

목차

1. 프로젝트 설명
 - a. 프로젝트 폴더 계층 구조
 - b. 사용 기술 및 언어
2. **ERD**
3. 게시판 및 댓글 관련 기능
 - a. 게시판 **CRUD**
 - b. 답변형 게시판 **CRUD**
 - c. 게시글 검색
 - d. 대댓글(중첩 페이징)
4. 유저 관련 기능
 - a. 내가 쓴 글 확인
5. 관리자 관련 기능
 - a. 글 및 회원관리
6. 문제점 및 개선점 및 고려점 (**option**)
 - a. **Community proj Improvements** 폴더 참고

1-a. 프로젝트 폴더 계층구조(MVC 모델)

*HealthChain->src->main에 가보면 **java**폴더와 **webapp**폴더가 있는데 이 두 폴더 기준으로 설명함. 아래 구조에서 설명안된 것들은 설정관련 파일들 이라 생각하시면 됩니다.

- **java**
 - **api** : api 관련 기능.
 - **controller** : MVC 중 Controller.
 - **model** : MVC 중 Model(DTO).
 - **dao** : DAO
 - **service** : MVC 중 Model의 service 레이어.
 - **xml files** : Mybatis SQL 코드들.
- **webapp**
 - **index.jsp** : 프로그램 진입점.
 - **view** : MVC 중 View.

1-b. 사용기술 및 언어

- **기술** : JSP, Servlet, ORACLE, Mybatis, AJAX
- **언어** : JS, JAVA, SQL

2. ERD

ERD 주소 : <https://www.erdcloud.com/d/r6jwhxYsMndAXySjf>


설명 : 의사면허 테이블 같은 경우에는 단순히 면허코드를 체크하기 위해 만든 테이블입니다. **ERD**에서 이와 같은 테이블을 만들면 안되지만 실제 면허코드를 다룰 수는 없으니 상황을 가정해서 만듦.

3-a. 게시판 CRUD

776	frbNotice1	관리자	2025-01-23	12
807	게시글작성	nickName1	2025-02-13	97
806	게게시	nickName1	2025-02-13	1
805	tjhahsgshg	nickName1	2025-02-12	4
804	sdfhdsfhs	nickName1	2025-02-12	0
802	fhadfhadf	nickName1	2025-02-12	1

- **create** : 일반적인 게시글이랑 동일. 관리자는 글쓰면 공지사항이 되게함(관리자글을 **SQL**에서 **UNION ALL**을 통해 위쪽으로 배치 나머지는 아래로 배치해 구현. **PK**는 중복안됨).
- **read** : 일반적인 **read**랑 동일.
- **update** : 일반적인 **update**랑 동일.
- **delete** : 일반적인 **delete**랑 동일.

3-b. 답변형 게시판 **CRUD**

3497	df	nickName1	2025-02-12	1
3495	게시글	nickName1	2025-02-12	0
3493	ssssssssssssssssssssss	nickName1	2025-02-12	16
 3494	dddddddddddddddddd	관리자	2025-02-12	2
3471	csTitle234	nickName1	2025-02-04	7
 3490	csTitle234-2	관리자	2025-02-06	2
 3472	csTitle234-1	관리자	2025-02-04	3
3470	csTitle233	nickName1	2025-02-04	0

- **create** : 위와 동일.
- **read** : 위와 동일.
- **update** : 위와 동일.
- **delete** : 답변형 게시판 구조 참고바람.

delete 프로세스 설명 : 답변형 게시판의 **delete** 프로세스는 일반형 게시판의 **delete** 프로세스와 다릅니다. 아래의 표를 한번 생각해 봅시다.

게시글 1
게시글 1-1 (게시글 1에 대한 답글)
게시글 1-1-1 (게시글 1-1에 대한 답글)

게시글 1을 삭제했다고 생각해봅시다. 그럼 다음과 같이 됩니다.

삭제된 게시글 입니다.
게시글 1-1 (게시글 1 에 대한 답글)
게시글 1-1-1 (게시글 1-1 에 대한 답글)

여기서 삭제된 게시글 이라는 게시글은 표시해도 되고 표시하지 않아도 됩니다.

그리고 다음과 같은 상황도 한번 생각해 봅시다.

어떤 폴더가 있는데, 이 폴더안에 여러파일들을 넣었습니다. 그리고 이 파일이 담겨진 폴더를 삭제를 하면 안에있는 파일들도 같이 삭제가 됩니다. 두 가지 상황을 예로 한번 들어봤는데 먼저 첫번째 상황에 대해 생각을 해봅시다.

- 삭제된 게시글
 - 표시한다 (가정 **A**)
 - 표시안한다 (가정 **B**)

가정 **A**일때, 사용자 입장에서 보면 표시하는게 맞을까요? 삭제된 게시물은 사용자 입장에서보면 확인 할수 없는 게시글일 뿐더러 불필요한 데이터 연동비용만 증가하게 되는 데이터라 생각이 됩니다. 실제로 이렇게 하는 웹페이지도 있지만 저는 장기적으로 볼때는 그다지 좋다고 생각하지는 않습니다.

가정 **B**일때, 그럼 게시글 **1-1**, 게시글 **1-1-1**은 어떻게 처리를 해야 할까요? 그리고 두 번째 상황에 대하여 사용자의 행위는 폴더를 삭제했는데 결과는 폴더와 파일들이 같이 삭제된 상황입니다. 그럼 이와 같은 현상들이 발생하는 근본적인 원인이 뭘까요? 바로 답변형 게시판 구조의 종속성 문제 즉, 트리 구조의 문제점에서 기인을 합니다.

트리구조는 **CRUD**에 대하여 **R**의 경우에는 빠른 조회가 가능할 뿐만 아니라 탐색과정에서 별다른 프로세스를 요구하지 않습니다. 하지만, **C**, **D**, **U**같은 경우에는 별도의 프로세스를 요구를 합니다. 바로 노드 재배치 프로세스를 요구합니다.

이러한 재배치 프로세스에 대해 설명을 하자면, 트리는 모든 노드가 연결되어 있습니다. 어찌보면 체이닝이 되어있다고 봐도되죠. **D**를 예시로 설명하자면, **DELETE** 연산은 노드의 삭제를 의미하고 노드간의 체이닝이 끊어짐을 의미합니다.

그리고 이 체이닝이 끊어짐에 따라 트리가 여러개로 나뉘게 됩니다. 그리고, 트리를 다시 원래의 **1**개로 만들어야하기 때문에 서로 나뉘어진 트리를 병합하는데 이 프로세스가 노드 재배치 프로세스 입니다.

그럼 가정 **B**와 같은 상황일 때 어떻게 해야 할까요? 분리된 게시판 트리를 어디에 배치해 두어야 할까요? 배치할 마땅한 장소가 있을까요? 저는 없다 생각했습니다. 그래서 삭제하는게 맞다 판단을 했습니다. 그리고 우리는 모두 이러한 시스템에 대해 이미 알고 있고 암묵적으로 동의한 상태라 생각했습니다.

두 번째 파일 상황에 대해서 여러분이 윈도우 컴퓨터에서 회사의 중요한 데이터 폴더를 삭제를 했습니다. 그럼 누가 책임을 지게 될까요? 이와 같은 시스템을 설계한 마이크로소프트 잘못인가요 아님 사용자 잘못인가요?

단점 : 단점이 있는데, 복구가 어렵습니다. 간접종속적인 데이터까지 삭제를 했으니 삭제될 당시의 상태를 파악하기 어렵죠.

단점극복점 : 삭제될 당시의 상태를 기록하면 된다 생각이 듭니다. 테이블쪽에 컬럼을 몇개 추가해 해당 트리에 대한 상태를 기록하는 것이죠.

3-c. 게시물 검색

"1" 으로 검색하신 결과

게시글아이디	제목	작성자	작성일자	조회수
805	tjahsgshg	nickName1	2025-02-12	4
785	forTestTitle1	nickName2	2025-01-29	442
781	gk1	nickName1	2025-01-26	13
778	1	nickName3	2025-01-26	1
767	frbTitle231	nickName1	2025-01-18	0
757	frbTitle221	nickName1	2025-01-18	0
755	frbTitle219	nickName1	2025-01-18	0
754	frbTitle218	nickName1	2025-01-18	0

설명 : ERD를 보시면 아시겠지만, 테이블이 서로 나뉘어져 있습니다. 페이징을 여태 한 테이블을 기준으로 해왔는데 전체 테이블을 대상으로 한다면 상황이 막막하기만 하지만 여러 테이블을 합쳐 한 개의 테이블로 다루면 페이징이 되지 않을까 생각을 했습니다.

하나의 테이블로 다루는 **SQL**문을 저는 **UNION ALL** 연산을 이용했습니다. **UNION ALL** 은 중복허용 및 공통컬럼필요 라는 특징을 가지고 있는데, 각 게시판 테이블은 구조가 비슷하니 공통컬럼만 가져왔고, 중복허용에 대한 설명은 다음과 같습니다.

각 게시판 **PK**가 중복이 되는 경우입니다. 각 테이블에 카테고리라는 컬럼을 추가해 해결했습니다. 서버는 클라이언트로부터 **PK**와 카테고리를 받고 카테고리를 통해 어느 테이블인지, **PK**를 통해 테이블의 어느 데이터인지를 식별할 수 있습니다.

3-d. 대댓글(중첩페이징)

설명 : 페이지안에 페이지가 존재하는 구조입니다. 일반적인 페이지는 부모**PK**가 필요가 없으나 중첩 페이지는 부모**PK**등이 필수로 필요합니다.

왜냐하면 자식 데이터가 부모 데이터 안에 포함되기 때문입니다.

N(>=2)중 페이지를 하려면 **N-1**개의 부모 컬럼이 필요합니다.

구현 : 서버랑 **DB**쪽은 패러미터 받은걸로 처리하면됨. 클라이언트에서 데이터를 받아서 동적으로 보여줘야 하기 때문에 메인구현은 클라이언트가 담당. 후에 설명하겠지만 데이터 동기화 필수. **qsBoardContent.jsp**에 코드 구현되어있음.

4-a. 내가 쓴 글 확인

마이페이지

회원정보 수정

내가 쓴 게시물

내가 쓴 댓글

탈퇴

내가 쓴 게시물

전체

자유게시판

게시글작성

2025-02-13

자유게시판

게게시

2025-02-13

자유게시판

tjhahsgshgl

2025-02-12

자유게시판

sdfhdsfhsj

2025-02-12

자유게시판

fhadfhadff

2025-02-12

자유게시판

agasfhfasdfj

2025-02-12

자유게시판

adgadj

2025-02-12

자유게시판

asdasdddddddddddddd

2025-02-12

자유게시판

gk1j

2025-01-26

자유게시판

하j

2025-01-26

자유게시판

안녕하세요j

2025-01-21

자유게시판

frbTitle234j

2025-01-18

자유게시판

frbTitle233j

2025-01-18

자유게시판

frbTitle232j

2025-01-18

자유게시판

frbTitle231j

2025-01-18

페이지 이동 :

이동

1

2

3

4

5

6

7

8

9

10

다음

설명 : 내가 쓴 게시물이나 댓글 확인가능.

구현 : 요청이 마이페이지에서 왔는지 아니면 일반게시판에서 왔는지에 대한 판별 코드가 필요함. 그리고, 비동기 요청이므로 데이터 동기화가 안되면 동기화 코드를 따로 삽입해야함. 이런 코드 삽입은 매우 좋지 않다고 생각.

페이징 처리는 페이지로 범위를 가져오는 연산의 반대인 범위로 페이지를 가져오는 역연산이 필요함. 게시판 검색에도 이 기능이 적용됨.

5-a. 글 및 회원관리

관리자 페이지

회원목록
공지사항 관리
고객센터 답변관리
회원목록
전체 ▼
userid9 nickName9 일반회원 세부정보 정지상태아님 탈퇴
userid8 nickName8 의사회원 세부정보 정지상태아님 탈퇴
userid7 nickName7 일반회원 세부정보 정지상태아님 탈퇴
userid6 nickName6 의사회원 세부정보 정지상태아님 탈퇴
userid5 nickName5 일반회원 세부정보 정지상태아님 탈퇴
userid4 nickName4 의사회원 세부정보 정지상태아님 탈퇴
userid3 nickName3 일반회원 세부정보 정지상태아님 탈퇴
userid2 nickName2 의사회원 세부정보 정지상태아님 탈퇴
userid1 nickName1 일반회원 세부정보 정지상태아님 탈퇴
페이지 이동 : <input type="text"/> 이동 1