

2024년 1학기

소프트웨어프로젝트1

1차 프로젝트

컴퓨터정보공학부

2021202039

이소영

1. Introduction

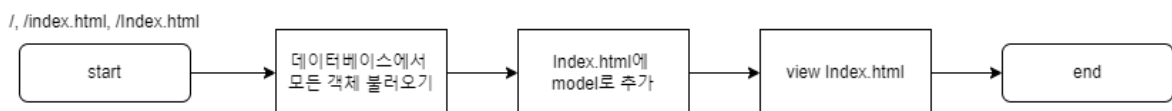
본 프로젝트는 Spring Boot를 활용한 서버 구현을 목표로 한다. 세부적으로는, MVC 패턴을 이용해 서버를 구현하고, H2 database를 사용하여 파일 업로드 리소스를 컨트롤한다.

사진을 조회할 수 있는 웹 페이지를 구현하는데, 크게 사진을 보여주는 홈 페이지와 사진 업로드 페이지로 구성되어 있다. 주요 기능으로는 사진 업로드, 사진 관련 타이틀과 글 작성 기능이 있다. 즉, 사진을 업로드하고 해당 사진에 대한 간단한 소개를 작성하는 홈페이지를 구현하는 것을 목표로 한다.

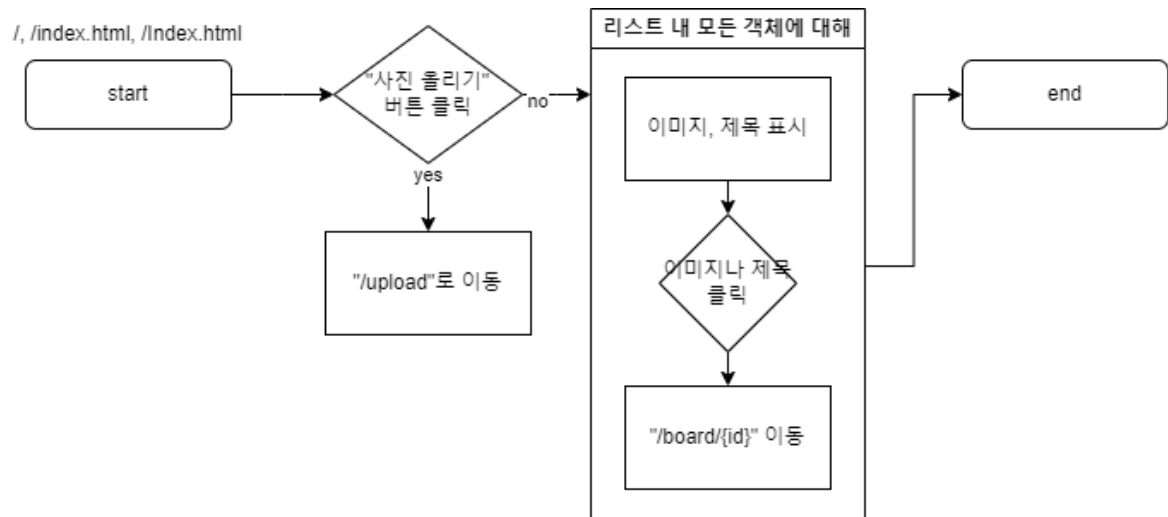
2. Flow Chart

1) 메인 홈페이지(Index.html)

Controller

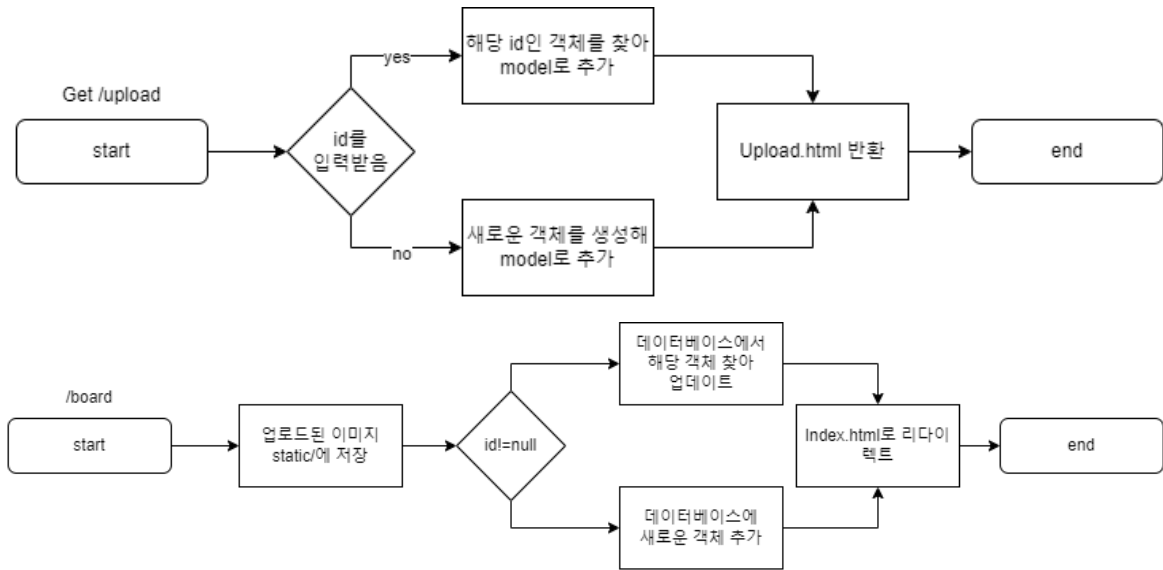


HTML

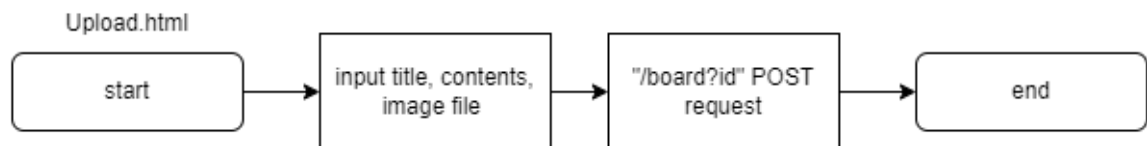


2) 업로드와 수정(Upload.html)

Controller

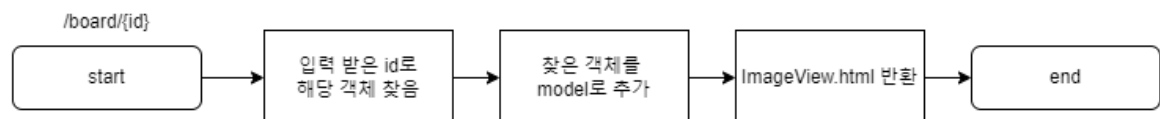


HTML

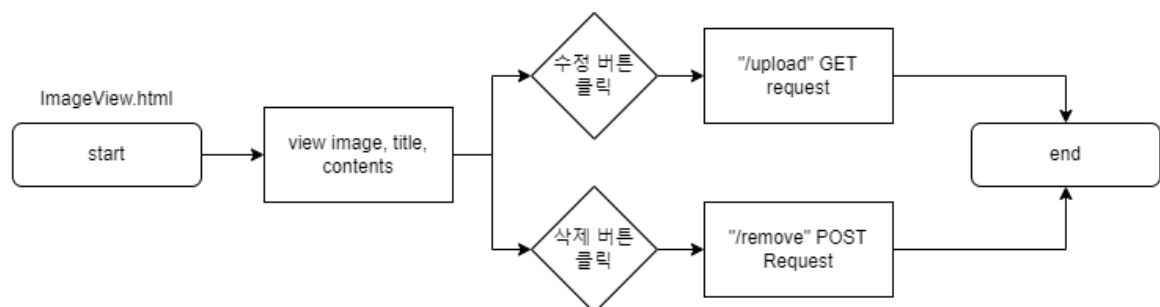


3) 이미지 조회(ImageView.html)

Controller

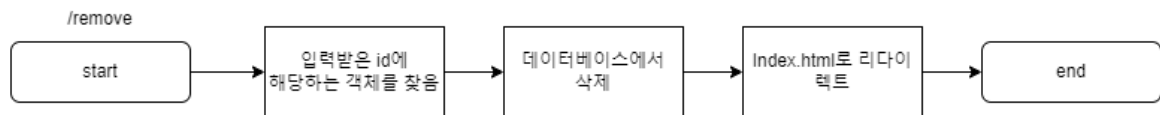


HTML



4) 삭제

Controller



3. Result

서버 실행 결과 화면과 구현 방식에 대한 설명이다. 프로젝트 내 MVC Pattern은 각 기능별에서 설명한다.

```

public class Board {
    @Id
    @Column(name="id", nullable = false)
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    // 데이터베이스 아이디
    private long id;

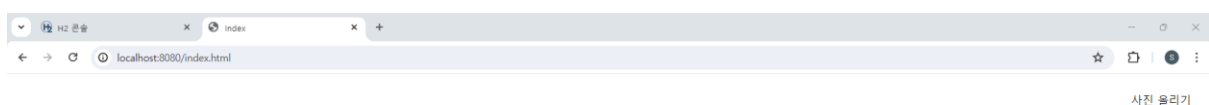
    // 제목
    @Column
    private String title;

    // 내용
    @Column
    private String contents;

    // 이미지 이름(경로)
    @Column
    private String image;
}
  
```

데이터베이스 테이블은 위와 같이 구성되었다. 데이터를 구분하기 위한 고유 id, 제목인 title, 내용인 contents, 이미지 경로를 의미하는 image가 있다.

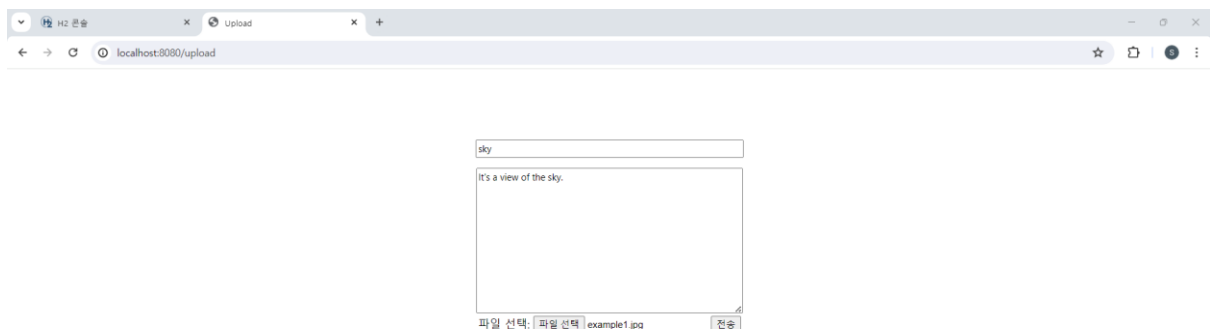
localhost:8080(localhost:8080/)으로 처음 접속한 화면이다. localhost:8080/index.html로도 접속이 가능하다.



“사진 올리기” 버튼을 클릭하여 Upload.html 양식으로 이동하였다. “사진 올리기”에는 “/upload” 링크가 연결되어 있다. 이 링크는 “Upload.html” 뷰를 반환한다.

“/upload”는 id를 매개변수로 전달받는데, 이때 Index.html에서 “사진 올리기” 버튼을 클릭하는 경우, 새로운 데이터를 생성하는 것이므로 기존 객체의 id를 보내지 않는다. 따라서 /Upload.html로 이동 시 컨트롤러에서 새로운 Board 객체를 생성하여 모델로 추가한다.

이때 매개변수로 id가 전달되지 않았으므로, 새로운 데이터 생성이라고 인식한다.



원하는 정보를 입력한 후, “전송” 버튼을 클릭하면, 서버로 “/board” POST 요청을 보낸다. 컨트롤러에서는 이를 확인하여 업로드된 이미지를 지정된 위치(static/)에 복사하고, 새로운 Board 객체를 생성하여 title, contents, image를 설정한다. 그 후 리포지터리와 서비스를 이용해 데이터베이스에 추가한다. 성공하면 Index.html 뷰를 반환한다.

Index.html로 이동하여, 새롭게 생긴 데이터를 확인할 수 있다.

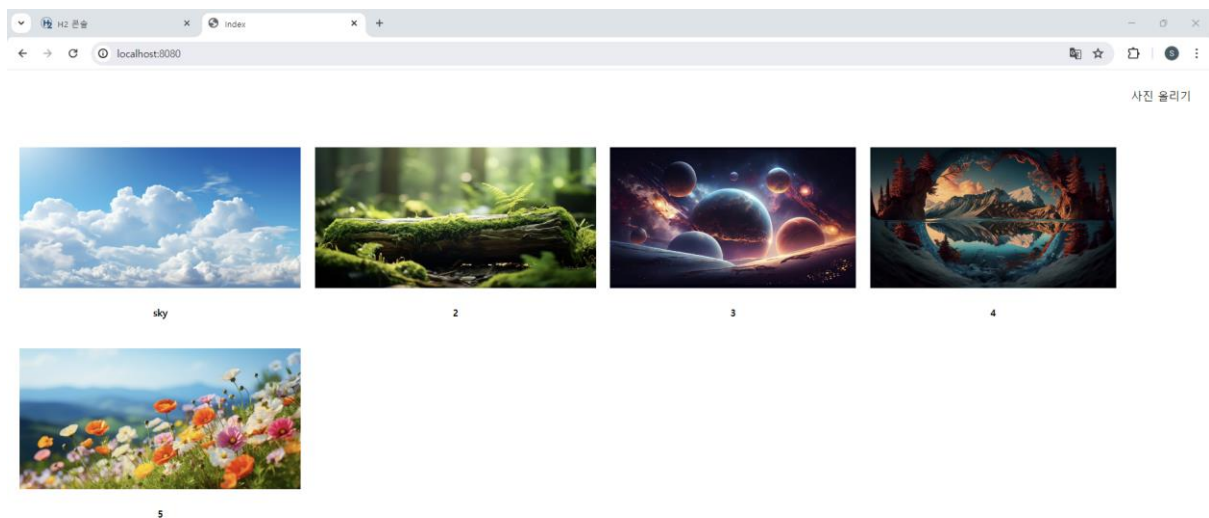


이미지나 제목을 클릭하면, 해당 데이터를 조회할 수 있는 ImageView.html로 이동한다. 이는

"/board/{id}" GET 요청으로 컨트롤러에서 해당 id의 객체를 찾아 모델로 추가한다. 그 후 "ImageView.html" 뷰를 반환한다.



같은 방식으로 여러 이미지를 추가한 모습이다. "/", "/index.html" GET 요청으로 Index.html에 접속할 때, 리포지터리에서 모든 Board 객체를 List로 불러온다. 이 List를 모델로 추가하여 Index.html 뷰를 반환하므로, 데이터베이스에 저장된 모든 데이터를 확인할 수 있다.



가장 첫 번째 이미지인 "sky"를 조회한 후 "수정" 버튼을 클릭하면 Upload.html로 이동한다. "/upload" GET 요청을 하면, 이번에는 script를 통해 id를 전달하므로 컨트롤러에서는 Upload.html 뷰 반환 시 해당 id를 가진 Board 객체를 전달한다.

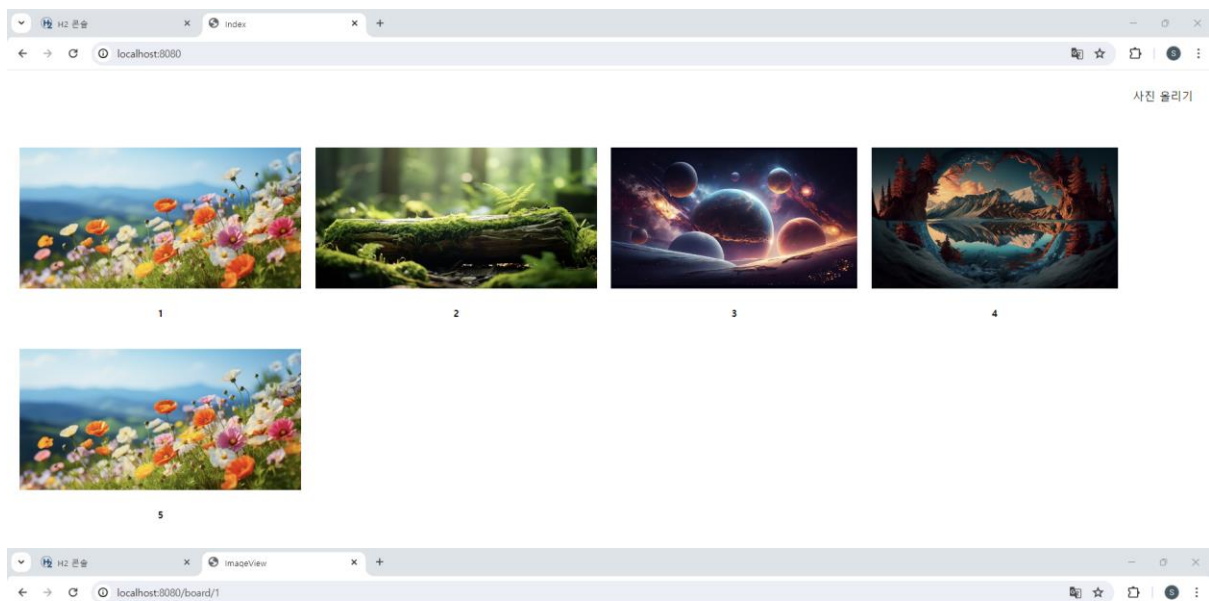
```
function UpdateRequest(id){  
    <!-- 수정 버튼을 눌렀을 경우 이동 -->  
    window.location.href = '/upload?id='+id;  
}
```

이때 해당 데이터 객체의 id 1이 매개변수로 전달된 것을 url에서 확인할 수 있다. 원하는 정보로 수정하여, "전송" 버튼을 클릭한다.

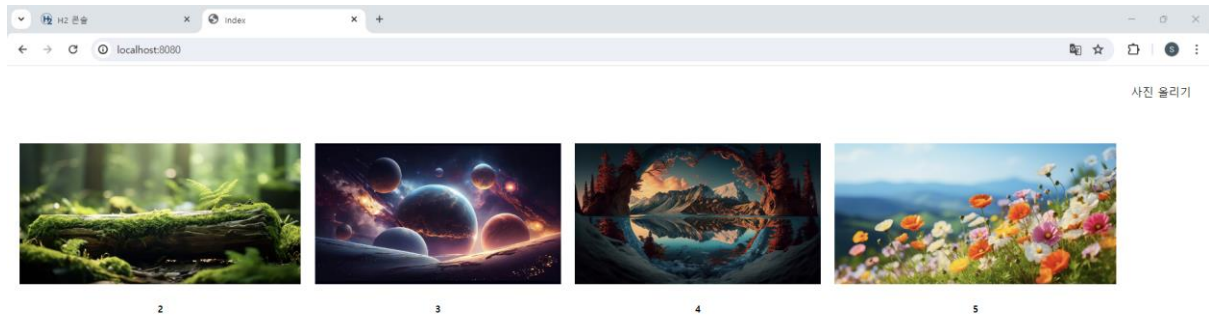
서버로 "/board" POST 요청을 하여 컨트롤러에서는 id가 null이 아니므로 수정으로 인식하여 id 객체를 찾아 수정 후 저장한다.



다시 Index.html 뷰를 반환하여 리다이렉트되고, 이미지를 조회했을 때 제목과 내용까지 데이터가 수정됐음을 확인할 수 있다. 수정이므로 업로드 순서는 유지된다.



다시 이미지나 제목을 클릭하여 데이터를 조회하는 ImageView.html(localhost:8080/board/1)에서 “삭제” 버튼을 클릭하는 경우 서버로 “/remove” POST 요청을 보낸다. 삭제하려는 데이터의 id를 매개변수로 받는데, 데이터베이스에서 삭제 후 Index.html 뷰를 반환한다. Index.html로 리다이렉트되고, 데이터가 삭제된 것을 확인할 수 있다.



데이터베이스에서 역시 데이터가 삭제됐음을 확인할 수 있다.

ID	CONTENTS	IMAGE	TITLE
1	2 2	example2.jpg	2
2	3 3	example3.jpg	3
3	4 4	example4.jpg	4
4	5 5	example5.jpg	5

4. Consideration

몇몇 GET 요청에 대해 오류가 발생하는 경우가 있었다. 오류 로그는 다음과 같다.

java.lang.IllegalArgumentException: Name for argument of type [java.lang.Long] not specified, and parameter name information not available via reflection. Ensure that the compiler uses the '-parameters' flag.

검색 결과 @PathVariable, @RequestParam 등의 어노테이션을 사용할 때 value 값을 정확히 지정 해주지 않으면 발생하는 오류였다. 컴파일러 옵션을 변경해서 해결 가능하지만, 명확성을 위해 코드에 value를 추가하였다.

```
//ImageView.html로 이동
@GetMapping("/{id}")
public String showImageView(Model model, @PathVariable("id") Long id){
    Optional<Board> board=boardRepository.findById(id); // id로 객체를 찾음
    Board boardEntity=board.get();
    model.addAttribute( attributeName: "board",boardEntity);
    return "ImageView";
}
```


Index.html에서는 이미지를 정상적으로 확인할 수 있지만, ImageView.html에서는 같은 링크를 연결하였음에도 이미지를 정상적으로 불러올 수 없는 오류가 있었다.

그 이유로는 url과 관련이 있다. Index.html에서는 localhost:8080/image.jpg와 같이 요청하여 static 디렉토리에서 이미지를 찾아 반환한다. 그러나 ImageView.html은 localhost:8080/board/{id}로 요청을 했기 때문에 Index.html에서와 같이 이미지 경로를 입력하면 localhost:8080/board/image.jpg와 같이 탐색했기 때문에 발생한 문제였다.

ImageView.html에서 이미지 출처를 /image.jpg와 같이 절대경로로 접근하도록 수정하였다. 그 결과 static 폴더에서 이미지를 찾아 정상적으로 동작했다.

```
<div>
  
```

데이터베이스에 정상적으로 정보가 업로드되고, 이미지 파일 역시 static에 잘 존재하며, 링크도 제대로 연결됐음에도 이미지가 보이지 않는 문제가 있었다. 이 문제는 static의 특성과 관련된 문제였다. static은 정적 콘텐츠를 가지고 있는 폴더이므로 프로젝트 빌드 후 실행 시 static에 있던 파일들은 자동으로 build(out) 폴더에 포함되고, 이 build(out)/resources/static에 있는 파일만이 정상적으로 보여졌던 것이다. 그러니 서버 실행 중 static에 추가되는 이미지 파일들은 build(out)에 포함되지 않았으므로 이 정적 콘텐츠를 불러올 수 없었다. 이 문제를 해결하기 위해서는 업로드 테스트에 쓸 이미지 파일들을 미리 static 폴더에 복사한 후 서버를 실행하는 것이다.