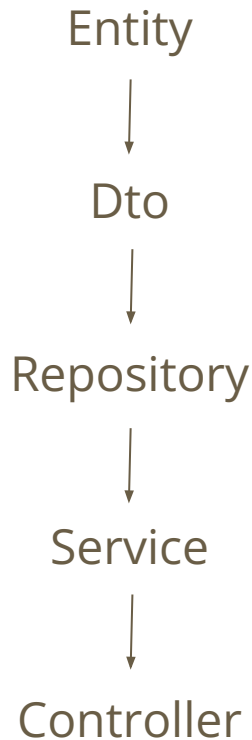
기능
명세서

ERD

코드 작성 과정



```
public class PostService {  
    6 usages  
    private final PostRepository postRepository;  
  
    ± blaumonde  
    public PostService(PostRepository postRepository) {  
        this.postRepository = postRepository;  
    }  
  
    1 usage ± blaumonde  
    @Transactional  
    public List<Post> getAllPost() {  
        return postRepository.findAll();  
    }  
  
    //조회  
    1 usage ± blaumonde  
    @Transactional  
    public PostResponseDto getPostById(Long id) {  
        Post post = postRepository.findById(id).orElseThrow  
        (() -> new IllegalArgumentException("존재하지 않는 게시물입니다. id =" + id));  
        return new PostResponseDto(post);  
    }  
  
    //생성  
    1 usage ± blaumonde  
    @Transactional  
    public Post save(PostSaveDto post) {  
        return postRepository.save(post.toEntity());  
    }  
}
```



Trouble Shooting

```
12 4 usages new *
13 @Service
14 public class PostService {
15     6 usages
16     private final PostRepository postRepository;
17     new *
18     public PostService(PostRepository postRepository) { this.postRepository = postRepository; }
19
20     1 usage new *
21     @Transactional
22     public List<Post> getAllPost() { return postRepository.findAll(); }
23
24     //조회
25     1 usage new *
26     @Transactional
27     public PostResponseDto getPostById(Long id) {
28         Post post = postRepository.findById(id).orElseThrow
29             (() -> new IllegalArgumentException("존재하지 않는 게시물입니다. id = " + id));
30         return new PostResponseDto(post);
31     }
32
33     //생성
34     1 usage new *
35     @Transactional
36     public Post save(PostSaveDto post) { return postRepository.save(post.toEntity()); }
37
38     //수정
39     1 usage new * 1 related problem
40     @Transactional
41     public Post update(Long id, PostUpdateDto updatedPost) {
42         Post post = postRepository.findById(id).orElseThrow
43             (() -> new IllegalArgumentException("존재하지 않는 게시물입니다. id = " + id));
44         post.update(updatedPost.getTitle(), updatedPost.getContent());
45         return post;
46     }
47 }

14 1 blaumonde
15 @RequiredArgsConstructor
16 @Controller
17 public class BoardController {
18     5 usages
19     private final BoardService boardService;
20
21     //게시판 생성
22     1 blaumonde
23     @GetMapping("/")
24     public String save(@RequestBody BoardSaveDto saveDto) {
25         return boardService.save(saveDto);
26     }
27
28     //수정
29     1 blaumonde 1 related problem
30     @PutMapping("/{id}")
31     public String update(@PathVariable Long id,
32                         @RequestBody BoardUpdateDto dto) {
33         return boardService.update(id, dto);
34     }
35
36     //삭제
37     1 blaumonde
38     @DeleteMapping("/{id}")
39     public String delete(@PathVariable Long id) { return boardService.delete(id); }
40
41     //조회
42     1 blaumonde
43     @GetMapping("/{id}")
44     public BoardResponseDto findById(@PathVariable Long id) {
45         BoardResponseDto responseDto = boardService.findById(id);
46         return responseDto;
47     }
48
49     //전체 게시판 조회
50     1 blaumonde
51     @GetMapping("/")
52     public String getAllPost() { return postService.getAllPost(); }
53
54     //생성
55     1 blaumonde
56     @PostMapping("/{id}/create")
57     public Post createPost(@RequestBody PostSaveDto postSaveDto) {
58         return postService.save(postSaveDto);
59     }
60
61     //수정
62     1 blaumonde
63     @PutMapping("/{id}/update")
64     public Post updatePost(@PathVariable Long id,
65                           @RequestBody PostUpdateDto updatedPost) {
66         return postService.update(id, updatedPost);
67     }
68 }
```

© 2014 Pearson Education, Inc. or its affiliate(s). All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or by any information storage or retrieval system, without prior written permission from Pearson Education, Inc. or its affiliate(s).

OVERVIEW

개선할 점

- 코드를 작성하기 전 설계를 제대로 하지 않아 필드명과 컬럼명을 알맞게 매치하지 않은 것
- 회원가입, 로그인 등 회원 기능 구현하지 못한 것
- 페이징 처리하지 못한 것

소감

프로젝트 진행 중 데이터베이스 관련 오류로 인해 백엔드와 프론트엔드의 연결이 원활하지 않아 결론적으로 웹페이지가 실행되지 않았지만 그동안 배웠던 내용을 실제로 개발 단계에 적용해 볼 수 있었던 좋은 경험이었다는 것 같습니다

김영준


커뮤니티 게시판


코드 작성 과정


회원가입 기능

```
@Transactional
public void signUP(UserSignUpRequest userSignUpRequest) {
    validate(userSignUpRequest);
    User user = User.builder()
        .userId(new UserId(userSignUpRequest.getUserId()))
        .email(userSignUpRequest.getEmail())
        .password(userSignUpRequest.getPassword())
        .build();
    userRepository.save(user);
}
```

Registration

 Id

 Email

 PassWord

☐ I agree to the terms & conditions

Already have an account? [Login](#)

코드 작성 과정

로그인 기능

```
@Transactional
public void login(UserLoginRequest loginRequest) {
    User user = userRepository.findByUserId(loginRequest.getUserId())
        .orElseThrow(InvalidUserNameException::new);

    if (!user.getPassword().equals(loginRequest.getPassword())) {
        throw new InvalidPasswordException();
    }

    HttpSession.setAttribute(s: "user", user);
}
```

Login

Id

Password

☐ 아이디 자동 로그인 [Forgot Password?](#)

Don't have an account? [Register](#)

시연 영상

X

Registration

 Id

 Email

 PassWord

☐ I agree to the terms & conditions

Already have an account? [Login](#)

개선할 점

```
2024-02-29T21:15:43.860+09:00 ERROR 29296 --- [io-8080-exec-10] o.a.c.c.C.[.[/].[dispatcherServlet] : Servlet.service() for servlet [dispatcherServlet] in context with path [] threw exception [Request processing failed: org.springframework.dao.InvalidDataAccessApiUsageException: Argument [test1] of type [java.lang.String] did not match parameter type [elicecrud.community.user.entity.UserId (n/a)]] with root cause
```

로그인시 발생하는 문제.

Mysql의 User테이블의 user_name컬럼과 UserId클래스가 매치가 안되는 오류.

엔티티 개발과 컨트롤러, 서비스등을 개발하고 프론트와 연결하려고 할때 에러가 많이난다.

또한 기능을 개발할때 구체적으로 구상해두지않아 에러가 많이 발생한다.

먼저 구상을 철저하게 한 뒤 개발을 해야할 꺼 같다.

안수민

질문 답변 게시판

코드 작성 과정 - 메인화면

글쓰기	카테고리 생성	회원가입	로그인
일반상식	코딩	영화	운동
제목11	코딩질문2	영화질문1	운동질문2
제목9	코딩질문	제목7	운동질문1
제목8	제목10	제목3	
제목5	제목6		
제목4	제목2		

메인화면에 카테고리 별로
최신글 5개씩 출력하기 위한
쿼리문
- JPA로 하기엔 복잡한 쿼리를
NativeQuery를 이용하여 처리

```
@Query(value = "select * from (select * , row_number() over " +  
    "(partition by category_id order by question_id desc) as rn from question) q " +  
    "where q.rn <= 5",nativeQuery = true)  
List<Question> findTop5QuestionsGroupByCategory();
```

코드 작성 과정 - 회원가입

회원가입	
아이디 입력	
비밀번호 입력	
닉네임 입력	
<input type="button" value="가입하기"/>	<input type="button" value="취소"/>

회원가입	
memberA	
•	
nameA	
<input type="button" value="가입하기"/>	<input type="button" value="취소"/>

회원가입	
아이디 입력	
비밀번호 입력	
닉네임 입력	
<input type="button" value="가입하기"/>	<input type="button" value="취소"/>
사용중인 아이디 또는 닉네임입니다.	

```
@PostMapping("/signUpForm")
public String addMember(@ModelAttribute MemberDto memberDto, RedirectAttributes redirectAttributes){
    Boolean signUpCheck = memberService.createMember(memberDto.toEntity());
    if(!signUpCheck){
        redirectAttributes.addAttribute("attributeName: signUp", "attributeValue: "사용중인 아이디 또는 닉네임입니다.");
        return "redirect:/signUpForm";
    }
    return "redirect:/";
}

public Boolean createMember(Member member){
    Member memberIdCheck = memberRepository.findById(member.getMemberId()).orElse(null);
    Member memberNameCheck = memberRepository.findByName(member.getNickName());
    if(!(Objects.isNull(memberIdCheck)) || !(Objects.isNull(memberNameCheck))){
        return false;
    }
    memberRepository.save(member);
    return true;
}
```

회원가입 화면

서비스에서 데이터가 있는지
판별 후 Boolean값 리턴

컨트롤러에서
RedirectAttributes로
사용중이라는 메시지를
전달해줌

코드 작성 과정 - 로그인

로그인

회원가입과 마찬가지로 아이디와 패스워드를
판별 후 데이터가 일치하면 세션에 멤버를 저장해 줌

```
@PostMapping("/login")
public String login(@ModelAttribute MemberDto memberDto, HttpServletRequest request, RedirectAttributes redirectAttributes){
    Boolean loginMember = memberService.login(memberDto.toEntity());
    if(!loginMember){
        redirectAttributes.addAttribute( attributeName: "login", attributeValue: "로그인에 실패 했습니다.");
        return "redirect:/loginForm";
    }
    Member memberSession = memberDto.toEntity();
    HttpSession session = request.getSession();
    session.setAttribute( s: "member", memberSession);
    return "redirect:/";
}
```

코드 작성 과정 - 삭제

타임리프로 삭제할 질문의 ID값을 넘겨주는 코드

```
<button id="deleteBtn" th:onclick="|location.href='@{/question/delete/{questionId}}(questionId=${question.questionId})'|" class="slide">삭제</button>
```

```
@GetMapping("/{questionId}")
```

```
public String deleteQuestion(@PathVariable("questionId") Long questionId){  
    questionService.deleteQuestion(questionId);  
    return "redirect:/";  
}
```

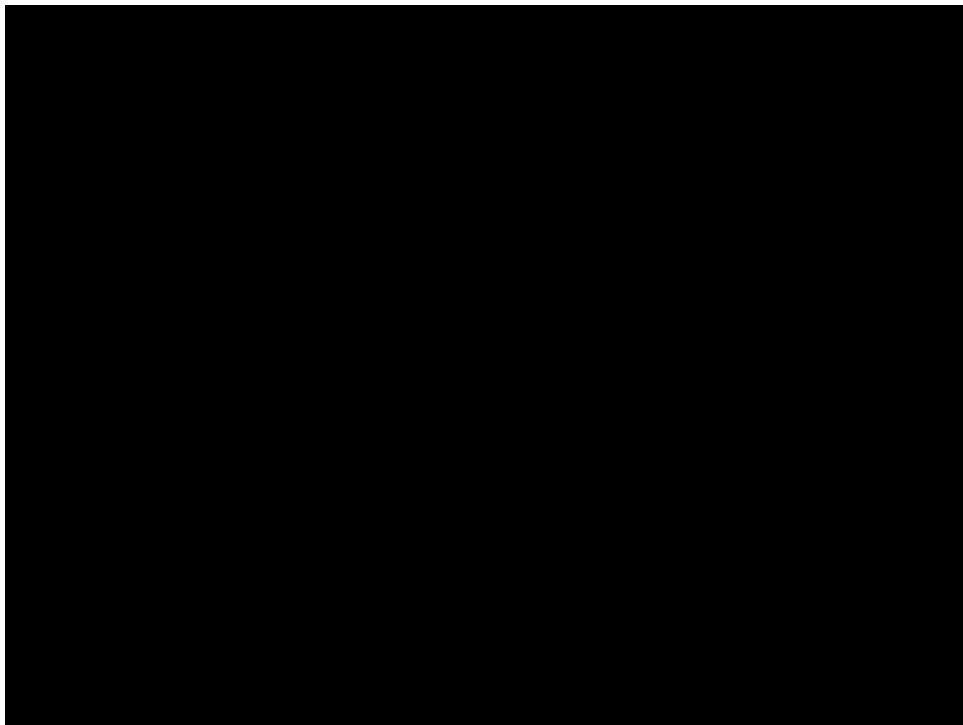
Controller 코드

```
@Transactional
```

```
public void deleteQuestion(Long questionId){  
    answerRepository.deleteByQuestion_QuestionId(questionId);  
    Question question = questionRepository.findById(questionId)  
        .orElseThrow(() -> new RuntimeException("잘못된 접근"));  
  
    questionRepository.delete(question);  
}
```

답변테이블이 질문테이블의 ID를
외래키로 사용하기 때문에
넘겨받은 question ID 값을 외래키로 갖는
답변의 레코드들을 먼저 삭제해주고
질문을 삭제함

시연 영상



질문하기 및 답변하기 시연

개선할 점 -

구현할 때 깊이 생각하지 않고 떠오르는 대로 코딩을 해서

개인 프로젝트 임에도 코드를 다시 해석하는 일이 생김

설계에 더 많은 시간을 투자 해야겠다는 반성을 함

임준용

피시방 음식 레시피 게시판

Board 도메인 제작 과정 - Repository, Service

```
private JdbcTemplate jdbcTemplate;

Wambatcode
@Override
public void save(Board board) {
    jdbcTemplate.update(sql: "INSERT INTO board (board_title, content, filename, filepath, user_id, password) VALUES (?, ?, ?, ?, ?, ?)",
        board.getBoardTitle(),
        board.getContent(),
        board.getFilename(),
        board.getFilePath(),
        board.getUser().getUserId(),
        board.getUser().getPassword());
}

Wambatcode *
@Override
public void update(Board board) {
    jdbcTemplate.update(sql: "UPDATE board SET board_title = ?, content = ?, filename = ?, filepath = ?" +
        "user_id = ?, password = ? WHERE id = ?",
        board.getBoardTitle(),
        board.getContent(),
        board.getFilename(),
        board.getFilePath(),
        board.getUser().getUserId(),
        board.getUser().getPassword(),
        board.getId());
}

Wambatcode
@Override
public void deleteById(Long boardId) { jdbcTemplate.update(sql: "DELETE FROM board WHERE id = ?", boardId); }

Wambatcode
@Override
public Board findById(Long boardId) { [Complexity is 3 Everything is cool]
    List<Board> result = jdbcTemplate.query(sql: "SELECT * FROM board WHERE id = ?",
        new Object[]{boardId},
        new BeanPropertyRowMapper<>(Board.class));

    return result.isEmpty() ? null : result.get(0);
}
```

```
@Service
@Transactional
public class BoardService { [Complexity is 3 Everything is cool]
    private final BoardRepository boardRepository;
    private final BoardFileService boardFileService;
    private final UserService userService;

    Wambatcode
    public BoardService(BoardRepository boardRepository, BoardFileService boardFileService, UserService userService){
        this.boardRepository = boardRepository;
        this.boardFileService = boardFileService;
        this.userService = userService;
    }

    Wambatcode
    public void saveBoard(Board board, MultipartFile file, String userId) throws IOException {
        User user = userService.findUserByUserId(userId);
        board = boardFileService.fileUpload(file, board);
        board.setUser(user);
        boardRepository.save(board);
    }

    Wambatcode
    public void updateBoard(Board board, MultipartFile file, String userId) throws IOException {
        User user = userService.findUserByUserId(userId);
        board = boardFileService.fileUpload(file, board);
        board.setUser(user);
        boardRepository.update(board);
    }

    Wambatcode
    public void deleteBoardById(Long boardId) { boardRepository.deleteById(boardId); }

    Wambatcode
    public Board findBoardById(Long boardId) { return boardRepository.findById(boardId); }

    Wambatcode
    public List<Board> findAllBoards() { return boardRepository.findAll(); }
}
```

Board 도메인 제작 과정 - Controller

```
BoardController.java
UserController.java
BoardRepositoryImpl.java
BoardService.java
UserService.java

@GetMapping("/")
public String getAllBoards(Model model) {
    List<Board> boards = boardService.findAllBoards();
    List<BoardResponseDTO> boardResponseDTOS = boards.stream().Stream<Board>
        .map(boardMapper::boardToBoardResponseDTO) .Stream<BoardResponseDTO>
        .collect(Collectors.toList());
    model.addAttribute("boards", boardResponseDTOS);
    return "board/boards";
}

//Wambatcode
@GetMapping("/{id}")
public String getBoard(@PathVariable Long id,
    @RequestParam(defaultValue = "0") int page,
    @RequestParam(defaultValue = "10") int size,
    @RequestParam(required = false) String keyword,
    Model model) {
    Board board = boardService.findBoardById(id);
    PageRequest pageRequest = PageRequest.of(page, size);
    Page<Post> postPage = postService.findPostsByBoardAndKeyword(board, keyword, pageRequest);
    BoardResponseDTO boardResponseDTO = boardMapper.boardToBoardResponseDTO(board);

    model.addAttribute("board", boardResponseDTO);
    model.addAttribute("keyword", keyword);
    model.addAttribute("postPage", postPage);
    return "board/board";
}

//Wambatcode
@GetMapping("/{create}")
public String showCreateForm(Model model) {
    model.addAttribute("boardRequestDTO", new BoardRequestDTO());
    return "board/createBoard";
}

//Wambatcode
@PostMapping(value = "{create}")
public String saveBoard(@ModelAttribute("board") BoardRequestDTO boardRequestDTO, MultipartFile file, String userId) throws IOException {
    Board board = boardMapper.boardRequestDTOToBoard(boardRequestDTO);
    boardService.saveBoard(board, file, userId);
    return "redirect:/boards";
}

//Wambatcode
@GetMapping("/{id}/edit")
public String showUpdateForm(@PathVariable("id") Long id, Model model) {
    Board board = boardService.findBoardById(id);
    if (board == null) {
        return "error";
    }
    BoardResponseDTO boardResponseDTO = boardMapper.boardToBoardResponseDTO(board);
    model.addAttribute("boardResponseDTO", boardResponseDTO);
    return "board/editBoard";
}

//Wambatcode
@PostMapping(value = "{id}/edit")
public String updateBoard(@PathVariable("id") Long id, @ModelAttribute("board") BoardRequestDTO boardRequestDTO,
    String userId, String password, MultipartFile file) throws IOException {
    Board board = boardMapper.boardRequestDTOToBoard(boardRequestDTO).toBuilder().id(id).build();
    if (userService.login(userId, password) != null){
        boardService.updateBoard(board, file, userId);
        return "redirect:/boards";
    }else{
        return "redirect:/boards/{id}/edit";
    }
}

/**@PostMapping(value = "{id}/edit", consumes = MediaType.APPLICATION_JSON_VALUE)
public String updateBoard_1(@PathVariable("id") Long id, @RequestBody BoardRequestDTO boardRequestDTO) {
    Board board = boardMapper.boardRequestDTOToBoard(boardRequestDTO).toBuilder().id(id).build();
    boardService.updateBoard(board);
    return "redirect:/boards";
}*/

//Wambatcode
@DeleteMapping("/{id}/delete")
public String deleteBoard(@PathVariable("id") Long id) {
    boardService.deleteBoardById(id);
    return "redirect:/boards";
}
}
```

Board 도메인 제작 과정 - Controller

```
Wombatcode
@PostMapping(value = "/{id}/edit")
public String updateBoard(@PathVariable("id") Long id, @ModelAttribute("board") BoardRequestDTO boardRequestDTO,
                          String userId, String password, MultipartFile file) throws IOException {
    Board board = boardMapper.boardRequestDTOToBoard(boardRequestDTO).toBuilder().id(id).build();
    if(userService.login(userId, password) != null){
        boardService.updateBoard(board, file, userId);
        return "redirect:/boards";
    }else{
        return "redirect:/boards/{id}/edit";
    }
}

/**@PostMapping(value = "/{id}/edit", consumes = MediaType.APPLICATION_JSON_VALUE)
public String updateBoard_1(@PathVariable("id") Long id, @RequestBody BoardRequestDTO boardRequestDTO) {
    Board board = boardMapper.boardRequestDTOToBoard(boardRequestDTO).toBuilder().id(id).build();
    boardService.updateBoard(board);
    return "redirect:/boards";
}**/
```

굳이 @RequestBody 파라미터로 받는 메소드를 하나 더 만든 이유?

MediaType.APPLICATION_JSON_VALUE !!

Post 도메인 제작 과정 - Repository, Service, Controller

```
10 @Repository
11 public interface PostRepository extends JpaRepository<Post, Long> {
12     Page<Post> findAllByBoardOrderByTitleAsc(Board board, Pageable pageable);
13     Page<Post> findAllByBoardAndTitleContaining(Board board, String keyword, Pageable pageable);
14 }
15
```

```
Wambatcode
public Post findPost(Long postId) {
    return postRepository.findById(postId)
        .orElseThrow(() -> new Exception(ExceptionEnum.POST_NOT_FOUND));
}
```

```
Wambatcode
public Post createPost(Post post, Long boardId, MultipartFile file, String userId) throws IOException {
    User user = userService.findUserById(userId);
    Board boardToCreate = boardService.findBoardById(boardId);
    post.setBoard(boardToCreate);
    post = postFileService.FileUpload(file, post);
    post.setUser(user);

    return postRepository.save(post);
}
```

```
Wambatcode
public Post updatePost(Post post, Long postId, MultipartFile file, String userId) throws IOException {
    User user = userService.findUserById(userId);
    post.setId(postId);
    Post foundPost = postRepository.findById(post.getId())
        .orElseThrow(() -> new Exception(ExceptionEnum.POST_NOT_FOUND));

    Optional.ofNullable(post.getTitle())
        .ifPresent(foundPost::setTitle);
    Optional.ofNullable(post.getContent())
        .ifPresent(foundPost::setContent);

    foundPost = postFileService.FileUpload(file, foundPost);
    foundPost.setUser(user);

    return postRepository.save(foundPost);
}
```

```
Wambatcode
public void deletePost(Long id) {
    Post foundPost = postRepository.findById(id)
        .orElseThrow(() -> new Exception(ExceptionEnum.POST_NOT_FOUND));
    postRepository.delete(foundPost);
}
```

```
@GetMapping("/{postId}")
public String getPostDetail(@PathVariable Long postId, Model model) {
    Post post = postService.findPost(postId);
    model.addAttribute("post", post);
    List<Comment> comments = commentService.findCommentsByPostId(postId);
    model.addAttribute("comments", comments);
    return "post/post";
}
```

```
Wambatcode
@GetMapping("/{create}")
public String createPost(@RequestParam Long boardId, Model model) {
    model.addAttribute("boardId", boardId);
    return "post/createPost";
}
```

```
Wambatcode
@PostMapping(value = "{create}")
public String createPost(@ModelAttribute PostRequestDTO postRequestDTO, @RequestParam Long boardId, MultipartFile file, String userId) throws IOException {
    Post post = postMapper.postRequestDTOToPost(postRequestDTO);
    Post createdPost = postService.createPost(post, boardId, file, userId);

    return "redirect:/boards/" + createdPost.getBoard().getId();
}
```

```
/**
 * @PostMapping(value = "/create", consumes = MediaType.APPLICATION_JSON_VALUE)
 * public String createPostPost_1(@RequestBody PostRequestDTO postRequestDTO, @RequestParam Long boardId) {
 *     Post post = postMapper.postRequestDTOToPost(postRequestDTO);
 *     Post createdPost = postService.createPost(post, boardId);
 *     return "redirect:/boards/" + createdPost.getBoard().getId();
 * }
 */
```

```
Wambatcode
@GetMapping("/{postId}/edit")
public String editPost(@PathVariable Long postId, Model model) {
    Post post = postService.findPost(postId);
    PostResponseDTO postResponseDTO = postMapper.postToPostResponseDTO(post);
    model.addAttribute("postResponseDTO", postResponseDTO);
    return "post/editPost";
}
```

Comment 도메인 제작 과정 - Repository, Service, Controller

```
Wambatcode
@Repository
public interface CommentRepository extends JpaRepository<Comment, Long> {
    Wambatcode
    List<Comment> findCommentsByPostId(Long postId);
}
```

```
Wambatcode
public void saveComment(Long postId, Comment comment, String userId){
    User user = userService.findUserById(userId);
    Post post = postRepository.findById(postId).orElseThrow(
        () -> new Exception(ExceptionEnum.POST_NOT_FOUND));
    comment.setPost(post);
    comment.setUser(user);
    commentRepository.save(comment);
}
```

```
Wambatcode
public Long findPostById(Long id){ Complexity is 3 Everything is cool
    Comment comment = commentRepository.findById(id).orElseThrow(() -> new Exception(ExceptionEnum.COMMENT_NOT_FOUND));
    return comment.getPost().getId();
}
```

```
Wambatcode
public List<Comment> findCommentsByPostId(Long postId) { return commentRepository.findCommentsByPostId(postId); }
```

```
Wambatcode
public Comment updateComment(Long commentId, Comment comment, String userId){ Complexity is 4 Everything is cool
    User user = userService.findUserById(userId);
    Comment lastComment = commentRepository.findById(commentId)
        .orElseThrow(() -> new Exception(ExceptionEnum.COMMENT_NOT_FOUND));
    Optional.ofNullable(comment.getContent())
        .ifPresent(lastComment::setContent);
    lastComment.setUser(user);

    return commentRepository.save(lastComment);
}
```

```
Wambatcode
public void deleteComment(Long commentId){
    Comment comment = commentRepository.findById(commentId)
        .orElseThrow(() -> new Exception(ExceptionEnum.COMMENT_NOT_FOUND));
    commentRepository.delete(comment);
}
```

```
@PostMapping
public String saveComment(@ModelAttribute CommentDTO commentDTO, @RequestParam Long postId, Complexity is 3 Everything is cool
    String userId, RedirectAttributes redirectAttributes){
    Comment comment = commentMapper.commentDTOToComment(commentDTO);
    if(userService.validation(userId)){
        commentService.saveComment(postId, comment, userId);
    }
    redirectAttributes.addAttribute("postId", postId);
    return "redirect:/posts/{postId}";
}
```

```
/**@PostMapping(consumes = MediaType.APPLICATION_JSON_VALUE)
public String saveComment_1(@RequestBody CommentDTO commentDTO, @RequestParam Long postId, RedirectAttributes redirectAttributes){
    Comment comment = commentMapper.commentDTOToComment(commentDTO);
    commentService.saveComment(postId, comment);
    redirectAttributes.addAttribute("postId", postId);
    return "redirect:/posts/{postId}";
}*/
```

```
Wambatcode
@PostMapping("/{commentId}/edit")
public String updateComment(@PathVariable Long commentId, @ModelAttribute CommentDTO commentDTO, Complexity is 3 Everything is cool
    String userId, String password, RedirectAttributes redirectAttributes){
    Comment comment = commentMapper.commentDTOToComment(commentDTO);
    if(userService.login(userId, password) != null){
        Comment updateComment = commentService.updateComment(commentId, comment, userId);
        redirectAttributes.addAttribute("postId", updateComment.getPost().getId());
    }else{
        redirectAttributes.addAttribute("postId", commentService.findPostById(commentId));
    }
    return "redirect:/posts/{postId}";
}
```

```
Wambatcode
@DeleteMapping("/{commentId}")
public String deleteComment(@PathVariable Long commentId){
    commentService.deleteComment(commentId);
    return "redirect:/posts";
}
```

웹페이지 구성(메인 화면)

localhost:8080/boards



음식 종류 카테고리

카테고리 추가



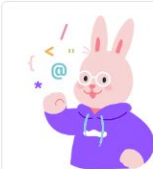
음료수

시원한 음료수~~



덮밥

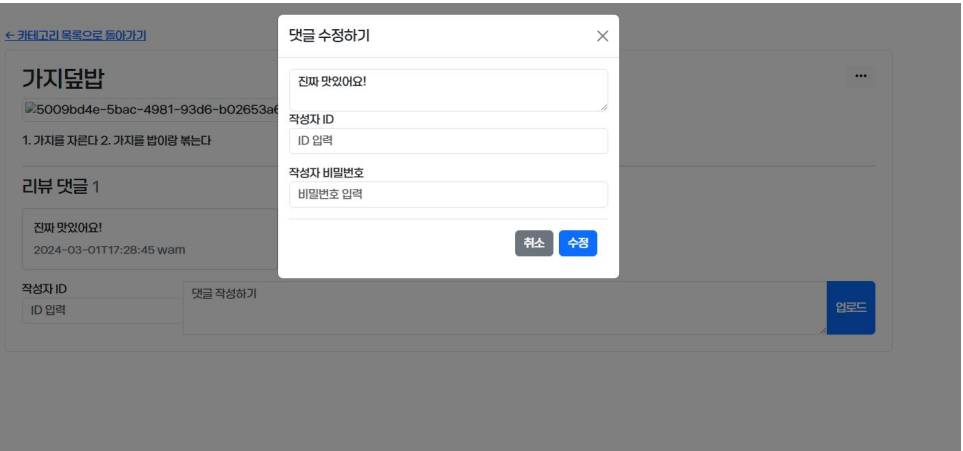
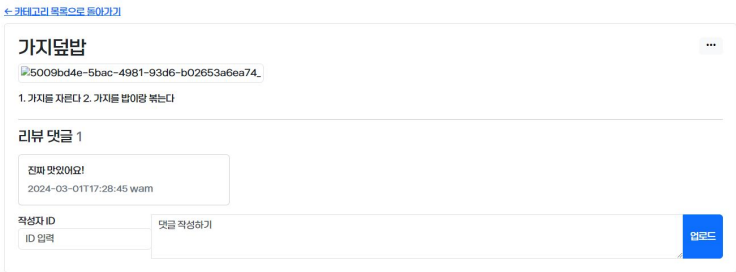
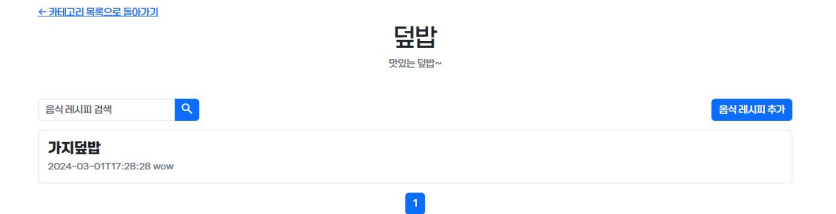
맛있는 덮밥~



디저트

맛있는 디저트~~

웹 페이지 구성(게시물/댓글)



User 도메인 제작 과정 - Repository, Service, Controller

```
7 @Repository
8 public interface UserRepository extends JpaRepository<User, Long> {
9     Wambatcode
10     User findById(String userId);
11     Wambatcode
12     boolean existsByUserId(String userId);
13 }
```

```
> Wambatcode
public boolean validation(String userId) { return userRepository.existsByUserId(userId); }

Wambatcode
public String findPassword(String userId){
    User user = userRepository.findById(userId);
    return user.getPassword();
}

Wambatcode
> public User findUserByUserId(String userId) { return userRepository.findById(userId); }

Wambatcode
public void signup(User user) {
    user.setUserId(user.getUserId());
    user.setPassword(user.getPassword());
    userRepository.save(user);
}

Wambatcode
public User login(String userId, String password){ Complexity is 6 It's time to do something...
    User user = userRepository.findById(userId);
    if(user != null && user.getPassword().equals(password)){
        return user;
    }
    return null;
}
```

```
@Controller
public class UserController { Complexity is 6 It's time to do something...
    private final UserService userService;

    Wambatcode
    public UserController(UserService userService) { this.userService = userService; }

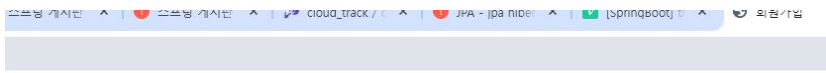
    Wambatcode
    @GetMapping("/{signup}")
    public String showSignupForm() { return "user/signup"; }

    Wambatcode
    @PostMapping("/{signup}")
    public String signup(@RequestParam("userId") String userId, @RequestParam("password") String password)
    {
        if(userService.validation(userId)){
            return "user/signup";
        }else {
            User user = new User(userId, password);
            userService.signup(user);
            return "redirect:/login";
        }
    }

    Wambatcode
    @GetMapping("/{login}")
    public String showLoginForm(Model model) {
        model.addAttribute("user", new User());
        return "user/login";
    }

    Wambatcode
    @PostMapping("/{login}")
    public String login(@ModelAttribute("user") User user, Model model) { Complexity is 5 Everything is cool!
        User loginUser = userService.login(user.getUserId(), user.getPassword());
        if (loginUser != null) {
            model.addAttribute("userId", user.getUserId());
            return "redirect:/boards";
        } else {
            return "user/login";
        }
    }
}
```

CRUD와 추가 기능 - 회원 가입 / 로그인



회원가입

사용자 ID

비밀번호

가입하기



로그인

사용자 ID

비밀번호

로그인

- 회원 가입 시 중복 사용자ID에 대한 검사는 **UserService**의 **validation**메소드를 통해 검사.
만약 중복 사용자ID가 존재할 경우 로그인 페이지로 넘어가지 않음.
- 로그인 시 잘못된 사용자ID 혹은 비밀번호를 입력했을 경우,
UserService내 **login**메소드의 제어에 따라 다음 페이지로 넘어가지 않음.

CRUD외 추가 기능 - 수정 / 삭제 시 작성자 정보 확인

카테고리 수정

카테고리 이름

음료수

카테고리 설명

시원한 음료수

이미지 업로드

파일 선택

선택된 파일 없음

작성자 ID

ID 입력

작성자 비밀번호

비밀번호 입력

게시판 수정

취소

- 게시판, 게시물, 댓글 모두 작성자만이 수정 / 삭제할 수 있음.
- 작성자 ID, 비밀번호 중 하나라도 일치 하지 않을 경우 수정 / 삭제 불가.

CRUD와 추가 기능 - 파일 업로드

```
@Service
@Transactional
public class PostFileService { Complexity is 5 Everything is cool!

    public Post FileUpload(MultipartFile file, Post post) throws IOException { Complexity is 4 Everything is cool!
        String projectPath = System.getProperty("user.dir") + "\\src\\main\\resources\\static\\files";

        UUID uuid = UUID.randomUUID();
        String fileName = uuid + "_" + file.getOriginalFilename();

        File saveFile = new File(projectPath, fileName);
        file.transferTo(saveFile);
        post.setFilename(fileName);
        post.setFilepath("/files/" + fileName);

        return post;
    }
}
```

음식 종류 카테고리



- 게시판이나 게시물에 이미지 파일을 함께 업로드하기 위한 **FileService**.
- 업로드할 컴퓨터 상의 로컬 파일을 쉽게 다루기 위한 **MultipartFile** 인터페이스 사용.
- **UUID**(고유 식별자)를 이용하여 중복된 파일명과 파일경로를 엔티티에 저장하지 않도록 함.

Trouble Shooting

1. **Post API를 @ModelAttribute를 통해 받던 데이터를 @RequestBody의 JSON형태로 받아 실행되는지 테스트가 해보고 싶었음(Postman).**
-> 앞서 언급했듯 **MediaType.APPLICATION_JSON_VALUE**를 통해 **JSON**형태의 데이터를 받을 수 있는 메소드 작성
2. **게시판과 게시물 생성 시, 이미지 파일을 첨부하는 로직을 생성 / 수정 두 메소드에 다 작성하여 가독성이 떨어짐.**
-> **Board**도메인은 **BoardFileService**클래스, **Post**도메인은 **PostFileService**클래스로 분리하여 가독성을 높임.

개선할 점

1. **Spring Security** 같은 프레임 워크를 사용하지 않고 만든 보안이 취약한 간단한 로그인
-> **Spring Security**에 대한 지식 숙지 필요성을 느낌.
2. 내가 작성한 **Java**코드를 **Thymeleaf**를 통해 **HTML**과 매핑할 때 너무 많은 오류 발생, 많은 시간 소모
-> **Thymeleaf**에 좀 더 익숙해져야 함을 느낌, 내가 직접 만든 **HTML/CSS/JS**를 통해 좀 더 세련된 웹페이지를 만들어보고 싶음.
3. 반쪽 짜리 이미지 파일 업로드 기능(실시간 반영이 안됨, 재빌드해야 이미지 파일이 보임)
-> 실시간으로 반영이 가능한 **Firebase** 데이터베이스 사용법 숙지 필요성을 느낌.

-> 전체적으로 아직 지식이 많이 부족함을 느낌..

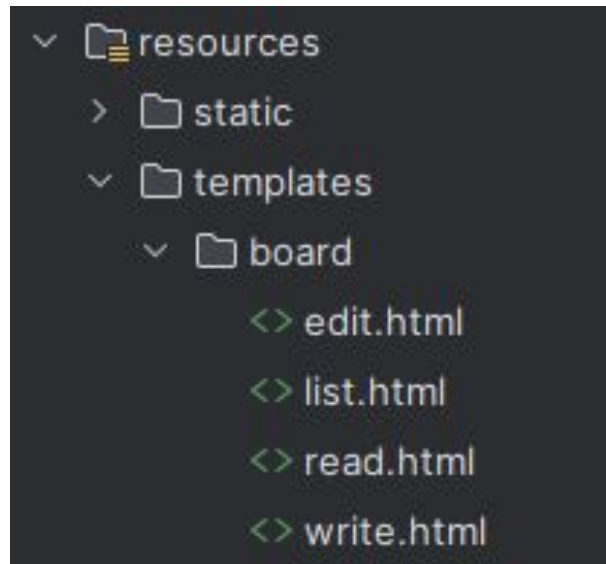
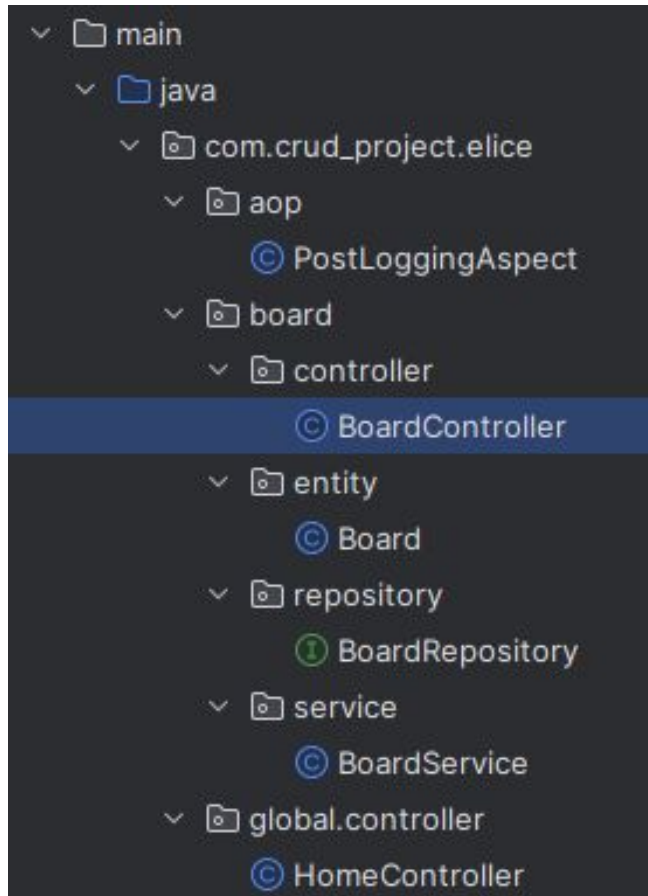
조웅진

화곡동 동네사람 게시판

코드 작성 과정

```
19 <dependencies>
20   <dependency>
21     <groupId>org.springframework.boot</groupId>
22     <artifactId>spring-boot-starter-data-jpa</artifactId>
23   </dependency>
24   <dependency>
25     <groupId>org.springframework.boot</groupId>
26     <artifactId>spring-boot-starter-thymeleaf</artifactId>
27   </dependency>
28   <dependency>
29     <groupId>org.springframework.boot</groupId>
30     <artifactId>spring-boot-starter-web</artifactId>
31   </dependency>
32
33   <dependency>
34     <groupId>org.springframework.boot</groupId>
35     <artifactId>spring-boot-devtools</artifactId>
36     <scope>runtime</scope>
37     <optional>true</optional>
38   </dependency>
```

코드 작성 과정



코드 작성 과정

```
@GetMapping("/write")
public String showWriteForm(Model model) {
    Board board = new Board();
    User user = new User();
    user.setId(user.getId());
    board.setUser(user);

    model.addAttribute(attributeName: "board", board);
    return "/board/write";
}
```

```
@PostMapping("/write")
public String write(Board board) {
    board.setBno(board.getBno());
    User user = new User();
    user.setId("운영자");
    board.setUser(user);
    board.setViewCnt((long)(Math.random()*20));
    board.setInDate(new Date());
    board.setUpDate(new Date());
    boardService.write(board);
    return "redirect:/board/list";
}
```

시연 영상



번호	제목	내용	작성일	조회수
1	공지사항 1	공지사항	2024-02-28	206
2	공지사항 2	공지사항	2024-02-28	166
3	공지사항 3	공지사항	2024-02-28	152
4	공지사항 4	공지사항	2024-02-28	121
5	공지사항 5	공지사항	2024-02-28	96
6	공지사항 6	공지사항	2024-02-28	87
7	공지사항 7	공지사항	2024-02-28	203
8	공지사항 8	공지사항	2024-02-28	122
9	공지사항 9	공지사항	2024-02-28	166
10	공지사항 10	공지사항	2024-02-28	174
11	공지사항 11	공지사항	2024-02-28	100
12	공지사항 12	공지사항	2024-02-28	100
13	공지사항 13	공지사항	2024-02-28	100
14	공지사항 14	공지사항	2024-02-28	82
15	공지사항 15	공지사항	2024-02-28	26
16	공지사항 16	공지사항	2024-02-28	100

개선할 점

- 예외 코드 처리하지 못한 점
- 회원가입, 게시판 카테고리, Comment 기능을 구현하지 못한 점
- Paging 처리하지 못한 점
- 게시물을 클릭하면 조회수가 1 증가하게 하지 못한 점

Thank you!

— 김민상 김영준 안수민 임준용 조용진 —
