JWT 토큰 생성 및 검사

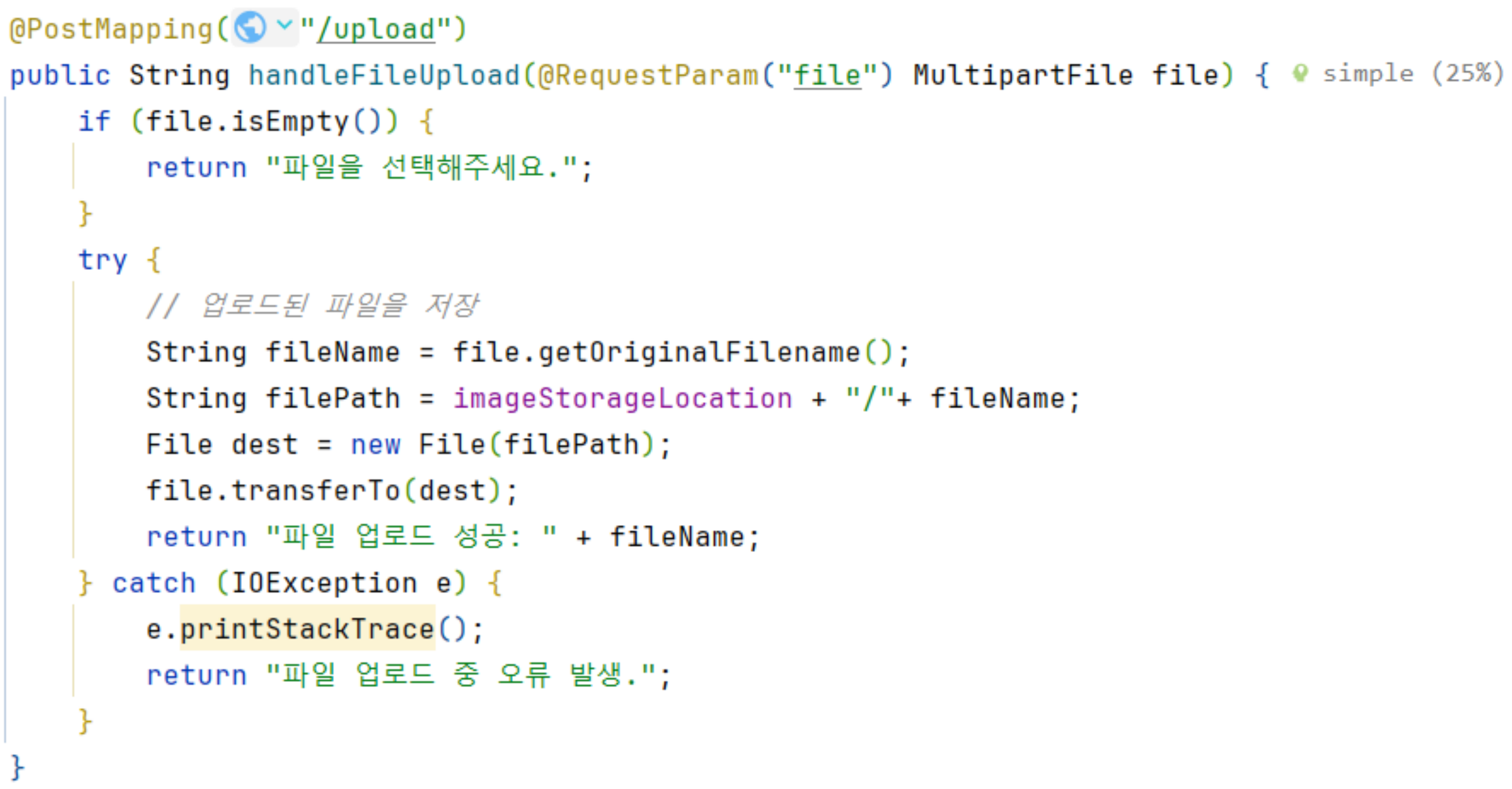
Springboot 파일업로드

https://github.com/dron512/git231229/tree/main/springbootApi_work/restapi05

박명회

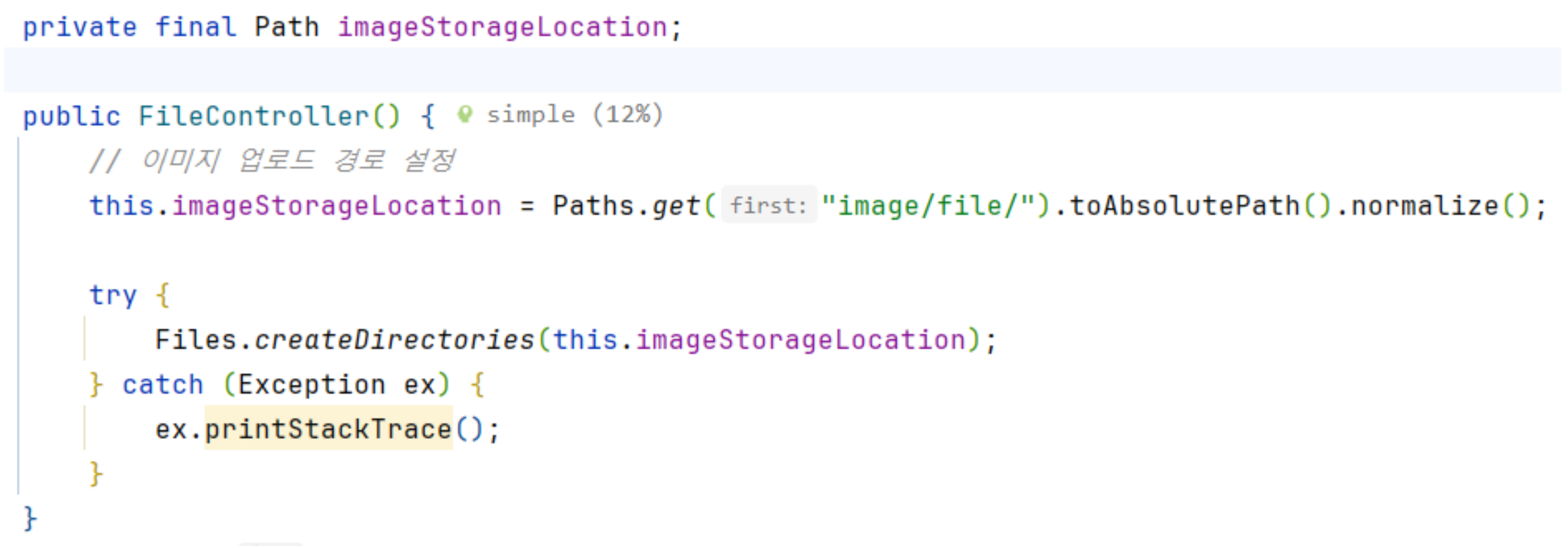
백엔드
스프링부트

1.1 파일 업로드 PostMapping



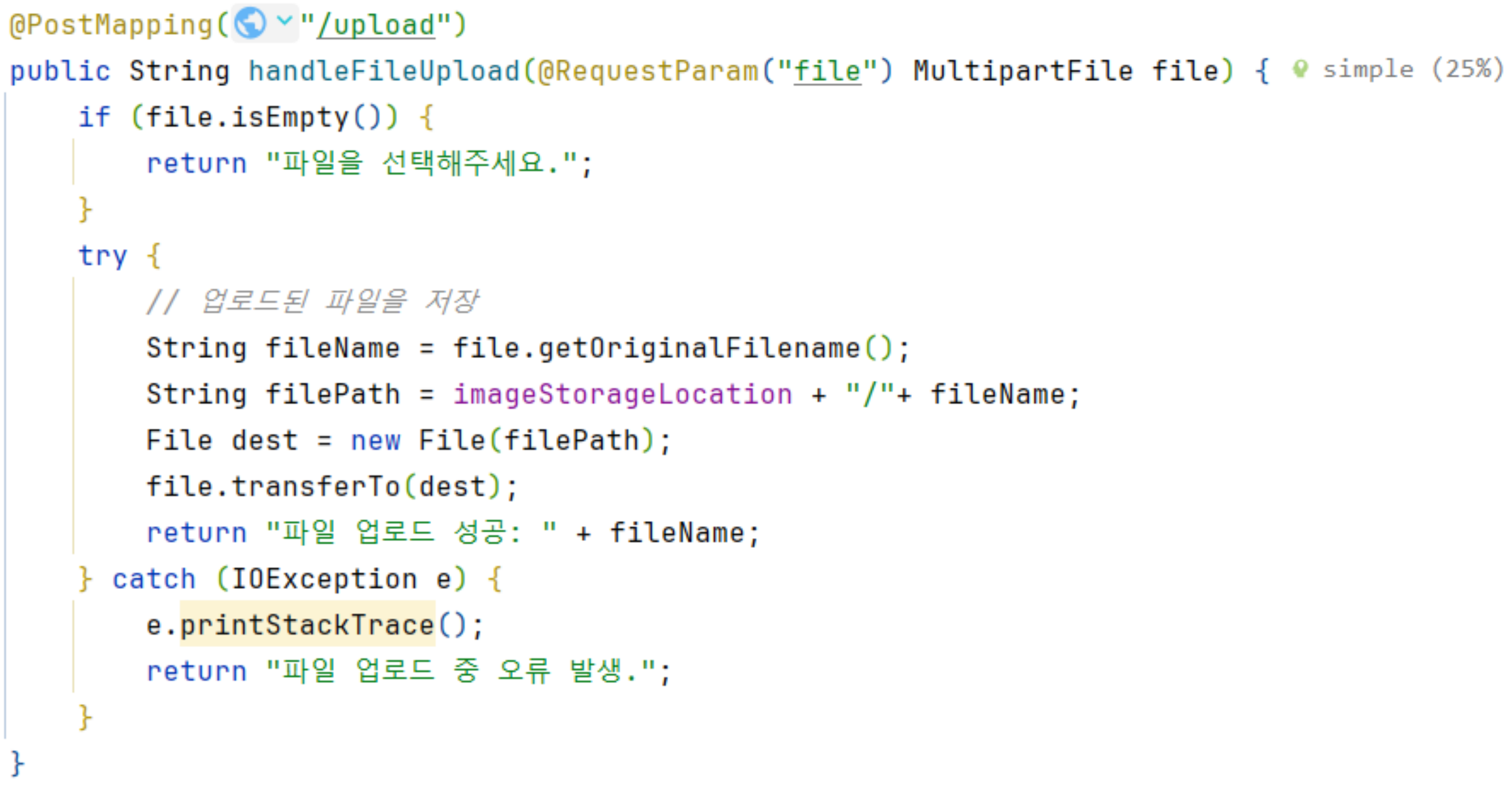
```java
@PostMapping("/upload")
public String handleFileUpload(@RequestParam("file") MultipartFile file) {  simple (25%)
    if (file.isEmpty()) {
        return "파일을 선택해주세요.";
    }
    try {
        // 업로드된 파일을 저장
        String fileName = file.getOriginalFilename();
        String filePath = imageStorageLocation + "/"+ fileName;
        File dest = new File(filePath);
        file.transferTo(dest);
        return "파일 업로드 성공: " + fileName;
    } catch (IOException e) {
        e.printStackTrace();
        return "파일 업로드 중 오류 발생.";
    }
}
```

1.1 파일 업로드 경로지정 및 경로생성



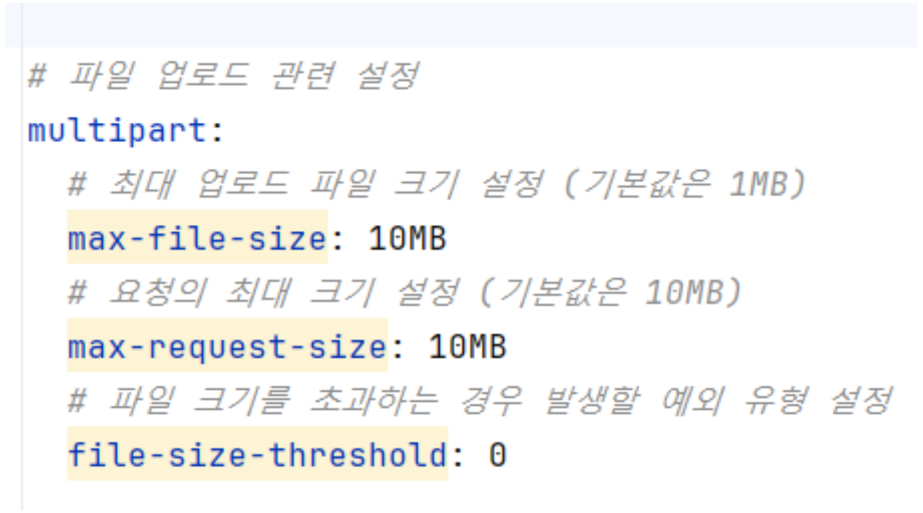
```java
private final Path imageStorageLocation;

public FileController() {  simple (12%)
    // 이미지 업로드 경로 설정
    this.imageStorageLocation = Paths.get( first: "image/file/").toAbsolutePath().normalize();

    try {
        Files.createDirectories(this.imageStorageLocation);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
```

1.1 파일 업로드



```
@PostMapping("/upload")
public String handleFileUpload(@RequestParam("file") MultipartFile file) {  simple (25%)
    if (file.isEmpty()) {
        return "파일을 선택해주세요.";
    }
    try {
        // 업로드된 파일을 저장
        String fileName = file.getOriginalFilename();
        String filePath = imageStorageLocation + "/"+ fileName;
        File dest = new File(filePath);
        file.transferTo(dest);
        return "파일 업로드 성공: " + fileName;
    } catch (IOException e) {
        e.printStackTrace();
        return "파일 업로드 중 오류 발생.";
    }
}
```

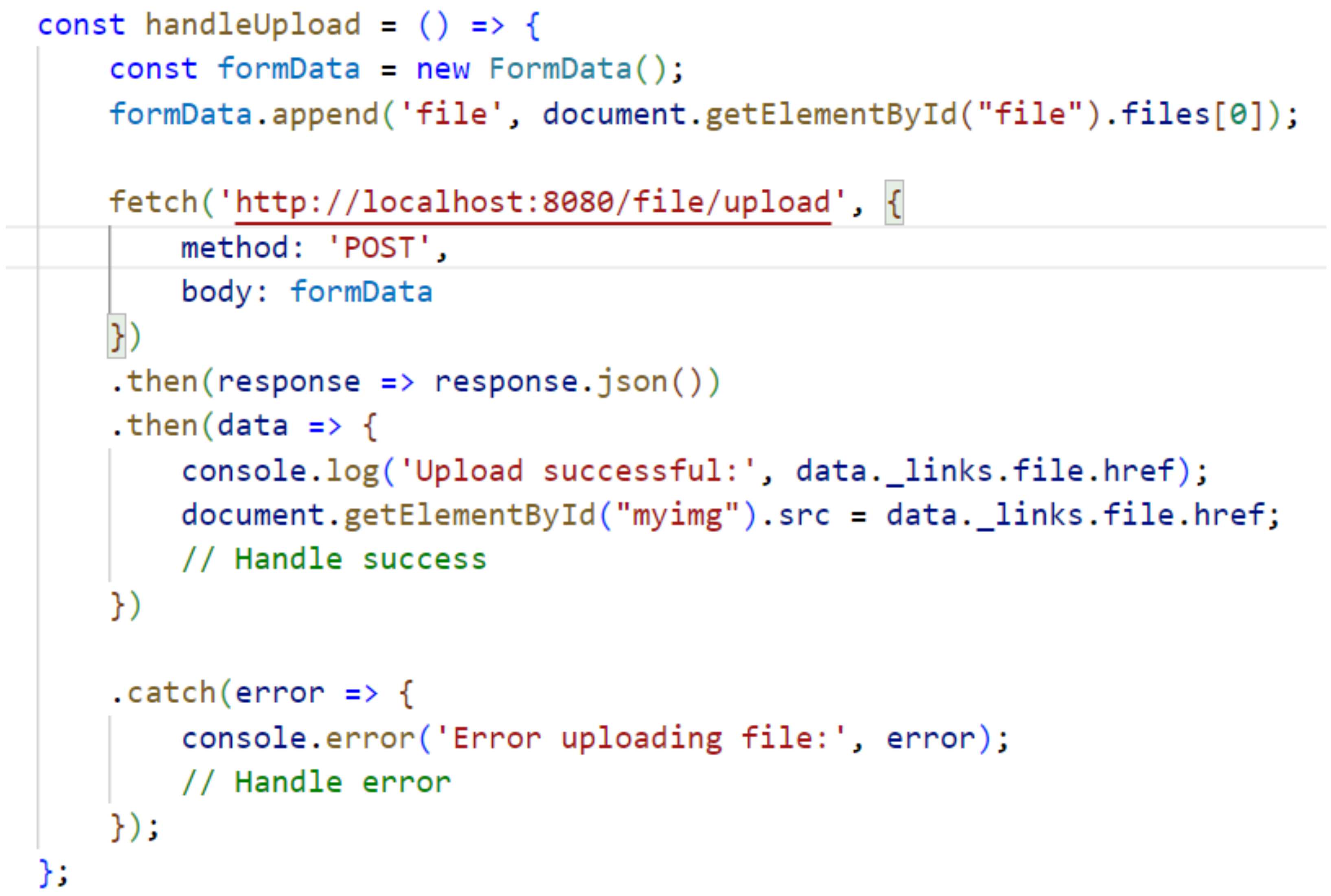
1.1 Application.yml 설정



```yaml
# 파일 업로드 관련 설정
multipart:
  # 최대 업로드 파일 크기 설정 (기본값은 1MB)
  max-file-size: 10MB
  # 요청의 최대 크기 설정 (기본값은 10MB)
  max-request-size: 10MB
  # 파일 크기를 초과하는 경우 발생할 예외 유형 설정
  file-size-threshold: 0
```

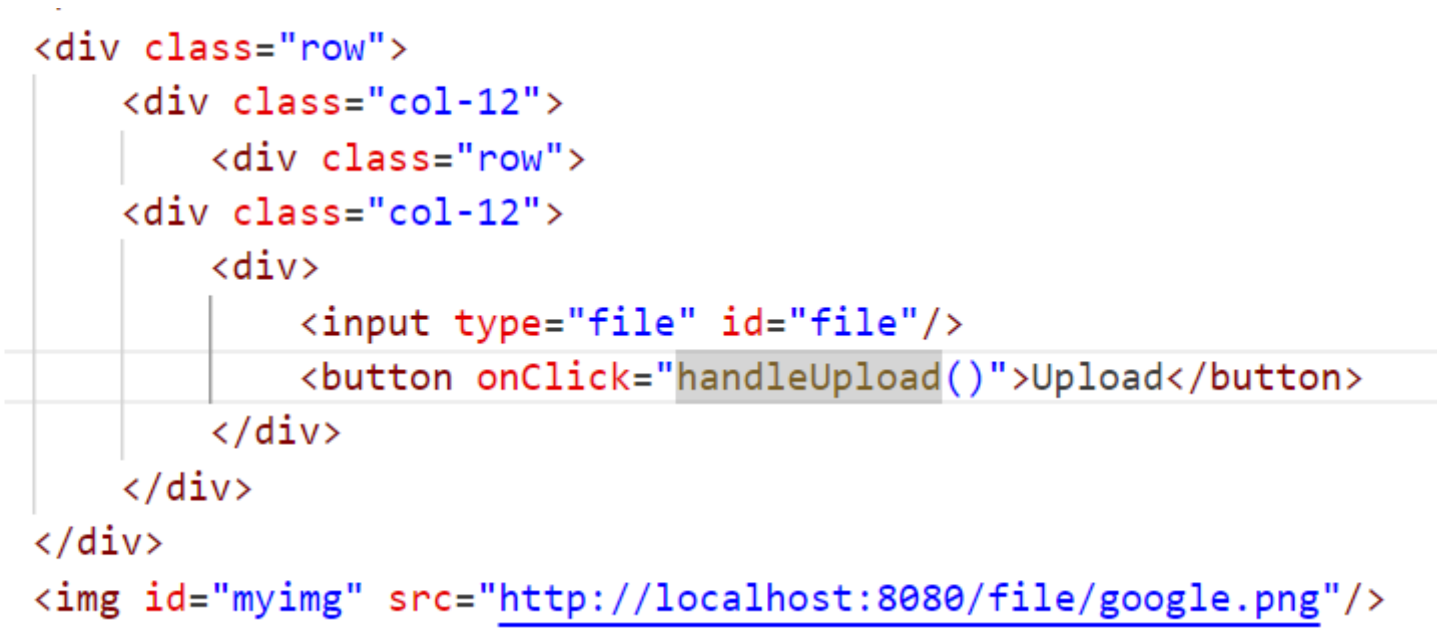
프론트
자바스크립트

1.1 javascript 함수



```
const handleUpload = () => {
    const formData = new FormData();
    formData.append('file', document.getElementById("file").files[0]);

    fetch('http://localhost:8080/file/upload', {
        method: 'POST',
        body: formData
    })
    .then(response => response.json())
    .then(data => {
        console.log('Upload successful:', data._links.file.href);
        document.getElementById("myimg").src = data._links.file.href;
        // Handle success
    })

    .catch(error => {
        console.error('Error uploading file:', error);
        // Handle error
    });
};
```

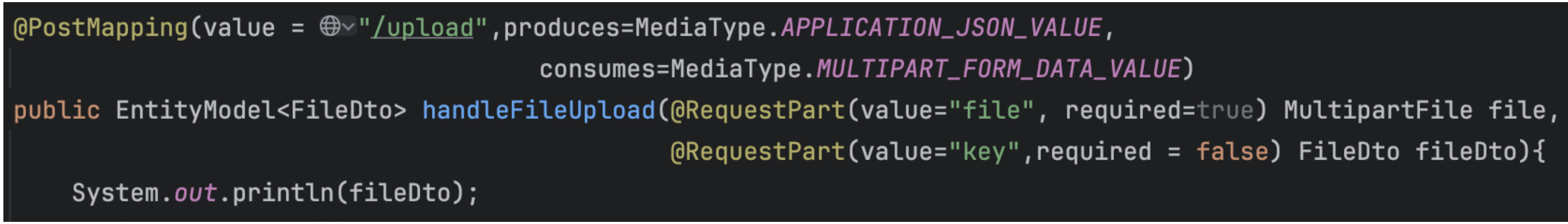
1.1 html 태그



```
<div class="row">
    <div class="col-12">
        <div class="row">
    <div class="col-12">
        <div>
            <input type="file" id="file"/>
            <button onClick="handleUpload()">Upload</button>
        </div>
    </div>
</div>
<img id="myimg" src="http://localhost:8080/file/google.png"/>
```

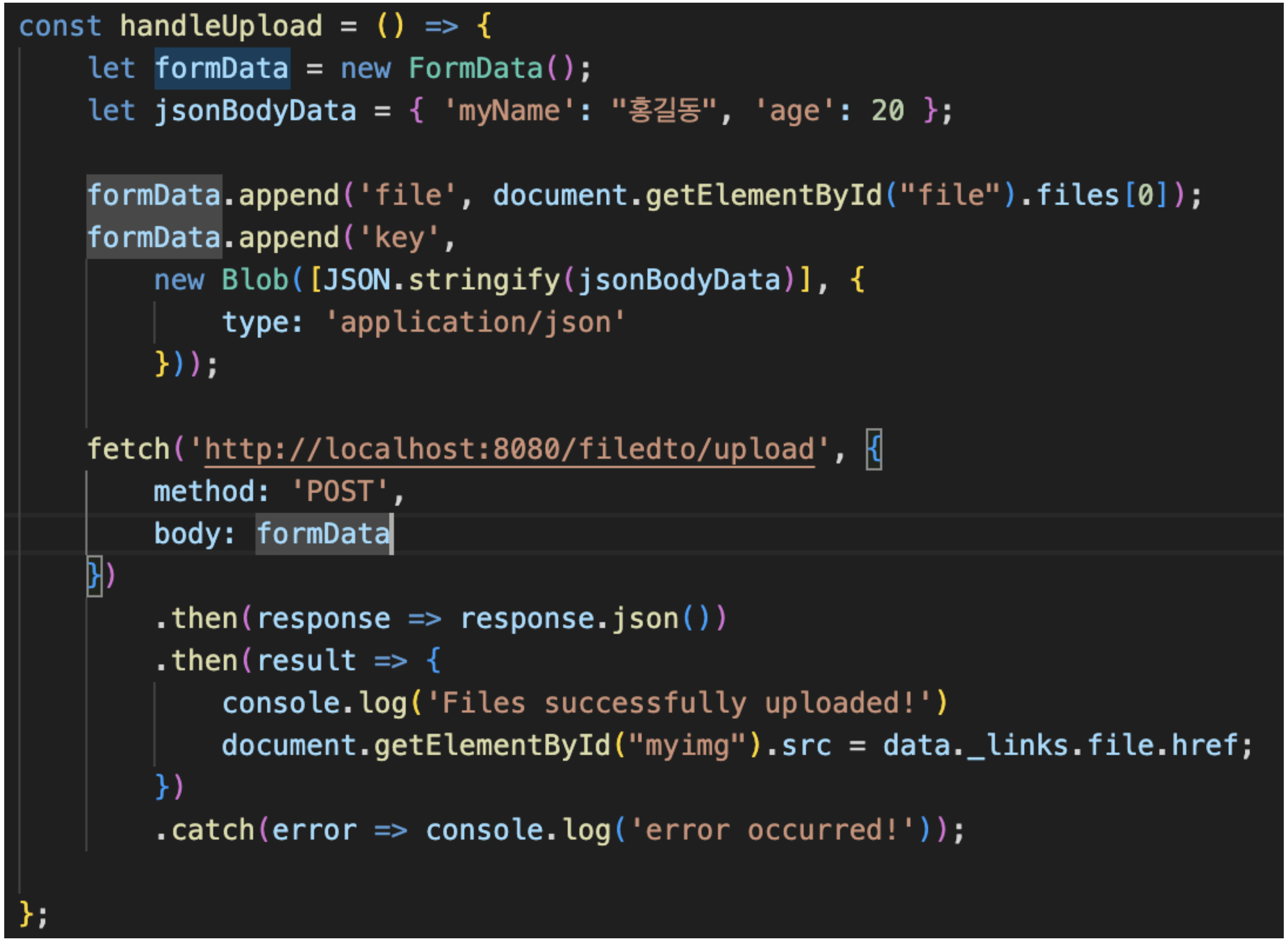
파일, DTO 보내기

1.1 백엔드



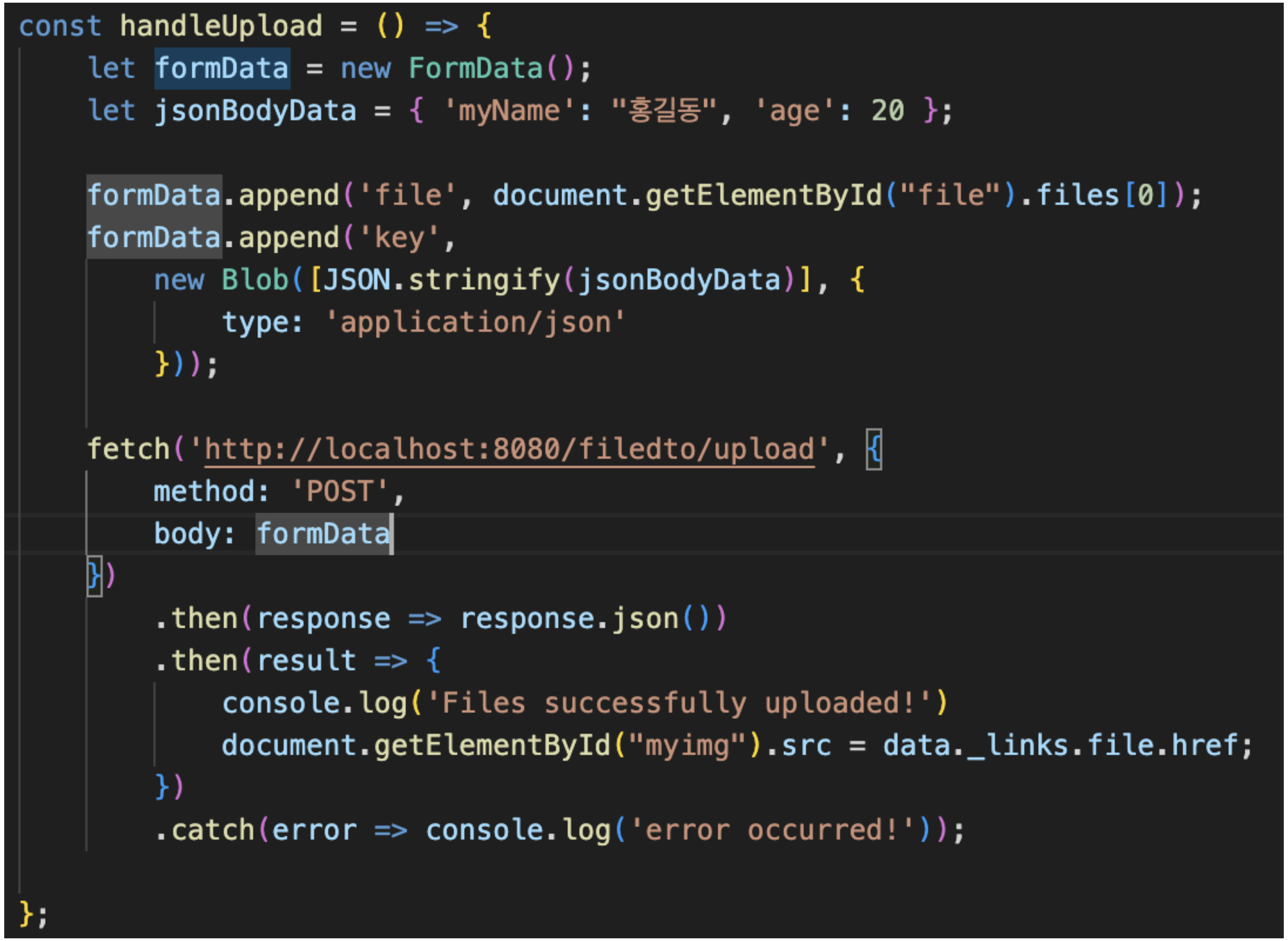
```
@PostMapping(value = "/upload",produces=MediaType.APPLICATION_JSON_VALUE,
                                    consumes=MediaType.MULTIPART_FORM_DATA_VALUE)
public EntityModel<FileDto> handleFileUpload(@RequestPart(value="file", required=true) MultipartFile file,
                                             @RequestPart(value="key",required = false) FileDto fileDto){
    System.out.println(fileDto);
```

1.1 프론트



```
const handleUpload = () => {
    let formData = new FormData();
    let jsonBodyData = { 'myName': "홍길동", 'age': 20 };

    formData.append('file', document.getElementById("file").files[0]);
    formData.append('key',
        new Blob([JSON.stringify(jsonBodyData)], {
            type: 'application/json'
        }));

    fetch('http://localhost:8080/filedto/upload', {
        method: 'POST',
        body: formData
    })
        .then(response => response.json())
        .then(result => {
            console.log('Files successfully uploaded!')
            document.getElementById("myimg").src = data._links.file.href;
        })
        .catch(error => console.log('error occurred!'));

};
```

1.1 프론트



```
const handleUpload = () => {
    let formData = new FormData();
    let jsonBodyData = { 'myName': "홍길동", 'age': 20 };

    formData.append('file', document.getElementById("file").files[0]);
    formData.append('key',
        new Blob([JSON.stringify(jsonBodyData)], {
            type: 'application/json'
        }));

    fetch('http://localhost:8080/filedto/upload', {
        method: 'POST',
        body: formData
    })
        .then(response => response.json())
        .then(result => {
            console.log('Files successfully uploaded!')
            document.getElementById("myimg").src = data._links.file.href;
        })
        .catch(error => console.log('error occurred!'));

};
```

1.1 참고사이트

https://satyajitpatnaik.medium.com/multipartfile-uploads-from-a-reactjs-ui-to-a-spring-boot-service-fdaeef9743dc

1.1 여러 파일 보내기

```
@RestController
@CrossOrigin
@SuppressWarnings({ "unchecked", "rawtypes" })
@RequestMapping(value = "/api/service")
public class FileUploadController {

@ApiOperation(value = "Upload of multiple files",
    response = Iterable.class)
  @ApiResponses(value = {
    @ApiResponse(code = 200,
      message = "Successfully uploaded multiple files"),
    @ApiResponse(code = 404, message = "Not found.")
   })
  @RequestMapping(value="/uploadfiles",
     method=RequestMethod.POST,
     produces=MediaType.APPLICATION_JSON_VALUE,
     consumes=MediaType.MULTIPART_FORM_DATA_VALUE)
  public ResponseEntity uploadFiles(
    @RequestPart("jsonBodyData") SomeModel dataReqData,
    @RequestPart("files") MultipartFile[] files) throws IOException{
      JSONObject response = new JSONObject();
      LOGGER.info("JSON Request Body Data: {}", dataReqData);
      for(MultipartFile file : files) {
        LOGGER.info(file.getOriginalFilename());
      }
      return new ResponseEntity<>(response, HttpStatus.OK);
  }
}
```

1.1 여러 파일 보내기

```
uploadJSONFiles(event) {
  event.preventDefault();
  let formData = new FormData();
  let jsonBodyData = { 'someKey': 'someValue' };
  for(let key of Object.keys(event.target.files)) {
    if (key !== 'length') {
      formData.append('files', event.target.files[key]);
    }
  }
  formData.append('jsonBodyData',
    new Blob([JSON.stringify(jsonBodyData)], {
      type: 'application/json'
    }));

  fetch('/api/service/uploadfiles', {
    method: 'POST',
    body: formData
  }).then(response => response.json())
  .then(result => console.log('Files successfully uploaded!'))
  .catch(error => console.log('error occurred!'));
}

render() {
  <div className="uk-margin-medium-top">
    <label>Upload Files</label>
    <input type="file"
      onChange={(event) => this.uploadJSONFiles(event)}
      multiple/>
 </div>
}
```

고맙습니다.