signup.js - 이메일 정규식

| **const** regExp = /^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/; |
| --- |

이메일, 인증키 저장 테이블 생성 구문(dbeaver)

| CREATE TABLE "TB\_AUTH\_KEY"(  "KEY\_NO" NUMBER PRIMARY KEY,  "EMAIL" NVARCHAR2(50) NOT NULL,  "AUTH\_KEY" CHAR(6) NOT NULL,  "CREATE\_TIME" DATE DEFAULT SYSDATE NOT NULL  );  COMMENT ON COLUMN "TB\_AUTH\_KEY"."KEY\_NO" IS '인증키 구분 번호(시퀀스)';  COMMENT ON COLUMN "TB\_AUTH\_KEY"."EMAIL" IS '인증 이메일';  COMMENT ON COLUMN "TB\_AUTH\_KEY"."AUTH\_KEY" IS '인증 번호';  COMMENT ON COLUMN "TB\_AUTH\_KEY"."CREATE\_TIME" IS '인증 번호 생성 시간';  CREATE SEQUENCE SEQ\_KEY\_NO NOCACHE; -- 인증키 구분 번호 시퀀스  SELECT \* FROM "TB\_AUTH\_KEY"; |
| --- |

비밀번호 정규식

| **const** regExp = /^[a-zA-Z0-9!@#\_-]{6,20}$/; |
| --- |

닉네임 정규식

| **const** regExp = /^[가-힣\w\d]{2,10}$/; |
| --- |

전화번호 정규식

| **const** regExp = /^01[0-9]{1}[0-9]{3,4}[0-9]{4}$/; |
| --- |

EmailServiceImpl - 인증번호 생성 메서드

| /\*\* 인증번호 생성 (영어 대문자 + 소문자 + 숫자 6자리)  \* **@return** authKey  \*/  **public** String createAuthKey() {  String key = "";  **for**(**int** i=0 ; i< 6 ; i++) {    **int** sel1 = (**int**)(Math.*random*() \* 3); // 0:숫자 / 1,2:영어    **if**(sel1 == 0) {    **int** num = (**int**)(Math.*random*() \* 10); // 0~9  key += num;    }**else** {    **char** ch = (**char**)(Math.*random*() \* 26 + 65); // A~Z    **int** sel2 = (**int**)(Math.*random*() \* 2); // 0:소문자 / 1:대문자    **if**(sel2 == 0) {  ch = (**char**)(ch + ('a' - 'A')); // 대문자로 변경  }    key += ch;  }    }  **return** key;  } |
| --- |

EmailConfig.java

| Properties prop = new Properties();  prop.setProperty("mail.transport.protocol", "smtp");  prop.setProperty("mail.smtp.auth", "true");  prop.setProperty("mail.smtp.starttls.enable", "true");  prop.setProperty("mail.debug", "true");  prop.setProperty("mail.smtp.ssl.trust","smtp.gmail.com");  prop.setProperty("mail.smtp.ssl.protocols","TLSv1.2");      mailSender.setUsername(userName);  mailSender.setPassword(password);  mailSender.setHost("smtp.gmail.com");  mailSender.setPort(587);  mailSender.setDefaultEncoding("UTF-8");  mailSender.setJavaMailProperties(prop); |
| --- |

DB 추가 테이블

| -- 파일 업로드 테스트용 테이블  **CREATE** **TABLE** **"UPLOAD\_FILE"**(  FILE\_NO **NUMBER** **PRIMARY** **KEY**,  FILE\_PATH **VARCHAR2**(500) **NOT** **NULL**,  FILE\_ORIGINAL\_NAME **VARCHAR2**(300) **NOT** **NULL**,  FILE\_RENAME **VARCHAR2**(100) **NOT** **NULL**,  FILE\_UPLOAD\_DATE **DATE** **DEFAULT** **SYSDATE**,  MEMBER\_NO **NUMBER** **REFERENCES** **"MEMBER"**  );  **COMMENT** **ON** **COLUMN** **"UPLOAD\_FILE"**.FILE\_NO **IS** **'파일 번호(PK)'**;  **COMMENT** **ON** **COLUMN** **"UPLOAD\_FILE"**.FILE\_PATH **IS** **'클라이언트 요청 경로'**;  **COMMENT** **ON** **COLUMN** **"UPLOAD\_FILE"**.FILE\_ORIGINAL\_NAME **IS** **'파일 원본명'**;  **COMMENT** **ON** **COLUMN** **"UPLOAD\_FILE"**.FILE\_RENAME **IS** **'변경된 파일'**;  **COMMENT** **ON** **COLUMN** **"UPLOAD\_FILE"**.FILE\_UPLOAD\_DATE **IS** **'업로드 날짜'**;  **COMMENT** **ON** **COLUMN** **"UPLOAD\_FILE"**.MEMBER\_NO **IS** **'MEMBER 테이블의 PK(MEMBER\_NO) 참조'**;  **CREATE** **SEQUENCE** SEQ\_FILE\_NO NOCACHE;  **SELECT** \* **FROM** ***"UPLOAD\_FILE"***; |
| --- |

myPage.js - 프로필사진 업데이트

| // -------------------------------------------------------  /\* 프로필 이미지 추가/변경/삭제 \*/  // 프로필 이미지 페이지 form 태그  **const** profile = document.querySelector("#profile");  // 프로필 이미지가 새로 업로드 되거나 삭제 되었음을 기록하는  // 상태 변수  // -1 : 초기 상태(변화 없음)  // 0 : 프로필 이미지 삭제  // 1 : 새 이미지 선택  **let** statusCheck = -1;  // input type="file" 태그의 값이 변경 되었을 때  // 변경된 상태를 백업해서 저장할 변수  // -> 파일이 선택/취소된 input을 복제해서 저장  // 요소. cloneNode(true|false) : 요소 복제(true 작성 시 하위 요소도 복제)  **let** backupInput;  // profile form태그가 화면에 있다면  if(profile != null){  // 1) 프로필 이미지 수정에 사용할 요소 모두 얻어오기  // img 태그 (프로필 이미지가 보여지는 요소)  **const** profileImg = document.querySelector("#profileImg");  // input type="file" 태그 (실제 업로드할 프로필 이미지를 선택하는 요소)  **let** imageInput = document.querySelector("#imageInput");  // x버튼 (프로필 이미지를 제거하고 기본 이미지로 변경하는 요소)  **const** deleteImage = document.querySelector("#deleteImage");  // 3) changeImageFn 함수 정의하기  /\* input type="file"의 값이 변했을 때 동작할 함수(이벤트 핸들러) \*/  **const** changeImageFn = e **=>** {  // 업로드 가능한 파일 최대 크기 지정하여 필터링  **const** maxSize = 1024 \* 1024 \* 5;  // 5MB == 1024KB \* 5 == 1024B \* 1024 \* 5  console.log("e.target", e.target); // input 태그  console.log("e.target.value", e.target.value); // 변경된 값(파일명)  // 선택된 파일에 대한 정보가 담긴 배열 반환  // -> 왜 배열?? multiple 옵션에 대한 대비(파일 여러개 받을 때)  console.log("e.target.files", e.target.files);  // 업로드된 파일이 1개 있으면 files[0]에 저장됨  // 업로드된 파일이 없으면 files[0] == undefined  console.log("e.target.files[0]", e.target.files[0]);  **const** file = e.target.files[0];  // ------------ 업로드된 파일이 없다면(취소한 경우)------------  if(file == undefined){  console.log("파일 선택 후 취소됨");    // 파일 선택 후 취소 -> value == ''  // -> 선택한 파일 없음으로 기록됨  // -> backupInput으로 교체 시켜서  // 이전 이미지가 남아 있는 것 처럼 보이게 함  // 백업의 백업본  **const** temp = backupInput.cloneNode(true);    console.log("temp", temp); // 백업용 input태그  // input 요소 다음에 백업 요소 추가  imageInput.after(backupInput);    // 화면에 존재하는 기존 input 제거  imageInput.remove();  // imageInput 변수에 백업을 대입해서 대신하도록 함  imageInput = backupInput;  // 화면에 추가된 백업본에는  // 이벤트 리스너가 존재하지 않기 때문에 추가  imageInput.addEventListener("change", changeImageFn);  // 한번 화면에 추가된 요소(backupInput)는 재사용 불가능  // backupInput의 백업본이 temp를 backupInput 으로 변경  backupInput = temp;  return; // 다른 코드 수행할필요없이 바로 return  }      // ----------- 선택된 파일이 최대 크기를 초과한 경우 ------------  if(file.size > maxSize){  alert("5MB 이하의 이미지 파일을 선택해 주세요.");  //파일을 선택할 때 5MB보다 큰 파일을 선택하면  //일단 무조건 선택은 됨.  //근데 우리는 5MB보다 큰 파일은 취급 안하고 싶음  //그래서 대입된 5MB 초과한 파일을 없애버리겠다    // 아직 변경된적없는 초기상태에서 5MB 초과하는 이미지를 선택한 경우  if(statusCheck == -1){  imageInput.value = '';    } else{ // 기존에 변경하려고 선택한 이미지가 있는데  // 다음에 선택한 이미지가 최대 크기를 초과한 경우  // -> 비워버리면 안되고, 그 전에 선택한 이미지로 계속 보이게끔 처리해야함.  // 백업의 백업본  **const** temp = backupInput.cloneNode(true);  // input 요소 다음에 백업 요소 추가  imageInput.after(backupInput);  // 화면에 존재하는 기존 input 제거  imageInput.remove();  // imageInput 변수에 백업을 대입해서 대신하도록 함  imageInput = backupInput;  // 화면에 추가된 백업본에는  // 이벤트 리스너가 존재하지 않기 때문에 추가  imageInput.addEventListener("change", changeImageFn);  // 한번 화면에 추가된 요소(backupInput)는 재사용 불가능  // backupInput의 백업본이 temp를 backupInput 으로 변경  backupInput = temp;  }  return; // 다른 코드 수행할필요없이 바로 return  }    // ------------- 선택된 이미지 미리보기 ----------------  // JS에서 파일을 읽을 때 사용하는 객체  // - 파일을 읽고 클라이언트 컴퓨터에 저장할 수 있음  /\*FileReader 객체는 웹 애플리케이션에서 비동기적으로 파일의 내용을 읽을 수 있게 해줍니다. \*/  **const** reader = new FileReader();  // 선택한 파일(file) 을 읽어와  // BASE64 인코딩 형태로 읽어와 result 변수에 저장  reader.readAsDataURL(file); // -> 읽어오기 이벤트(load)  // readAsDataURL() : 파일을 BASE64 형식의 데이터 URL로 읽어들입니다.    // console.log("reader:",reader);  // result에 "" 이런식으로 들어감  // 읽어오기 끝났을 때 (파일 읽기 작업이 완료되면 이벤트 핸들러 함수를 실행)  reader.addEventListener("load", e **=>** {  // e.target == reader  // 읽어온 이미지 파일이 BASE64 형태로 반환됨  **const** url = e.target.result; // reader.result  // 프로필 이미지(img)에 src속성으로 url값 세팅  profileImg.setAttribute("src", url);    // 새 이미지 선택 상태를 기록  statusCheck = 1;  // 파일이 선택된 input을 복제해서 백업  backupInput = imageInput.cloneNode(true);  });  };    // 2) imageInput에 change 이벤트로 changeImageFn 등록  // change 이벤트 : 새로운 값이 기존 값과 다를 경우 발생  imageInput.addEventListener("change", changeImageFn);  // ------------ 4) x버튼 클릭 시 기본 이미지로 변경 ----------------  deleteImage.addEventListener("click", () **=>** {  // 프로필 이미지(img)를 기본 이미지로 변경  profileImg.src = "/images/user.png";  // input에 저장된 값(value)를 ''(빈칸)으로 변경  // -> input에 저장된 파일 정보가 모두 사라짐 == 데이터 삭제  imageInput.value = '';  backupInput = undefined; // 백업본도 삭제  // 삭제 상태임을 기록  statusCheck = 0;  });  // ------------ 5) #profile (form) 제출 시 -----------------  profile.addEventListener("submit", e **=>** {    **let** flag = true;  // loginMemberProfileImg : myPage-profile.html 하단에 script를 이용하여 타임리프로 선언해둔 변수    // submit 해도 되는 경우 :  // 1. 기존 프로필 이미지가 없다가 새 이미지가 선택된 경우  if(loginMemberProfileImg == null && statusCheck == 1) flag = false;  // 2. 기존 프로필 이미지가 있다가 삭제한 경우  if(loginMemberProfileImg != null && statusCheck == 0) flag = false;    // 3. 기존 프로필 이미지가 있다가 새 이미지가 선택된 경우  if(loginMemberProfileImg != null && statusCheck == 1) flag = false;  // 나머지의 경우는 기존 상태에서 변경사항이 없는 경우임.-> 제출막기  if(flag){ // flag 값이 true인 경우  e.preventDefault();  alert("이미지 변경 후 클릭하세요")  }  });  }  /\* [input type="file" 사용 시 유의 사항]  1. 파일 선택 후 취소를 누르면  선택한 파일이 사라진다 (value == '')  2. value로 대입할 수 있는 값은 '' (빈칸)만 가능하다  3. 선택된 파일 정보를 저장하는 속성은  value가 아니라 files이다  \*/ |
| --- |

| // -------------------------------------------------------  /\* 프로필 이미지 추가/변경/삭제 \*/  // 프로필 이미지 페이지 form 태그  **const** profile = document.querySelector("#profile");  // 프로필 이미지가 새로 업로드 되거나 삭제 되었음을 기록하는  // 상태 변수  // -1 : 초기 상태(변화 없음)  // 0 : 프로필 이미지 삭제  // 1 : 새 이미지 선택  **let** statusCheck = -1;  // input type="file" 태그의 값이 변경 되었을 때  // 변경된 상태를 백업해서 저장할 변수  // -> 파일이 선택/취소된 input을 복제해서 저장  // 요소. cloneNode(true|false) : 요소 복제(true 작성 시 하위 요소도 복제)  **let** backupInput;  // profile form태그가 화면에 있다면  if(profile != null){  // 1) 프로필 이미지 수정에 사용할 요소 모두 얻어오기  // img 태그 (프로필 이미지가 보여지는 요소)  **const** profileImg = document.querySelector("#profileImg");  // input type="file" 태그 (실제 업로드할 프로필 이미지를 선택하는 요소)  **let** imageInput = document.querySelector("#imageInput");  // x버튼 (프로필 이미지를 제거하고 기본 이미지로 변경하는 요소)  **const** deleteImage = document.querySelector("#deleteImage");  // 3) changeImageFn 함수 정의하기  /\* input type="file"의 값이 변했을 때 동작할 함수(이벤트 핸들러) \*/  **const** changeImageFn = e **=>** {  // 업로드 가능한 파일 최대 크기 지정하여 필터링  **const** maxSize = 1024 \* 1024 \* 5;  // 5MB == 1024KB \* 5 == 1024B \* 1024 \* 5  console.log("e.target", e.target); // input 태그  console.log("e.target.value", e.target.value); // 변경된 값(파일명)  // 선택된 파일에 대한 정보가 담긴 배열 반환  // -> 왜 배열?? multiple 옵션에 대한 대비(파일 여러개 받을 때)  console.log("e.target.files", e.target.files);  // 업로드된 파일이 1개 있으면 files[0]에 저장됨  // 업로드된 파일이 없으면 files[0] == undefined  console.log("e.target.files[0]", e.target.files[0]);  **const** file = e.target.files[0];  // ------------ 업로드된 파일이 없다면(취소한 경우)------------  if(file == undefined){  console.log("파일 선택 후 취소됨");    // 파일 선택 후 취소 -> value == ''  // -> 선택한 파일 없음으로 기록됨  // -> backupInput으로 교체 시켜서  // 이전 이미지가 남아 있는 것 처럼 보이게 함  // 백업의 백업본  **const** temp = backupInput.cloneNode(true);    console.log("temp", temp); // 백업용 input태그  // input 요소 다음에 백업 요소 추가  imageInput.after(backupInput);    // 화면에 존재하는 기존 input 제거  imageInput.remove();  // imageInput 변수에 백업을 대입해서 대신하도록 함  imageInput = backupInput;  // 화면에 추가된 백업본에는  // 이벤트 리스너가 존재하지 않기 때문에 추가  imageInput.addEventListener("change", changeImageFn);  // 한번 화면에 추가된 요소(backupInput)는 재사용 불가능  // backupInput의 백업본이 temp를 backupInput 으로 변경  backupInput = temp;  return; // 다른 코드 수행할필요없이 바로 return  }      // ----------- 선택된 파일이 최대 크기를 초과한 경우 ------------  if(file.size > maxSize){  alert("5MB 이하의 이미지 파일을 선택해 주세요.");  //파일을 선택할 때 5MB보다 큰 파일을 선택하면  //일단 무조건 선택은 됨.  //근데 우리는 5MB보다 큰 파일은 취급 안하고 싶음  //그래서 대입된 5MB 초과한 파일을 없애버리겠다    // 아직 변경된적없는 초기상태에서 5MB 초과하는 이미지를 선택한 경우  if(statusCheck == -1){  imageInput.value = '';    } else{ // 기존에 변경하려고 선택한 이미지가 있는데  // 다음에 선택한 이미지가 최대 크기를 초과한 경우  // -> 비워버리면 안되고, 그 전에 선택한 이미지로 계속 보이게끔 처리해야함.  // 백업의 백업본  **const** temp = backupInput.cloneNode(true);  // input 요소 다음에 백업 요소 추가  imageInput.after(backupInput);  // 화면에 존재하는 기존 input 제거  imageInput.remove();  // imageInput 변수에 백업을 대입해서 대신하도록 함  imageInput = backupInput;  // 화면에 추가된 백업본에는  // 이벤트 리스너가 존재하지 않기 때문에 추가  imageInput.addEventListener("change", changeImageFn);  // 한번 화면에 추가된 요소(backupInput)는 재사용 불가능  // backupInput의 백업본이 temp를 backupInput 으로 변경  backupInput = temp;  }  return; // 다른 코드 수행할필요없이 바로 return  }    // ------------- 선택된 이미지 미리보기 ----------------  // JS에서 파일을 읽을 때 사용하는 객체  // - 파일을 읽고 클라이언트 컴퓨터에 저장할 수 있음  /\*FileReader 객체는 웹 애플리케이션에서 비동기적으로 파일의 내용을 읽을 수 있게 해줍니다. \*/  **const** reader = new FileReader();  // 선택한 파일(file) 을 읽어와  // BASE64 인코딩 형태로 읽어와 result 변수에 저장  reader.readAsDataURL(file); // -> 읽어오기 이벤트(load)  // readAsDataURL() : 파일을 BASE64 형식의 데이터 URL로 읽어들입니다.    // console.log("reader:",reader);  // result에 "" 이런식으로 들어감  // 읽어오기 끝났을 때 (파일 읽기 작업이 완료되면 이벤트 핸들러 함수를 실행)  reader.addEventListener("load", e **=>** {  // e.target == reader  // 읽어온 이미지 파일이 BASE64 형태로 반환됨  **const** url = e.target.result; // reader.result  // 프로필 이미지(img)에 src속성으로 url값 세팅  profileImg.setAttribute("src", url);    // 새 이미지 선택 상태를 기록  statusCheck = 1;  // 파일이 선택된 input을 복제해서 백업  backupInput = imageInput.cloneNode(true);  });  };    // 2) imageInput에 change 이벤트로 changeImageFn 등록  // change 이벤트 : 새로운 값이 기존 값과 다를 경우 발생  imageInput.addEventListener("change", changeImageFn);  // ------------ 4) x버튼 클릭 시 기본 이미지로 변경 ----------------  deleteImage.addEventListener("click", () **=>** {  // 프로필 이미지(img)를 기본 이미지로 변경  profileImg.src = "/images/user.png";  // input에 저장된 값(value)를 ''(빈칸)으로 변경  // -> input에 저장된 파일 정보가 모두 사라짐 == 데이터 삭제  imageInput.value = '';  backupInput = undefined; // 백업본도 삭제  // 삭제 상태임을 기록  statusCheck = 0;  });  // ------------ 5) #profile (form) 제출 시 -----------------  profile.addEventListener("submit", e **=>** {    **let** flag = true;  // loginMemberProfileImg : myPage-profile.html 하단에 script를 이용하여 타임리프로 선언해둔 변수    // submit 해도 되는 경우 :  // 1. 기존 프로필 이미지가 없다가 새 이미지가 선택된 경우  if(loginMemberProfileImg == null && statusCheck == 1) flag = false;  // 2. 기존 프로필 이미지가 있다가 삭제한 경우  if(loginMemberProfileImg != null && statusCheck == 0) flag = false;    // 3. 기존 프로필 이미지가 있다가 새 이미지가 선택된 경우  if(loginMemberProfileImg != null && statusCheck == 1) flag = false;  // 나머지의 경우는 기존 상태에서 변경사항이 없는 경우임.-> 제출막기  if(flag){ // flag 값이 true인 경우  e.preventDefault();  alert("이미지 변경 후 클릭하세요")  }  });  }  /\* [input type="file" 사용 시 유의 사항]  1. 파일 선택 후 취소를 누르면  선택한 파일이 사라진다 (value == '')  2. value로 대입할 수 있는 값은 '' (빈칸)만 가능하다  3. 선택된 파일 정보를 저장하는 속성은  value가 아니라 files이다  \*/ |
| --- |

ㅎㅎㅎ

Pagination 객체 필드

| **private** **int** currentPage; // 현재 페이지 번호  **private** **int** listCount; // 전체 게시글 수    **private** **int** limit = 10; // 한 페이지 목록에 보여지는 게시글 수  **private** **int** pageSize = 10; // 보여질 페이지 번호 개수    **private** **int** maxPage; // 마지막 페이지 번호  **private** **int** startPage; // 보여지는 맨 앞 페이지 번호  **private** **int** endPage; // 보여지는 맨 뒤 페이지 번호    **private** **int** prevPage; // 이전 페이지 모음의 마지막 번호  **private** **int** nextPage; // 다음 페이지 모음의 시작 번호 |
| --- |

| <!DOCTYPE html>  <html lang="ko" xmlns:th="http://thymeleaf.org">  <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>게시판 이름</title>  <th:block th:replace="~{common/common}"></th:block>  <link rel="stylesheet" href="/css/board/boardList-style.css">  </head>  <body>  <main>  <th:block th:replace="~{common/header}"></th:block>  <section class="board-list">  <th:block th:each="boardType : ${application.boardTypeList}">    <h1 class="board-name"  th:if="${boardType.boardCode} == ${boardCode}"  th:text="${boardType.boardName}"  >게시판 이름</h1>    </th:block>  <div class="list-wrapper">  <table class="list-table">  <thead>  <tr>  <th>글번호</th>  <th>제목</th>  <th>작성자</th>  <th>작성일</th>  <th>조회수</th>  <th>좋아요</th>  </tr>  </thead>  <tbody>  <!-- 게시글이 존재하지 않을 때 -->    <!-- 여러행 조회 시 결과가 없을 경우 == List 가 비어있음 -->    <!-- #lists : 타임리프에서 list 관련 기능을 제공하는 객체 -->  <!-- ${#lists.size(boardList) == 0} -->  <th:block th:if="${#lists.isEmpty(boardList)}">  <tr>  <th colspan="6">게시글이 존재하지 않습니다.</th>  </tr>  </th:block>  <!-- 게시글이 존재할 때 -->  <th:block th:unless="${#lists.isEmpty(boardList)}">    <tr th:each="board : ${boardList}" th:object="${board}">    <td th:text="\*{boardNo}">게시글 번호</td>    <td>  <!-- 썸네일 추가 예정 -->    <a th:text="\*{boardTitle}">게시글 제목</a>    <th:block th:text="|[\*{commentCount}]|">댓글 수</th:block>  </td>    <!-- 작성자 닉네임 -->  <td th:text="\*{memberNickname}">닉네임</td>    <!-- 작성일 -->  <td th:text="\*{boardWriteDate}">2023-10-26</td>    <!-- 조회수 -->  <td th:text="\*{readCount}">0</td>    <!-- 좋아요 수 -->  <td th:text="\*{likeCount}">0</td>    </tr>  </th:block>  </tbody>  </table>  </div>  <div class="btn-area">  <!-- 로그인 상태일 때만 글쓰기 버튼 노출 -->  <button id="insertBtn">글쓰기</button>  </div>  <!-- ============================================================================================================== -->  <!-- 게시글이 있을 때만 페이지네이션이 보이게 할 예정 -->  <div class="pagination-area">  <ul class="pagination" th:object="${pagination}">  <!-- 첫 페이지로 이동 -->  <li><a th:href="@{/board/{boardCode}(boardCode=${boardCode}, cp=1)}">&lt;&lt;</a></li>  <!-- 이전 목록 마지막 번호로 이동 -->  <li><a th:href="@{/board/{boardCode}(boardCode=${boardCode}, cp=\*{prevPage})}">&lt;</a></li>  <!-- 특정 페이지로 이동 -->    <th:block th:each="i : \*{#numbers.sequence(startPage, endPage)}">    <!-- 현재 보고있는 페이지 -->  <li th:if="${i} == \*{currentPage}">  <a class="current" th:text="${i}">현재페이지</a>  </li>    <!-- 보고있지 않은 페이지 -->  <li th:unless="${i} == \*{currentPage}">  <a th:href="@{/board/{boardCode}(boardCode=${boardCode}, cp=${i})}"  th:text="${i}">이동할 페이지</a>  </li>  </th:block>      <!-- 다음 목록 시작 번호로 이동 -->  <li><a th:href="@{/board/{boardCode}(boardCode=${boardCode}, cp=\*{nextPage})}">&gt;</a></li>  <!-- 끝 페이지로 이동 -->  <li><a th:href="@{/board/{boardCode}(boardCode=${boardCode}, cp=\*{maxPage})}">&gt;&gt;</a></li>  </ul>  </div>  <!-- 검색창 -->  <form action="#" method="get" id="boardSearch">  <select name="key" id="searchKey">  <option value="t">제목</option>  <option value="c">내용</option>  <option value="tc">제목+내용</option>  <option value="w">작성자</option>  </select>  <input type="text" name="query" id="searchQuery" placeholder="검색어를 입력해주세요.">  <button>검색</button>  </form>  </section>  </main>  <th:block th:replace="~{common/footer}"></th:block>  </body>  </html> |
| --- |

게시글 상세 조회에서 사용할 select문 3개

| -------------------------------------------------------  /\* 게시글 상세 조회 \*/  SELECT BOARD\_NO, BOARD\_TITLE, BOARD\_CONTENT, BOARD\_CODE, READ\_COUNT,  MEMBER\_NO, MEMBER\_NICKNAME, PROFILE\_IMG,    TO\_CHAR(BOARD\_WRITE\_DATE, 'YYYY"년" MM"월" DD"일" HH24:MI:SS') BOARD\_WRITE\_DATE,  TO\_CHAR(BOARD\_UPDATE\_DATE, 'YYYY"년" MM"월" DD"일" HH24:MI:SS') BOARD\_UPDATE\_DATE,    (SELECT COUNT(\*)  FROM "BOARD\_LIKE"  WHERE BOARD\_NO = 1960) LIKE\_COUNT,    (SELECT IMG\_PATH || IMG\_RENAME  FROM "BOARD\_IMG"  WHERE BOARD\_NO = 1960  AND IMG\_ORDER = 0) THUMBNAIL  FROM "BOARD"  JOIN "MEMBER" USING(MEMBER\_NO)  WHERE BOARD\_DEL\_FL = 'N'  AND BOARD\_CODE = 1  AND BOARD\_NO = 1960;  ---------------------------------------------  /\* 상세조회 되는 게시글의 모든 이미지 조회 \*/  SELECT \*  FROM "BOARD\_IMG"  WHERE BOARD\_NO = 1960  ORDER BY IMG\_ORDER;  ---------------------------------------------  /\* 상세조회 되는 게시글의 모든 댓글 조회 \*/  /\*계층형 쿼리\*/  SELECT LEVEL, C.\* FROM  (SELECT COMMENT\_NO, COMMENT\_CONTENT,  TO\_CHAR(COMMENT\_WRITE\_DATE, 'YYYY"년" MM"월" DD"일" HH24"시" MI"분" SS"초"') COMMENT\_WRITE\_DATE,  BOARD\_NO, MEMBER\_NO, MEMBER\_NICKNAME, PROFILE\_IMG, PARENT\_COMMENT\_NO, COMMENT\_DEL\_FL  FROM "COMMENT"  JOIN MEMBER USING(MEMBER\_NO)  WHERE BOARD\_NO = 1960) C  WHERE COMMENT\_DEL\_FL = 'N'  OR 0 != (SELECT COUNT(\*) FROM "COMMENT" SUB  WHERE SUB.PARENT\_COMMENT\_NO = C.COMMENT\_NO  AND COMMENT\_DEL\_FL = 'N')  START WITH PARENT\_COMMENT\_NO IS NULL  CONNECT BY PRIOR COMMENT\_NO = PARENT\_COMMENT\_NO  ORDER SIBLINGS BY COMMENT\_NO  ; |
| --- |