

# 소프트웨어프로젝트1 Report

## Project #1

담당교수: 이우신 교수님

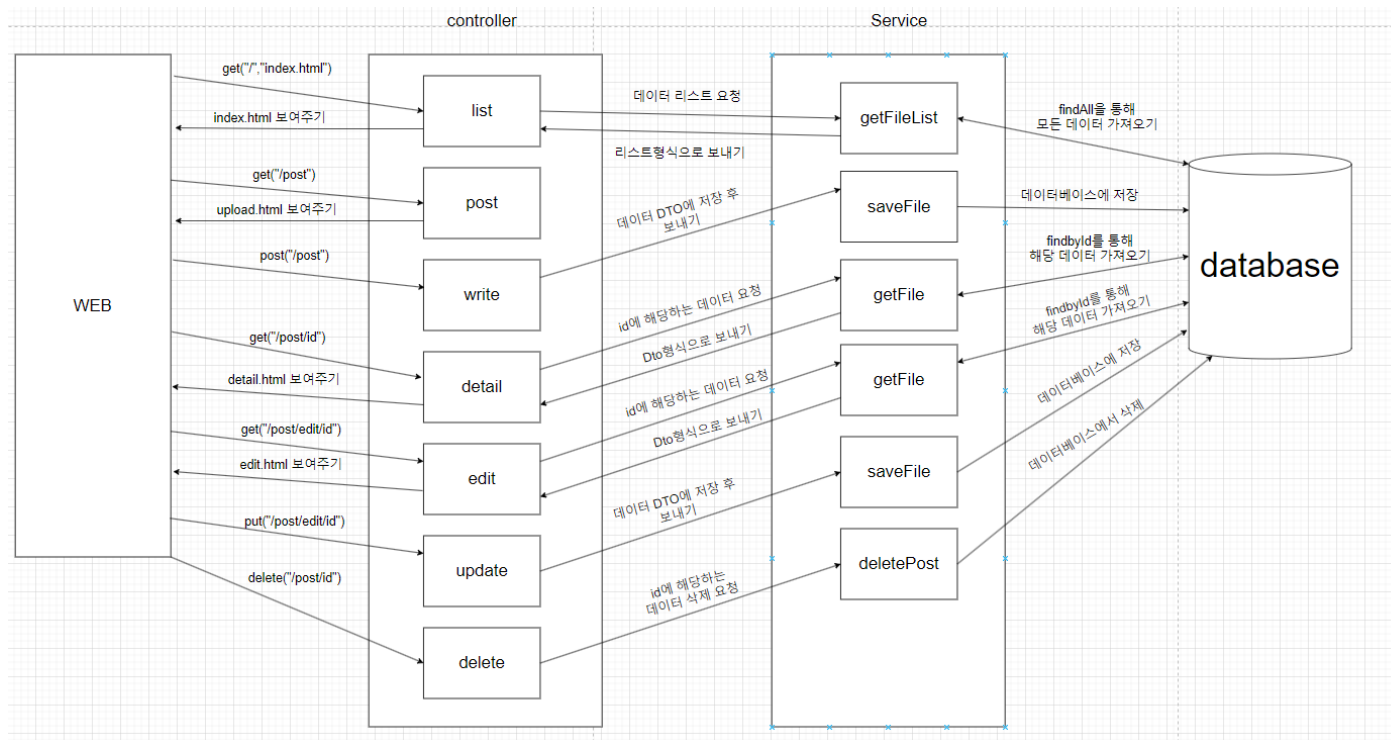
학번: 2019202050

이름: 이강현

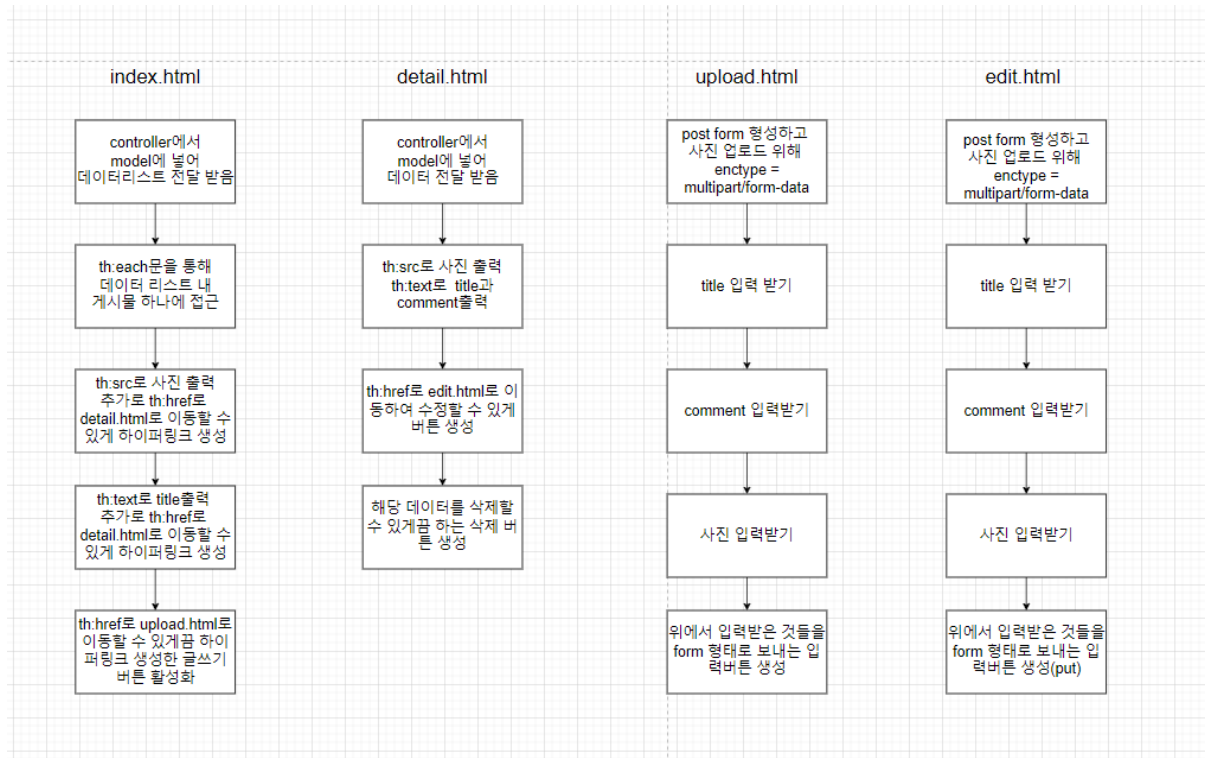
## [Introduction]

이번 프로젝트에서는 사진을 업로드하고 해당 사진에 대한 title과 comment를 작성하면 게시물을 생성하고 이를 홈페이지에서 보여주는 서버를 구현한다. 생성된 게시물은 서버의 database에서 관리되며 게시물은 수정과 삭제가 가능하다. spring boot를 이용하여 서버를 구현하고 h2 database를 사용하여 데이터베이스를 관리한다. MVC 패턴을 이용하여 요청에 따른 controller의 올바른 동작과 데이터베이스에서 올바른 데이터를 뽑아내거나 저장하여 html형식으로 온전히 보여질 수 있게끔 설계한다. html에서는 thymeleaf를 사용하여 데이터를 홈페이지에 보여줄 수 있다.

## [Flow Chart]



전체적인 코드의 흐름은 위와 같다. web에서의 요청을 받아 해석하고 상황에 맞는 응답을 보내는 controller와 controller와 communication하여 필요한 데이터를 controller와 주고받는 Service, 데이터가 저장되는 database로 나누어 서로 어떤 상호작용을 하는지를 보였다.



각 html에 대한 흐름도이다.

index.html은 메인 화면으로 업로드되어 있는 게시물 전체를 보여준다. controller가 Service에게 데이터베이스에 저장된 모든 게시물을 요청하고 해당 데이터리스트를 받아 html내에서 thymeleaf를 이용한 반복문으로 모든 데이터에 접근하여 홈페이지에 뿌려준다.

detail.html은 index.html과 흐름도가 비슷하나 게시물 하나만 자세히 보여주면 되므로 요청받은 해당 데이터만을 가져와 보여준다.

upload.html과 edit.html은 폼을 형성하여 title과 comment 사진을 입력받고 이를 controller에게 post요청과 함께 보낸다. edit.html은 put요청으로 보내게 된다.

## [Result]

MVC 패턴에 입각하여 설명한다면

Model은 데이터가 저장되는 데이터베이스와 비즈니스 로직을 관리하는 부분이다. 프로젝트 내 코드에서 service 패키지 내 FileService와 domain.repository패키지 내 FileRepository가 이에 해당한다. FileService에는 데이터베이스에서 데이터를 한번에 가져올 수 있는 getFileList함수, 특정 id에 해당하는 데이터를 가져오는 getFile함수, 특정 id의 데이터를 지울 수 있는 deletePost함수, 데이터를 저장하는 saveFile함수가 존재한다. Repository는 JpaRepository를 이용한 interface로 구현하였다.

View는 실질적으로 사용자에게 보여지는 html파일들이 이에 해당한다.

html의 흐름도는 위에서 설명했으니 세부 구현방식을 설명한다.

index.html에서 데이터를 사용하는 방법은 controller에서 model.addAttribute를 통해 model에 게시물 전체 리스트를 postList라는 이름으로 html에서 사용할 수 있게끔 넣어두고 이를 thymeleaf문법을 이용하여 구현했다. th:each문을 통해 \${postList}를 이용해 불러온 리스트에서 post단위로 뽑아 \${post.title} ,@\${post.origFilename}}과 같이 화면에 보여주거나 사진 경로를 달아야 할 때 사용하였다.

detail.html은 리스트가 아닌 특정 데이터 하나만을 사용하므로 위와 비슷하지만 반복문을 사용하지 않았고 추가적으로 comment만 더 보여주고 수정 및 삭제 버튼을 활성화해야 하므로 th:href에 @{/post/" + \${post.id}} 이런 식으로 경로를 형성하기 위해서는 데이터의 id가 필요하므로 이를 thymeleaf문법을 통해 구현했다.

upload.html과 edit.html은 폼을 형성하여 업로드할 데이터를 입력받아야 하므로 html 문법을 통해 이를 구성하고 name = title,comment...형식으로 필요한 정보들을 구분하여 입력받았다. 입력은 커다란 form내에서 받도록 구현했으며 form의 마지막에는 데이터를 controller에 post 요청과 함께 전달되도록 하는 버튼을 구현했다.

Controller는 web에서 요청에 대한 응답으로 모델 및 뷰를 업데이트하는 로직을 말한다.

getmapping으로 어노테이션하여 get요청을 받아들이고 데이터를 보여준다.

postmapping으로 어노테이션하여 post요청을 받아들이고 입력받은 데이터로 데이터베이스를 업데이트한다.

putmapping으로 어노테이션하여 기존 데이터베이스의 데이터를 수정한다.

deletemapping으로 어노테이션하여 데이터베이스의 데이터를 삭제한다.

다양한 요청에 따라 응답이 다르며 과제 요구조건에 맞게 타이틀의 길이제한, 이미지 파일만 받기, 요청 응답 후 redirect와 같은 행동들은 주로 이곳에서 정의된다.

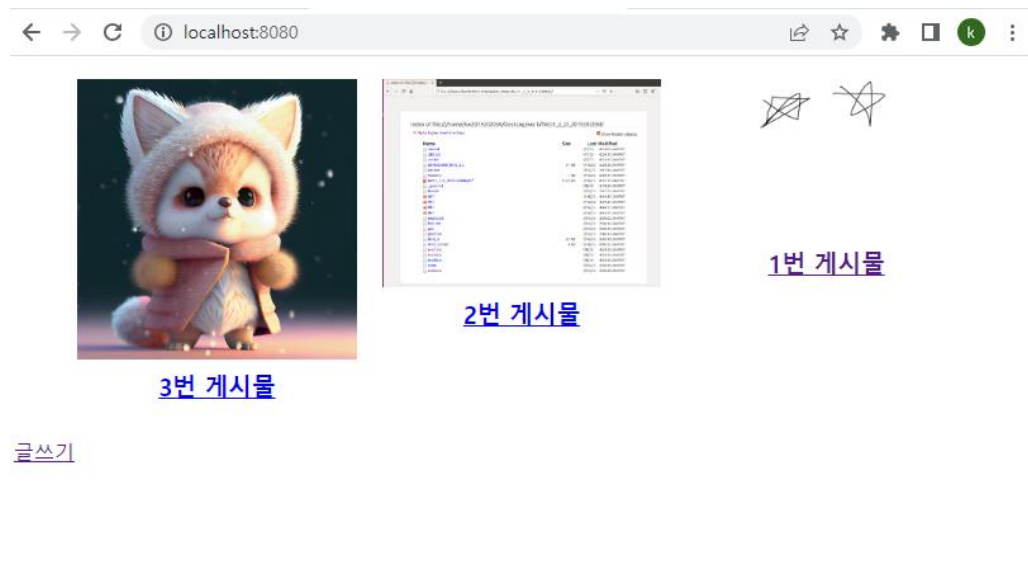
다음은 각 html의 홈페이지를 결과로 보여주고 과제 요구사항을 점검한다.

title이 20자를 넘거나 comment가 100자가 넘거나 처음 업로드시 사진이 존재하지 않으면 업로드를 하지 않고 다시 index.html로 redirect되는 것으로 처리하였다.

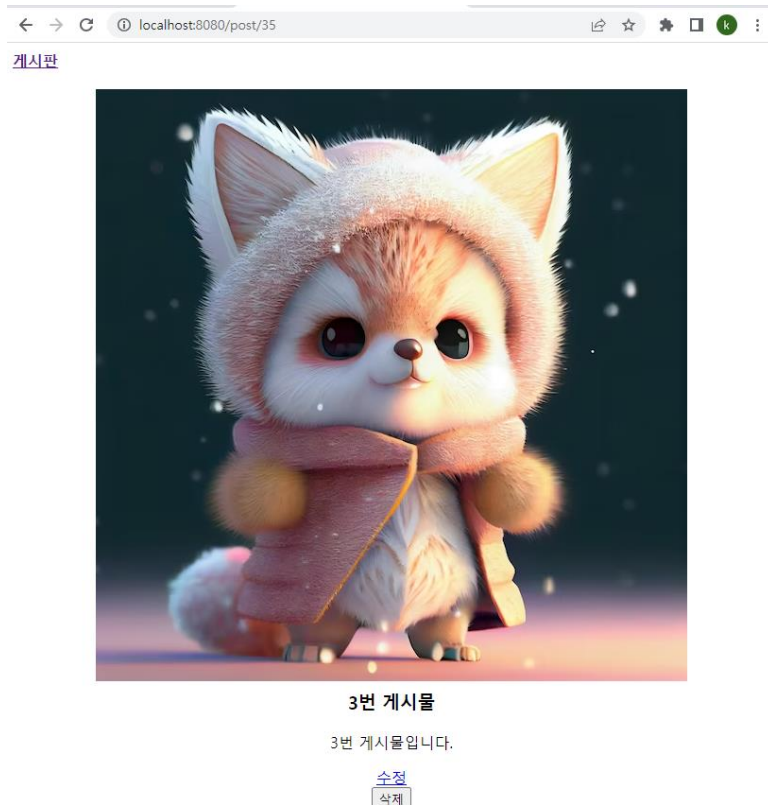
수정 시에는 title,comment,사진 업로드가 선택이며 이때는 정보를 수정하지 않으면

기존 정보가 유지되게끔 구현하였다.

또한 사진을 업로드하게 되면 static폴더로 업로드한 파일을 저장하고 이를 html에서 사진 경로로 넣어주기 때문에 업로드 후 서버를 재시작하면 사진이 페이지에 정상적으로 로드됨을 명시한다.

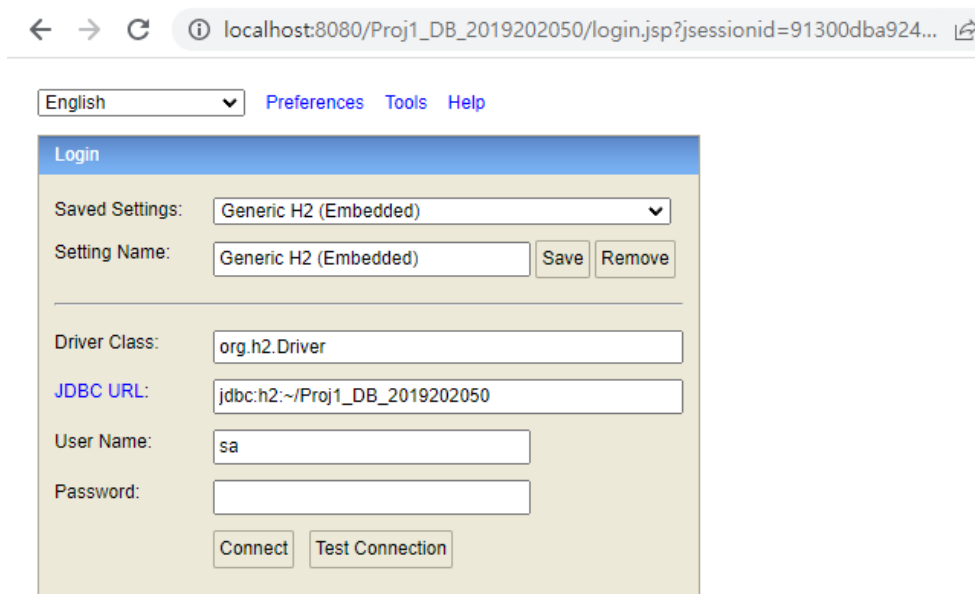


index.html의 모습이다. 가장 최근에 업로드한 게시물이 먼저 나오도록 정렬되었다. 사진 혹은 title을 클릭하면 detail.html이 보여지며 해당게시물의 comment가 추가로 보여지고 수정 및 삭제가 가능하다.



3번 게시물을 클릭하였을 때 위와 같은 결과를 확인할 수 있다.

localhost:8080/Proj1\_DB\_2019202050으로 접속하면 아래와 같은 페이지로 접속되어 database를 확인할 수 있다.



connect를 누르면 아래와 같이 저장된 정보를 테이블로 확인할 수 있다.

jpg,jpeg,png와 같은 사진만을 저장한다. 그 외의 파일을 저장할 시 게시물이 업로드 되지 않고 index.html로 redirect한다.

Run

Run Selected

Auto complete

Clear

SQL statement:

SELECT \* FROM FILE

아래는 업로드시 보여지는 화면이다. 제목, 내용, 사진을 첨부할 수 있다.

← → ↺

localhost:8080/post

🔗 ☆ ⚙️ □ k ⋮

게시판

제목

내용

첨부 파일

파일 선택


선택된 파일 없음

글쓰기


← → ↺

localhost:8080/index.html


🔗 ☆ ⚙️ □ k ⋮



3번 게시물



2번 게시물



1번 게시물

글쓰기

위와 같이 localhost:8080/index.html로 접속하여도 localhost:8080/로 접속한 것과 동일한 페이지를 보여주게끔 다중매핑하였다.

localhost:8080/post/edit/35

☆

k

게시판

제목

3번 게시물

내용

첨부 파일


파일 선택

선택된 파일 없음

수정

수정시에는 게시물 업로드양식과 동일한 모습을 보인다.

게시판



3번 게시물

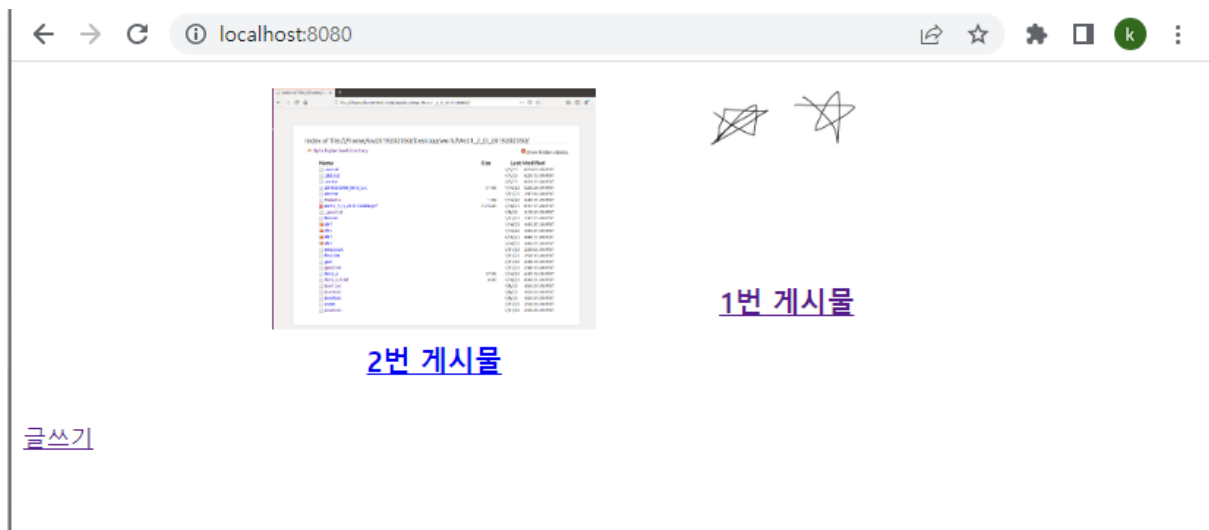
3번 게시물 수정하였습니다.

수정

삭제

사진과 제목을 유지한채 comment만 변경시 기존 사진과 제목을 유지한다.





3번 게시물을 삭제한 모습이다. index.html에서 3번 게시물이 삭제되고 아래와 같이 database내에서도 삭제된다.

Run

Run Selected

Auto complete

Clear

SQL statement:

SELECT \* FROM FILE

SELECT \* FROM FILE:

ID	CONTENT	CREATED_DATE	FILE_PATH	MODIFIED_DATE	ORIG_FILENAME	TITLE
33	1번 게시물입니다.	2023-06-03 23:09:45.553094	C:\Users\USER-PC\바탕 화면\Proj1_2019202050\src\main\resources\static\good.jpg	2023-06-03 23:09:45.553094	good.jpg	1번 게시물
34	2번 게시물입니다.	2023-06-03 23:10:01.594346	C:\Users\USER-PC\바탕 화면\Proj1_2019202050\src\main\resources\static\KakaoTalk_20230416_224732097.png	2023-06-03 23:10:01.594346	KakaoTalk_20230416_224732097.png	2번 게시물

(2 rows, 0 ms)

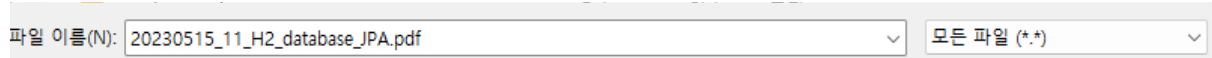
## <Consideration>

이번 프로젝트를 진행하면서 단순한 게시판을 만드는 프로젝트였으나 구현이 정말 어려웠다. 실행 흐름과 html구성은 비교적 쉬웠으나 스프링부트라는 툴을 이용하는 것이 정말 낯설었다. 또한 thymeleaf를 이용하는 부분에서 th:src에 경로를 넘겨주는 것에서 어려움을 많이 겪었는데 th:src에 절대경로를 넣어줘보고 html파일이 있는 부분을 기준으로 ../을 사용하여 static폴더에 접근하도록 넣어봤지만 사진이 뜨지 않았다. 하지만 예시로 123.jpg라는 사진을 사진이름 그대로 th:src에 넣어주면 localhost:8080/123.jpg로 들어가고 이는 결과적으로 ../static/123.jpg의 사진경로로 바뀐다는 것을 알고 나서 이를 해결했다. 또한 이것은 index.html에서 사진 경로를 넣을 때 잘 작동했으나 detail.html에서는 사진이 안 나오는 문제점을 알았다. localhost:8080/post/123.jpg와 같이 들어가기 때문에 th:src에 ../123.jpg와 같이 넣어주어야 사진을 읽을 수 있었다. 이는 chrome의 f12를 눌러 오류메시지를 잘 보고 사진을 어떻게 요청하는지를 분석하여 알게되었다. 또한 lombok을 사용하면서도 오류가 발생했는데 코드를 구현하던 중 controller에서 Dto를 사용하는데 문제가 발생하였다. 이에 대한 오류의 이유는 제대로 분석하진 못했지만

```
compileOnly 'org.projectlombok:lombok:1.18.10'
```

위와 같이 의존성을 추가하여 해결하였다.

또한 사진파일만을 입력받고 jpg,jpeg,png와 같은 타입만 받게끔 하는 요구조건을 수행할때 input 태그의 accept만을 사용했었는데



형식을 모든 파일로 맞춘 후 업로드하면 이미지파일이 아닌 파일도 업로드가 가능했다. 따라서 controller에서

```
private boolean isImageExtension(String extension)//extension을 인자로 받고
{
    List<String> imageExtensions = Arrays.asList("jpg", "jpeg", "png");
    return imageExtensions.contains(extension.toLowerCase());//위의 확장자 리스
}
```

위와 같은 함수를 정의하여 파일 유형 검사를 진행하는 것으로 해결하였다.

```
@Column(length = 20, nullable = false)//필수 입력 조건임
private String title;
```

위와 같이 title의 length를 설정하면 title의 값을 20으로 한정하여 받는 줄 알고 코드를 구현했으나 해당 Column의 length기능은 자동 ddl을 생성할 때만 사용된다는 것을 구글링을 통해 알게 되고

```
if(fileDto.getTitleLength()>20)
{
    return "redirect:/";
}
if(fileDto.getCommentLength()>100)
{
    return "redirect:/";
}
```

```
public int getTitleLength() { return title.length(); }//
2 usages
public int getCommentLength() {return content.length();}
```

사진과 같이 controller에서 예외를 처리함으로 해결하였다.

정말 많은 오류와 어려움이 있었고 초기에는 어노테이션의 의미나 model을 통한 데이터 전달과 같은 것을 알지 못해 막막했다. 하지만 구글링과 강의자료를 참고해가며 하나씩 해결해보며 스프링의 편리함을 느낄 수 있었던 좋은 예제였던 것 같다.