Software Project

Project 1

(Upload a Picture)

학 과: 컴퓨터정보공학부

담당교수: 이우신 교수님

실습분반: 화2 목1

학 번: 2020202060

성 명: 홍왕기

Introduction

이 프로젝트는 MVC 패턴을 기반으로 하며, H2 데이터베이스와 Spring 프레임워크, HTML을 활용한다. 사용자는 사진을 업로드하고, 제목과 내용을 작성하며, 업로드 된 사진을 최신순으로 조회할 수 있다. 또한, 업로드 된 사진의 상세 정보를 확인하고 수정하거나 삭제할 수 있는 것을 목표로 한다.

홈페이지(Index.html)는 데이터베이스에 저장된 사진들을 최신순으로 표시하며, 업로드 된 사진이 없을 경우 업로드 기능을 제공한다. 사용자가 사진을 업로드하면, 사진과 함께 제목과 내용이 데이터베이스에 저장된다. 업로드 된 사진은 다시 홈페이지에 반영되어 최신순으로 보이게 구현해야 한다.

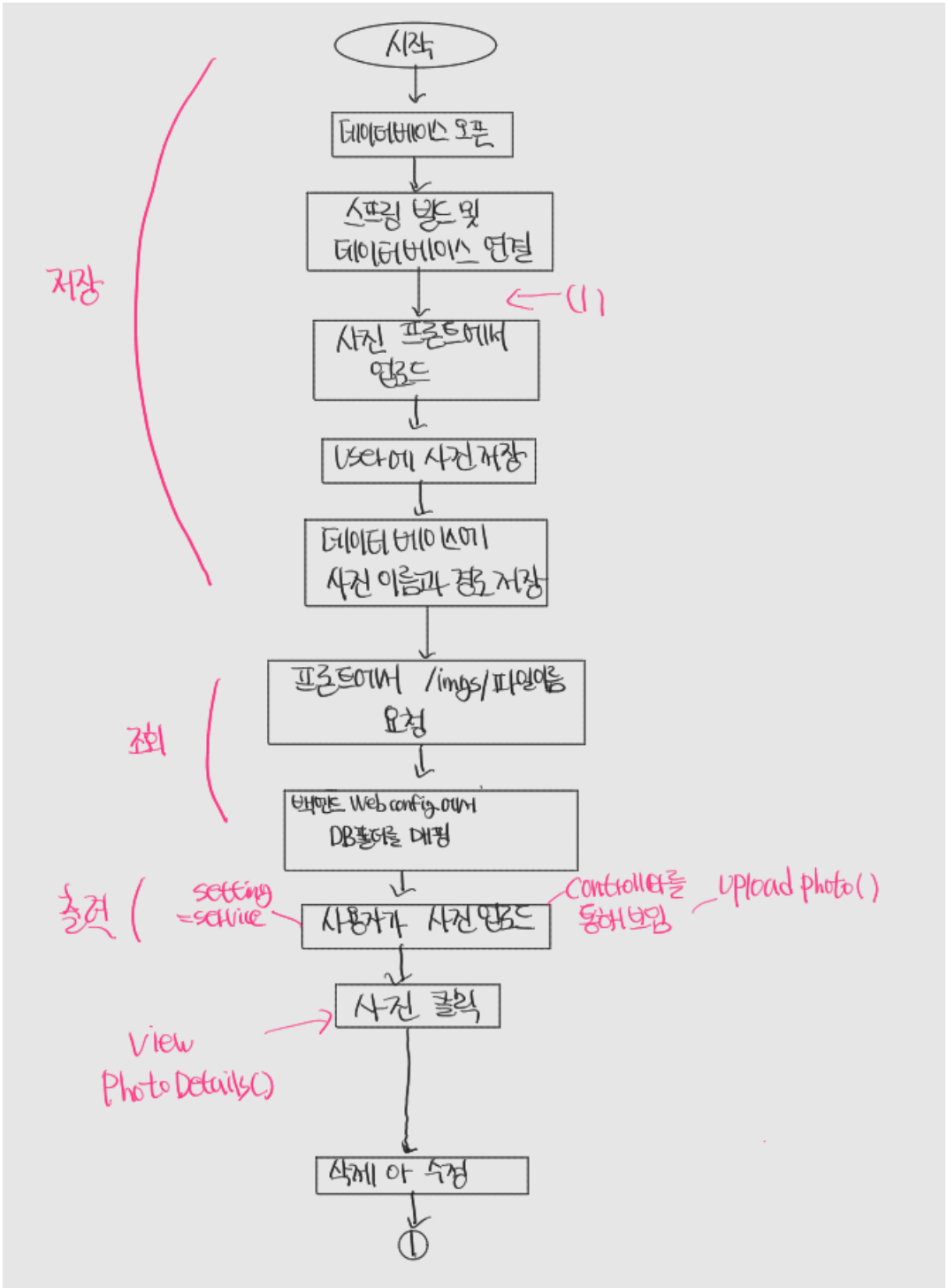
사진 업로드 페이지(Upload.html)는 사용자가 사진과 그에 대한 제목, 내용을 입력하여 업로드할 수 있는 인터페이스를 제공한다. 사용자가 입력한 정보는 Spring 컨트롤러를 통해 처리되고, H2 데이터베이스에 저장된다. 저장된 사진은 홈페이지에서 최신순으로 조회할 수 있다.

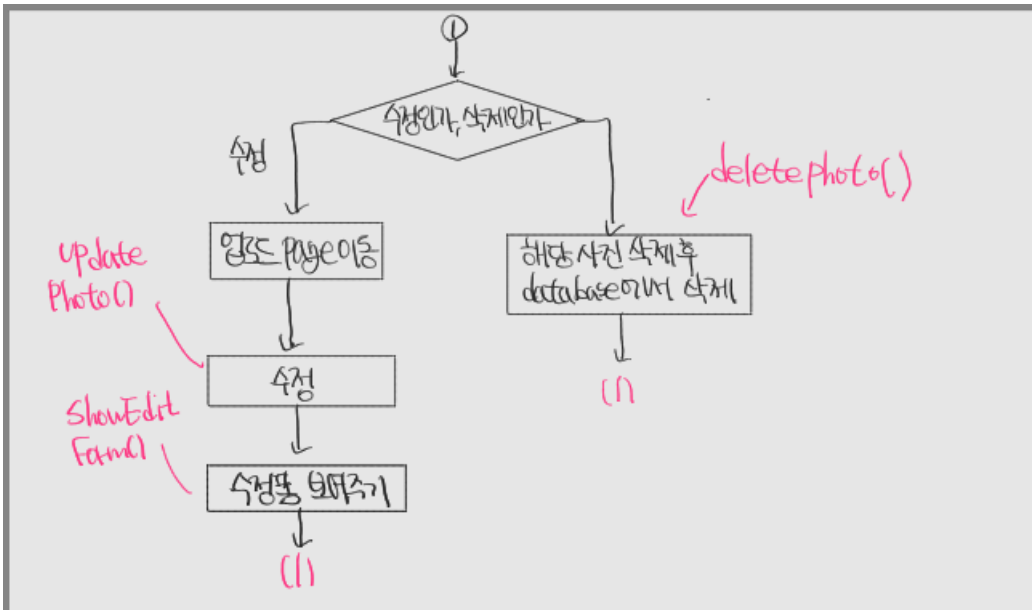
사진 상세보기 및 관리 페이지(ImageView.html)는 사용자가 특정 사진을 클릭했을 때 접근할 수 있는 페이지로, 사진의 제목, 내용, 사진 자체를 확인할 수 있으며, 삭제 및 수정 기능을 제공해야 한다. 사진을 삭제하면 데이터베이스에서 해당 사진이 제거되고, 수정된 내용은 데이터베이스에 반영된다.

이 프로젝트는 Spring 프레임워크를 통해 애플리케이션의 구조를 관리하고, H2 데이터베이스를 사용하여 데이터를 저장하며, HTML을 사용해 사용자 인터페이스를 구현한다. 이를 통해 사용자는 사진을 쉽게 업로드하고, 관리하며, 최신 정보를 확인할 수 있을 것이다.

Flow Chart

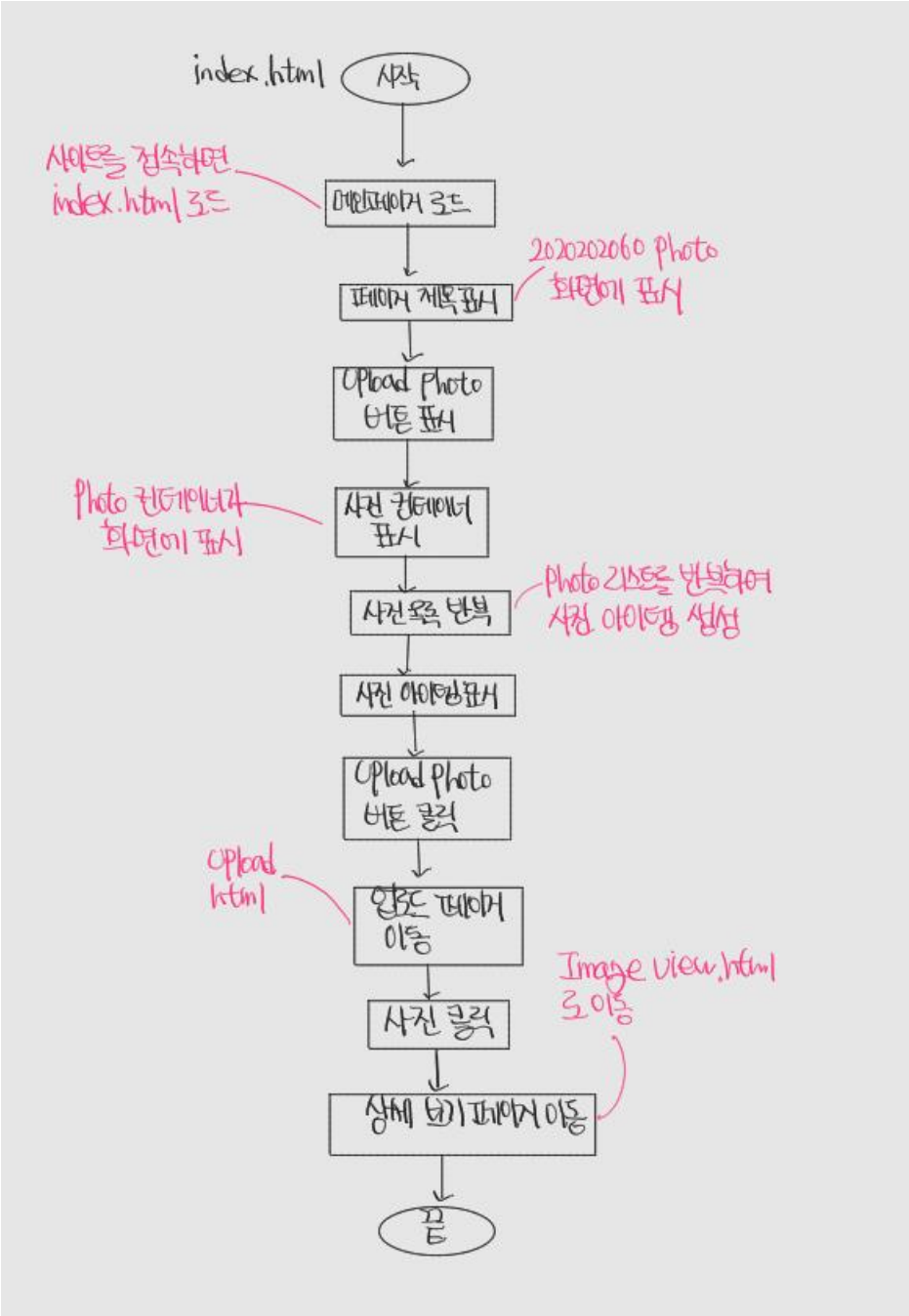
전체적인 흐름도:




수정인가, 삭제인가
수정
삭제
delete photo()
update Photo()
ShowEdit Form()
(1)
(1)

각 View(html) 마다의 흐름도 설명:

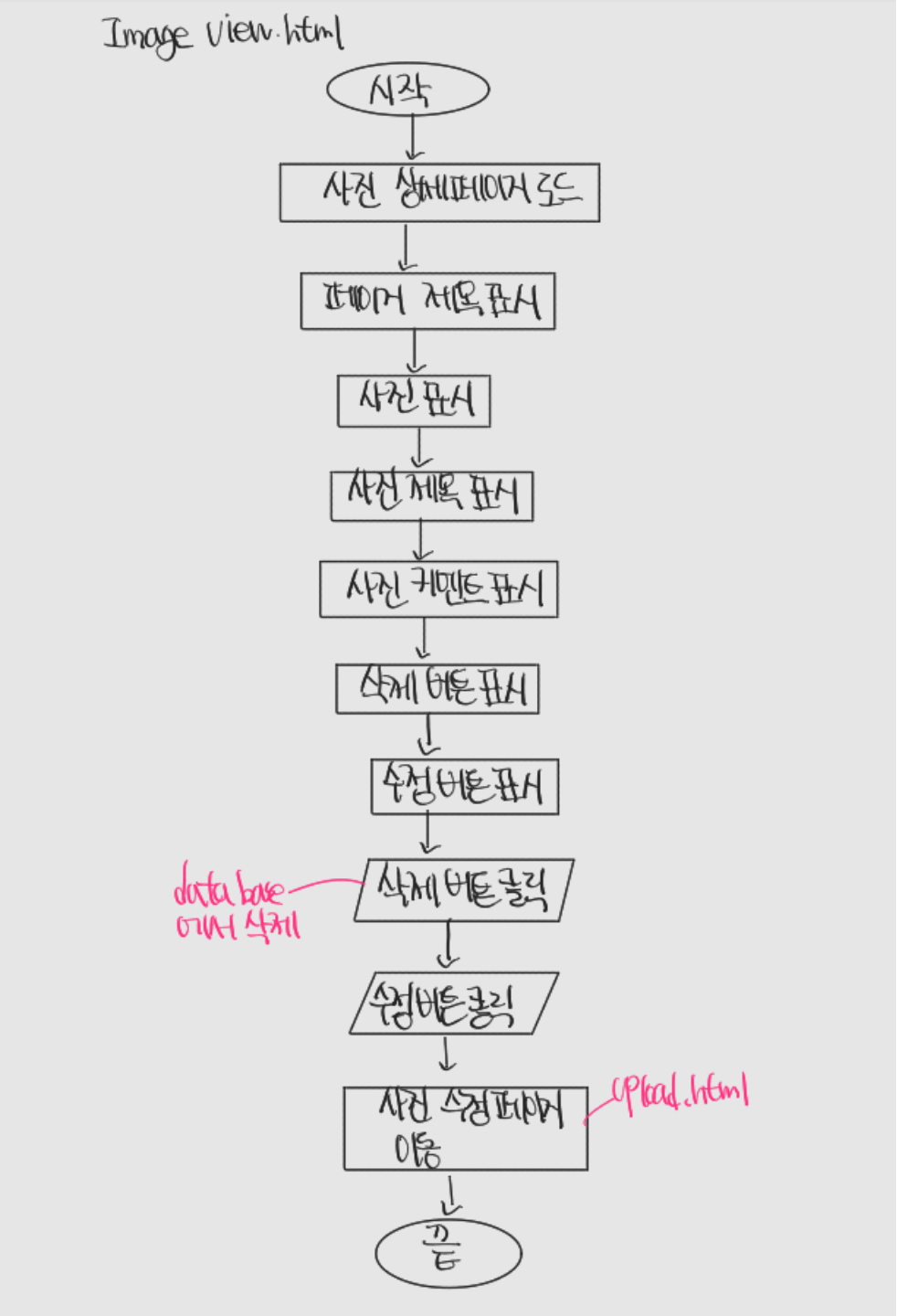
Index.html



Upload.html


Upload.html
시작
Upload.html 로드
업로드 페이지 로드
Upload Photo 폼
페이지 제목 폼 서식
업로드 폼 서식
Input : title (최대 20자)
제목입력 필드 서식
Input : command (최대 100자)
코멘트입력 필드 서식
파일선택 필드 서식
전송 버튼 서식
폼입력
전송버튼 클릭
데이터가 서버로 전송
폼데이터 제출
서버에서 파일 업로드 처리
업로드 완료
끝

Imageview.html



Result

MVC Pattern:

이 코드에서 MVC 패턴은 다음과 같이 나타내어진다.

Model (모델):

모델은 애플리케이션의 데이터를 정의하고 조작하는 부분이다. 이 부분은 데이터의 구조와 데이터베이스와의 상호작용을 담당한다. 주로 데이터베이스에서 데이터를 가져오거나 저장하고, 비즈니스 로직을 처리한다.

이 코드에서는 Photo 클래스가 모델을 나타내며, 각 사진의 제목, 코멘트, 파일명, 파일 경로 등의 데이터를 포함한다. 또한, PhotoRepository 인터페이스를 통해 데이터베이스와의 상호작용을 처리한다.

View (뷰):

뷰는 사용자에게 데이터를 시각적으로 표시하고 사용자 인터페이스를 제공하는 부분이다. 이 부분은 HTML, CSS 등의 웹 기술을 사용하여 사용자에게 정보를 표시한다. 또한, 사용자의 입력을 받아 컨트롤러에 전달한다. 이 코드에서는 Thymeleaf를 사용하여 HTML 파일이 뷰를 나타내며, 사용자에게 사진 갤러리를 표시하고 사용자의 요청을 컨트롤러에 전달하는 역할을 한다.

Controller (컨트롤러):

컨트롤러는 사용자의 요청을 받아 처리하고, 모델과 뷰 사이의 상호작용을 조정하는 부분이다. 이 부분은 요청을 분석하여 적절한 모델의 메서드를 호출하고, 그 결과를 뷰에 전달한다

이 코드에서는 PhotoController 클래스가 컨트롤러를 나타내며, 사용자의 요청을 처리하고 모델과의 상호작용을 관리한다. 각 요청에 대한 핸들러 메서드를 정의하고, 해당 요청에 따른 데이터를 뷰에 전달하는 역할을 한다.

WebConfig:

WebConfig 클래스는 웹 애플리케이션의 설정을 담당하는 부분이다. 이 부분은 정적 리소스에 대한 핸들링을 설정하고, 웹 애플리케이션의 전반적인 동작을 구성한다.

이 코드에서는 정적 리소스에 대한 핸들러를 추가하여, 이미지와 같은 정적 파일들을 제공한다. 설정된 핸들러는 브라우저로부터의 요청에 따라 적절한 정적 파일을 제공하는 역할을 한다.

Service (서비스):

PhotoService 클래스가 서비스를 나타냅니다. 서비스는 주로 비즈니스 로직을 처리하고, 데이터 엑세스 계층과 컨트롤러 간의 중개자 역할을 한다. 이 코드에서는 PhotoService가 비즈니스 로직을 담당하며, 데이터베이스와의 상호작용을 처리한다. 또한, PhotoService는 PhotoController에 데이터를 전달하여 웹 요청에 대한 응답을 생성한다.

각 View(html)별 구현 방식 설명 및 결과 화면 설명:

index.html:

이 HTML 파일은 애플리케이션의 홈페이지를 나타내는 파일이다. 사용자가 처음 접하는 페이지로서, 주로 다음과 같은 구성으로 이루어져 있다.

페이지 제목과 버튼:

페이지 상단에 본인 학번인 2020202060 Photo라는 제목이 표시되도록 설정하였다. 이는 해당 웹 애플리케이션의 이름이나 타이틀을 나타낸다.

"Upload Photo"라는 버튼을 생성하여 이 버튼을 클릭하면 사진을 업로드하는 페이지(upload.html)로 이동할 수 있도록 구현하였다.

사진 갤러리:

갤러리는 서버에서 가져온 사진 목록을 표시한다. Thymeleaf의 반복문을 사용하여 사진 목록을 동적으로 생성하였다. 각 사진은 이미지와 함께 해당 사진의 제목이 표시된다. 이때 사용자가 사진을 클릭하면 해당 사진의 세부 정보 페이지(imageview.html)로 이동한다.

이 HTML 파일은 사용자가 애플리케이션을 처음 접했을 때 보게 되는 화면이므로, 사용자에게 기능과 정보를 제공하는 역할을 한다.

upload.html:

이 HTML 파일은 사용자가 사진을 업로드하는 페이지를 나타낸다. 이는 다음과 같이 구현을 하였다.

페이지 제목과 폼:

먼저 "Upload Photo"라는 제목이 페이지 상단에 표시되도록 설정하였다.

사용자는 사진의 제목과 코멘트를 입력할 수 있는 폼을 제공하여 설정할 수 있게 구현하였다.

이미지 업로드 폼:

사용자는 로컬 파일 시스템에서 이미지 파일을 선택할 수 있는 입력 필드를 제공하게 구현하였다.

전송 버튼:

사용자가 입력한 정보와 선택한 이미지 파일을 서버로 전송하기 위한 "전송" 버튼을 배치하였다.

이 HTML 파일은 사용자가 새로운 사진을 업로드할 때 사용되는 페이지로서, 사진 업로드와 관련된 기능을 제공하는 파일이다.

imageview.html:

이 HTML 파일은 특정 사진의 세부 정보를 보여주는 페이지를 나타낸다. 사용자가 사진을 클릭하여 해당 페이지로 이동하면 다음과 같은 내용을 볼 수 있다.

사진 및 정보 표시:

선택한 사진이 크게 표시하여 사진의 제목과 내용을 표시할 수 있도록 구현하였다.

작업 버튼:

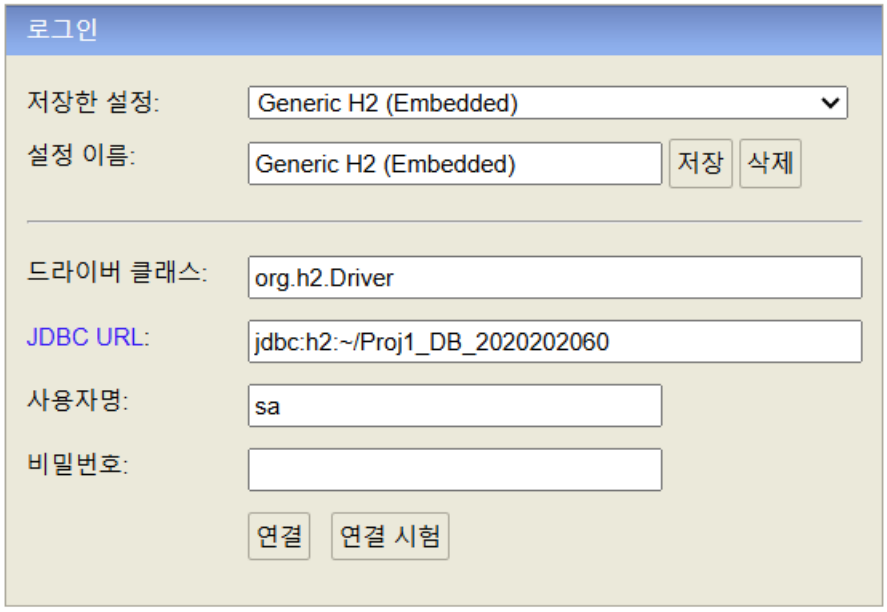
"Delete"와 "Edit" 버튼을 배치하여 사진을 삭제하거나 편집할 수 있도록 구현하였다.

"Delete" 버튼을 클릭하면 해당 사진이 삭제되고, "Edit" 버튼을 클릭하면 해당 사진을 편집할 수 있는 페이지(upload.html)로 이동할 수 있게 하여 사진을 다시 수정할 수 있게 사용자에게 제공하였다.

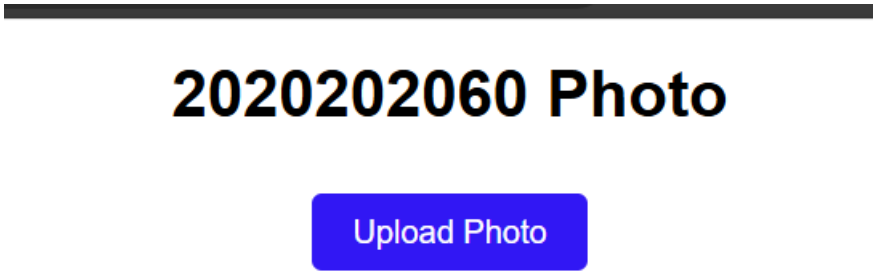
이 HTML 파일은 사용자가 선택한 사진의 세부 정보를 보여주고, 해당 사진에 대한 작업을 수행할 수 있는 페이지로서 사용된다.

결과 화면 및 설명

먼저 이 프로그램을 실행하기 위해서 데이터 베이스에 대한 설정을 다음과 같이 입력하였다.



연결한 후 디버깅을 하여 local.host 808을 입력하여 다음과 같이 나타내었다.



Index.html로 이동하여 사진을 업로드를 할 수 있는 버튼이 있음을 확인할 수 있다. Upload Photo를 클릭하면 다음과 같이 나타난다.

타이틀과 내용을 작성할 수 있고 파일을 선택하여 이미지를 전송할 수 있는 버튼이 있음을 확인할 수 있다.

위와 같이 이미지를 선택하여 전송한다면 다음과 같다.

2020202060 Photo

Upload Photo

이미지1

제목, 이미지 등이 업로드가 정상적으로 이루어진 것을 확인할 수 있다.

SELECT * FROM PHOTO;

ID	COMMENT	FILE_PATH	FILENAME	TITLE
321	이미지 1입니다.	C:\Users\leejd	im1.jpeg	이미지1

(1 row, 4 ms)

편집

이때 데이터 베이스에도 정상적으로 저장된 것을 볼 수 있다.

Photo Details

이미지1

이미지 1입니다.

Delete Edit

이미지를 클릭한다면 해당 이미지는 확대되고 삭제와 수정 기능이 생긴 것을 볼 수 있다. 만약 수정 버튼을 누른다면

Upload Photo

Input : Title (최대 20자)

Input : Comment (최대 100자)

파일 선택:

파일 선택 선택된 파일 없음

전송

Upload html로 이동하여 사진을 수정 및 추가를 할 수 있게 된다.

2020202060 Photo

Upload Photo

이미지2

이미지1

수정하여 위와 같이 사진이 하나 더 추가된 것을 볼 수 있다. 만약 이미지 1을 삭제할 경우

2020202060 Photo

Upload Photo

이미지2

SELECT * FROM PHOTO;

ID	COMMENT	FILE_PATH	FILENAME	TITLE
322	이미지 2입니다.	C:\Users\leejd	im2.jpeg	이미지2

(1 row, 2 ms)

편집

이미지 1이 삭제되고 데이터 베이스에 삭제된 형태를 볼 수 있다.

만약 모든 것을 종료하고 다시 디버깅을 한다면 다음과 같이 기존에 있던 데이터 베이스에 대한 사진은 시작과 동시에 출력됨을 확인할 수 있다.

고찰:

이 프로젝트를 진행하면서 여러 가지 문제점이 발생했고, 이를 해결하기 위해 다음과 같이 해결하였다.

첫 번째 문제는 대용량 파일 업로드 시의 성능 문제이다. 고해상도 사진과 같은 대용량 파일을 업로드할 때 서버의 메모리 사용량이 급격히 증가하면서 서버 응답 속도가 느려지고, 심할 경우 서버가 다운되는 문제가 발생했다. 이를 해결하기 위해 대용량 파일 업로드를 처리하는 방식을 properties를 수정하여 개선했다. 이를 통해 서버의 안정성을 높이고 대용량 파일 업로드 시의 성능 문제를 해결했다.

두 번째 문제는 브라우저 호환성 문제이다. 일부 오래된 브라우저나 특정 브라우저에서 사진 업로드 페이지가 제대로 표시되지 않거나 기능이 작동하지 않는 문제가 있었다. 이를 해결하기 위해 웹 페이지를 개발할 때 다양한 브라우저에서의 호환성을 고려하여 테스트를 수행하고, CSS 및 Java 코드를 표준화하여 브라우저 간의 호환성을 개선했다.

세 번째로는 데이터베이스 연동 문제이다. 파일 업로드 시 메타데이터를 데이터베이스에 저장할 때, 데이터베이스 연결이 불안정하거나 트랜잭션이 올바르게 처리되지 않는 문제가 발생했다. 이를 해결하기 위해 데이터베이스 연결 설정을 최적화하고, 트랜잭션 처리를 강화하여 데이터 일관성을 유지하도록 설계하였다. 또한, H2 데이터베이스를 사용한 테스트 환경에서 실제 운영 환경과 유사한 설정으로 테스트를 반복하여 안정성을 높였다.

마지막으로, 사용자 인터페이스(UI)와 사용자 경험(UX) 문제이다. 초기에는 사진 업로드 과정이 복잡하고 페이지가 혼잡하여 사용자가 혼란을 느낄 만한 부분들이 있었다. 이를 개선하기 위해 업로드 프로세스를 단순화하고, 사용자에게 명확한 지침을 제공하는 UI를 설계했습니다. 업로드 진행 상황을 시각적으로 표시하고, 완료 후 결과 페이지로 자동 이동하도록 하여 사용자의 편의성을 높였다

이를 통해 다양한 문제들을 해결하는 과정에서 많은 것을 배울 수 있었다. 우선, 대용량 파일 업로드와 관련된 문제를 해결하면서 서버의 성능 최적화와 메모리 관리의 중요성을 깊이 깨닫았다. 이를 통해 실무 환경에서 발생할 수 있는 다양한 시나리오에 대비하는 법을 배우게 되었다. 브라우저 호환성 문제를 해결하는 과정에서는 웹 표준의 중요성과 다양한 환경에서의 테스트의 필요성을 실감할 수 있었다. 특히, 최신 기술만을 사용할 때 발생할 수 있는 호환성 문제를 고려하여 코드를 작성하는 법을 배웠다. 데이터베이스 연동 문제를 해결하면서는 데이터의 일관성과 안정성을 유지하는 것이 얼마나 중요한지, 그리고 이를 위해 트랜잭션과 연결 설정을 최적화하는 방법을 학습하게 되었다. 또한, 테스트 환경을 실제 운영 환경과 유사하게 구성하여 반복적인 테스트를 수행하는 것이 시스템의 안정성을 높이는 데 큰 도움이 된다는 것을 깨닫았다. 마지막으로, 사용자 인터페이스와 사용자 경험을 개선하는 과정에서 사용자의 입장에서 생각하는 법을 배웠다. 단순하고 직관적인 인터페이스가 사용자 만족도에 얼마나 큰 영향을 미치는지 알게 되었고, 이를 통해 앞으로의 개발에서 UI/UX를 더욱 신경 쓰게 될 것이다. 이 프로젝트를 통해 얻은 경험과 지식을 바탕으로, 앞으로 더 나은 시스템을 설계하고 개발할 수 있는 능력을 키울 수 있었다.