

K-Digital Training 웹 풀스택 과정

파일업로드

목차

- Multer 란
- 일반 폼 태그를 이용한 파일 업로드
- Postman으로 간단히 실습하기
- Multer 문법 (single(), array(), fields())
- FormData

파일 업로드 Multer 미들웨어

이미지, 동영상 등을 비롯한 여러 가지 파일들을 **멀티파트 형식**으로 업로드할 때 사용하는 미들웨어이다.

멀티파트 형식이란 enctype이 multipart/form-data 인 폼을 통해 업로드 하는 데이터 형식을 말한다

일반 폼 태그를 이용한 파일 업로드

- 클라이언트에서 서버로 파일 전송하는 법
 - input 태그, type = "file"로 지정
 - name 속성은 서버에서 파일을 인식할 이름이 됨 (서버와 동일하게 설정하기)

```
<input type="file" name="userfile">
```

파일 선택

선택된 파일 없음

클라이언트 준비 - 일반 폼 전송

```
<form action="/upload" method="POST" enctype="multipart/form-data">  
  <input type="file" name="userfile" /><br />  
  <input type="text" name="title" /><br /><br />  
  <button type="submit">업로드</button>  
</form>
```

- form 태그의 enctype 속성으로 "multipart/form-data" 반드시 설정

주: Multer는 multipart (multipart/form-data)가 아닌 폼에서는 동작하지 않습니다.

출처: multer 공식 문서

multer 미들웨어 – 서버

- 파일 업로드를 위해 사용되는 미들웨어
- express로 서버를 구축할 때 가장 많이 사용되는 미들웨어

```
npm install multer
```

multer 설치하기

```
const multer = require( 'multer' );
```

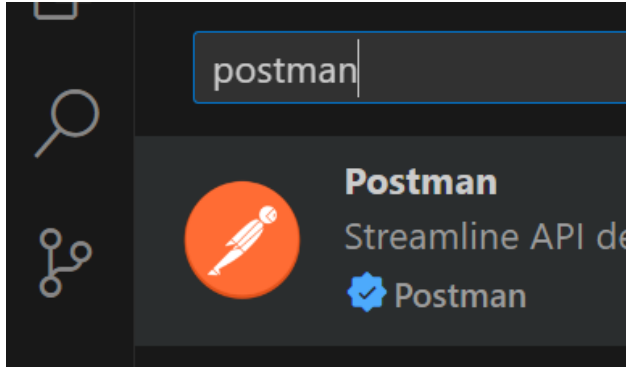
app.js에 multer 불러오기

Multer 문법

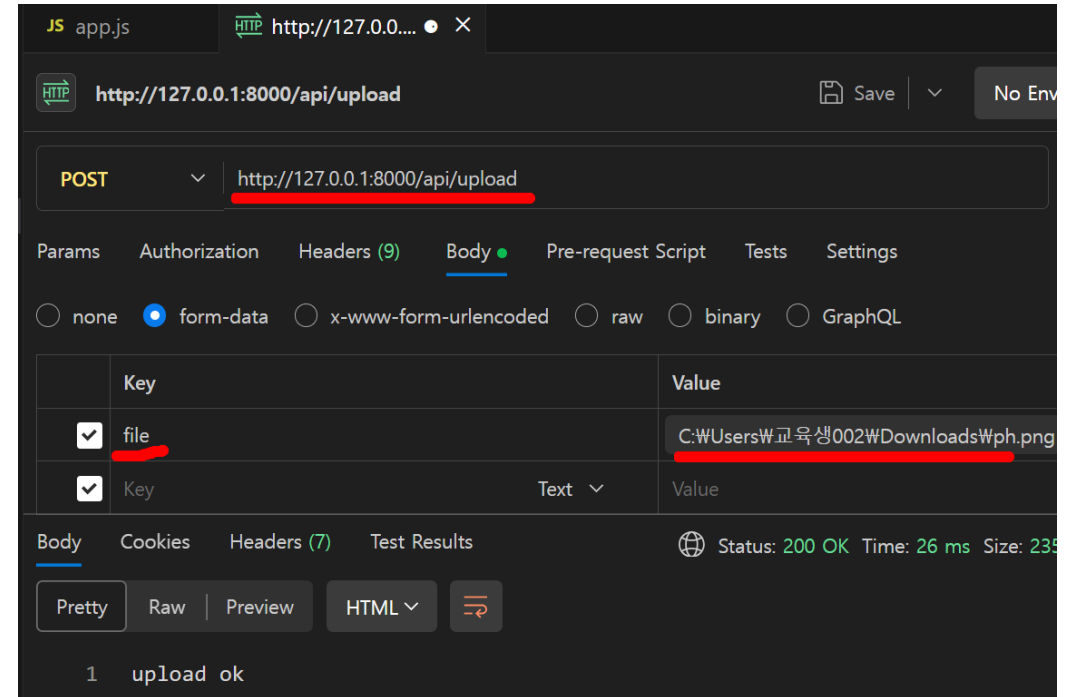
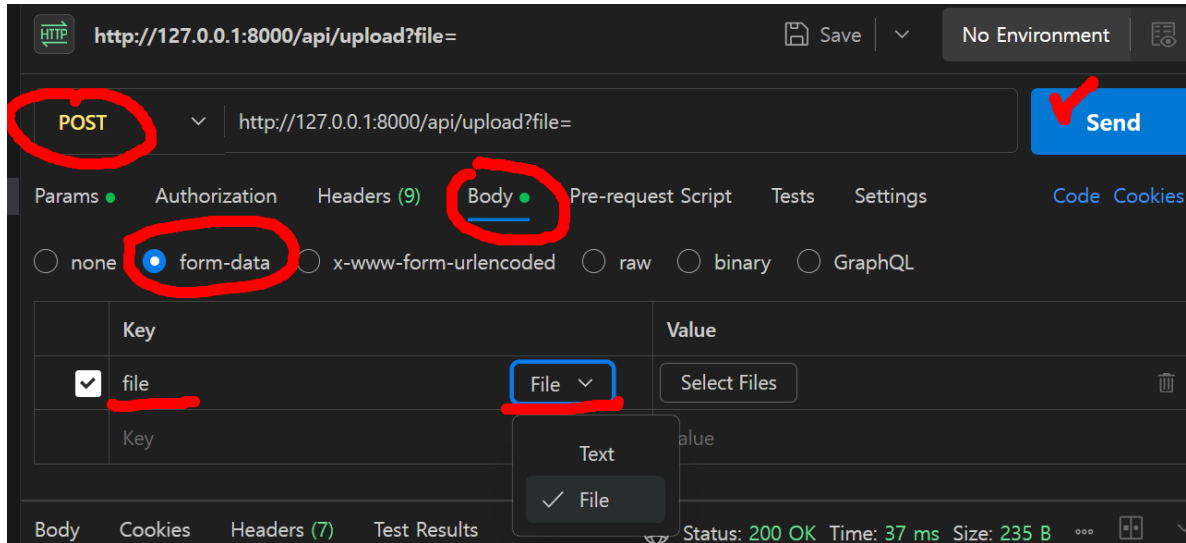
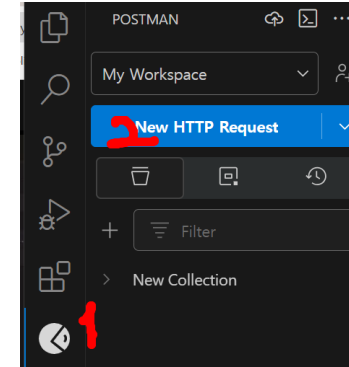
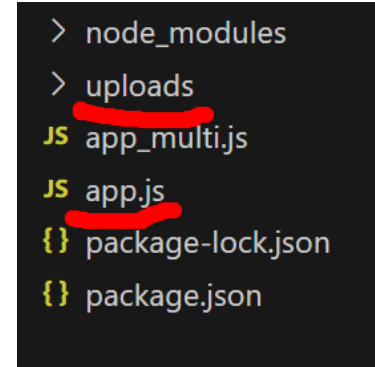
- storage는 저장할 공간에 대한 정보. 디스크나 메모리 저장 가능.
- diskStorage는 하드디스크에 업로드 파일을 저장한다는 것
- destination은 저장할 경로
- filename은 저장할 파일명(파일명+날짜+확장자 형식)
- Limits는 파일 개수나 파일 사이즈를 제한할 수 있다.

- `single()` : 1개의 파일만 업로드 할때
 - `single('file')` 과 `<input type="file" name="file">` 일치
 - 파일이 업로드 된 후 `req.file` 객체생성, 업로드된 결과는 `req.file`객체안에 들어간다.
 - `req.body` : 나머지 정보
- `array()` : 하나의 요청에 여러 개의 파일을 업로드할때
 - 업로드된 정보들은 배열로 저장된다.
 - `req.files` : 파일 n개
- `.field()` : 여러 개의 요청에 여러 개의 파일을 따로 업로드 할 때
 - `field`미들웨어를 사용해야 한다. 별개로 취급되기 때문에 `input`태그에 `name`속성을 각각 지정해야 한다.

Postman으로 간단히 실습하기



설치해야 할 모듈:
npm install Express
npm install --save multer
Postman 로그인 필요
패킷 받아서 등록



파일 업로드 경로 설정

```
const multer = require('multer');  
const upload = multer({  
  dest: 'uploads/',  
});
```

app.js

- dest : 파일을 업로드하고 그 파일이 저장될 경로를 지정하는 속성

multer – 하나의 파일 업로드

- **single()** : 하나의 파일 업로드

```
<form action="/upload" method="POST" enctype="multipart/form-data">
  <input type="file" name="userfile" /><br />
  <input type="text" name="title" /><br /><br />
  <button type="submit">업로드</button>
</form>
```

index.ejs

```
const multer = require('multer');
const upload = multer({
  dest: 'uploads/',
});

app.post('/upload', upload.single('userfile'), function (req, res) {
  console.log(req.file); // req.file: 파일 업로드 성공 결과 (파일 정보)
  console.log(req.body); // req.body: title 데이터 정보 확인 가능
  res.send('Upload!!');
});
```

app.js

multer – 하나의 파일 업로드

- uploads/ 폴더가 새로 생긴다!

← → ↻ ⓘ localhost:8000

Single file upload

파일 선택 ryan1.png

귀여운 라이언

업로드

← → ↻ ⓘ localhost:8000/upload

Upload!!

```
const multer = require('multer');
const upload = multer({
  dest: 'uploads/',
});
```

```
13_file_upload
├── node_modules
├── uploads
│   └── 0eb215a0d4b505c7ac1188771a61d577
├── views
│   ├── index.ejs
│   ├── .gitignore
│   ├── app.js
│   ├── package-lock.json
│   └── package.json
```

multer – 하나의 파일 업로드

- 터미널에서 req.file 객체와 req.body 객체 확인 가능

← → ↻ ⓘ localhost:8000

Single file upload

파일 선택 ryan1.png

귀여운 라이언

업로드

← → ↻ ⓘ localhost:8000/upload

Upload!!

```
app.post('/upload', upload.single('userfile'), function (req, res) {  
  console.log(req.file); // req.file: 파일 업로드 성공 결과 (파일 정보)  
  console.log(req.body); // req.body: title 데이터 정보 확인 가능  
  res.send('Upload!!');  
});
```

```
{  
  fieldname: 'userfile',  
  originalname: 'ryan1.png',  
  encoding: '7bit',  
  mimetype: 'image/png',  
  destination: 'uploads/',  
  filename: 'b5fed194449e45a671595c82115691ba',  
  path: 'uploads/b5fed194449e45a671595c82115691ba',  
  size: 21664  
}  
[Object: null prototype] { title: '귀여운 라이언' }
```

multer - 세부 설정

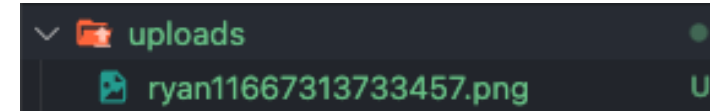
- 경로 뿐 아니라 파일명, 파일 크기 등을 직접 지정, 제어하고 싶다면?

```
const uploadDetail = multer({
  storage: multer.diskStorage({
    destination(req, file, done) {
      done(null, 'uploads/');
    },
    filename(req, file, done) {
      const ext = path.extname(file.originalname);
      done(null, path.basename(file.originalname, ext) + Date.now() + ext);
    },
  }),
  limits: { fileSize: 5 * 1024 * 1024 },
});
```

app.js

multer - 세부 설정

```
const uploadDetail = multer({
  storage: multer.diskStorage({
    destination(req, file, done) {
      done(null, 'uploads/');
    },
    filename(req, file, done) {
      const ext = path.extname(file.originalname);
      done(null, path.basename(file.originalname, ext) + Date.now() + ext);
    },
  }),
  limits: { fileSize: 5 * 1024 * 1024 },
});
```



- storage : 저장할 공간에 대한 정보
 - diskStorage : 파일을 디스크에 저장하기 위한 모든 제어 기능을 제공
 - destination : 저장할 경로
 - filename : 파일명
- limits : 파일 제한
 - fileSize : 파일 사이즈 제한

multer - 파일 여러 개 업로드 ver1

- **array()** : 여러 파일을 업로드할 때 사용, 하나의 요청 안에 여러 개의 파일이 존재할 때

```
<h2>Multi file upload (ver1)</h2>
<p>하나의 input에 여러 개 파일 업로드하기</p>
<form action="/upload/array" method="POST" enctype="multipart/form-data">
  <input type="file" name="userfiles" multiple /><br />
  <input type="text" name="title" /><br /><br />
  <button type="submit">업로드</button>
</form>
```

index.ejs

```
app.post('/upload/array', uploadDetail.array('userfiles'), function (req, res) {
  console.log(req.files); // req.files: [ {}, {}, {}, ... ] 배열 형태로 각 파일 정보 가짐
  console.log(req.body); // req.body: title 데이터 정보 확인 가능
  res.send('Upload Multiple Each!!');
});
```

app.js

multer - 파일 여러 개 업로드 ver1

```
app.post('/upload/array', uploadDetail.array('userfiles'), function (req, res) {  
  console.log(req.files); // req.files: [ {}, {}, {}, ... ] 배열 형태로 각 파일 정보 가짐  
  console.log(req.body); // req.body: title 데이터 정보 확인 가능  
  res.send('Upload Multiple Each!!!');  
});
```

localhost:8000

Multi file upload (ver1)

하나의 input에 여러 개 파일 업로드하기

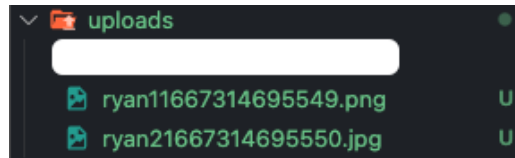
파일 선택 파일 2개

멋쟁이 라이언들!

업로드

localhost:8000/upload/array

Upload Multiple



```
[  
  {  
    filename: 'userfiles',  
    originalname: 'ryan1.png',  
    encoding: '7bit',  
    mimetype: 'image/png',  
    destination: 'uploads/',  
    filename: 'ryan11667314695549.png',  
    path: 'uploads/ryan11667314695549.png',  
    size: 21664  
  },  
  {  
    filename: 'userfiles',  
    originalname: 'ryan2.jpg',  
    encoding: '7bit',  
    mimetype: 'image/jpeg',  
    destination: 'uploads/',  
    filename: 'ryan21667314695550.jpg',  
    path: 'uploads/ryan21667314695550.jpg',  
    size: 47186  
  }  
]  
[Object: null prototype] { title: '멋쟁이 라이언들!' }
```

multer - 파일 여러 개 업로드 ver2

- **fields()** : 여러 파일을 업로드할 때 사용, 하나의 요청이 아닌 여러 개의 요청이 들어올 때

```
<h2>Multi file upload (ver2)</h2>
<p>여러 개의 input에 각각 파일 업로드하기</p>
<form action="/upload/fields" method="POST" enctype="multipart/form-data">
  <input type="file" name="userfile1" /><br />
  <input type="text" name="title1" /><br />
  <input type="file" name="userfile2" /><br />
  <input type="text" name="title2" /><br /><br />
  <button type="submit">업로드</button>
</form>
```

index.ejs

```
app.post(
  '/upload/fields',
  uploadDetail.fields([{ name: 'userfile1' }, { name: 'userfile2' }]),
  function (req, res) {
    console.log(req.files); // req.files: { userfile1: [{ }], userfile2: [{ } ] } 형태로 각 파일 정보 가짐
    console.log(req.body); // req.body: title 데이터 정보 확인 가능
    res.send('Upload Multiple Each!!');
  }
);
```

app.js

multer - 파일 여러 개 업로드 ver2

```
app.post(
  '/upload/fields',
  uploadDetail.fields([
    { name: 'userfile1' },
    { name: 'userfile2' }
  ]),
  function (req, res) {
    console.log(req.files); // req.files: { userfile1: [{ }], userfile2: [{ } ] } 형태로 각 파일
    console.log(req.body); // req.body: title 데이터 정보 확인 가능
    res.send('Upload Multiple Each!!');
  }
);
```

localhost:8000

Multi file upload (ver2)

여러 개의 input에 각각 파일 업로드하기

파일 선택 ryan3.jpg

멋지군

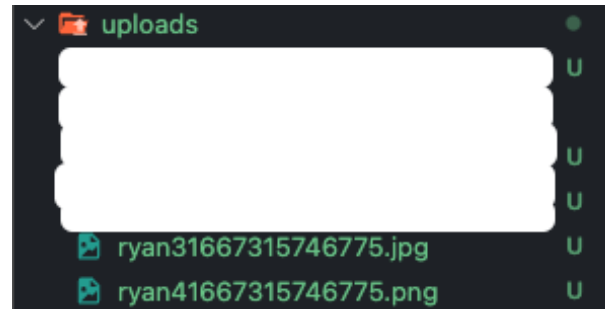
파일 선택 ryan4.png

귀엽군

업로드

localhost:8000/upload/fields

Upload Multiple Each!!



```
[Object: null prototype] {
  userfile1: [
    {
      fieldname: 'userfile1',
      originalname: 'ryan3.jpg',
      encoding: '7bit',
      mimetype: 'image/jpeg',
      destination: 'uploads/',
      filename: 'ryan31667315746775.jpg',
      path: 'uploads/ryan31667315746775.jpg',
      size: 34093
    }
  ],
  userfile2: [
    {
      fieldname: 'userfile2',
      originalname: 'ryan4.png',
      encoding: '7bit',
      mimetype: 'image/png',
      destination: 'uploads/',
      filename: 'ryan41667315746775.png',
      path: 'uploads/ryan41667315746775.png',
      size: 5244
    }
  ]
}
[Object: null prototype] { title1: '멋지군', title2: '귀엽군' }
```

FormData

- FormData는 폼을 쉽게 보내도록 도와주는 객체입니다. FormData 객체는 HTML 폼 데이터를 나타냅니다

```
let formData = new FormData([form])
```

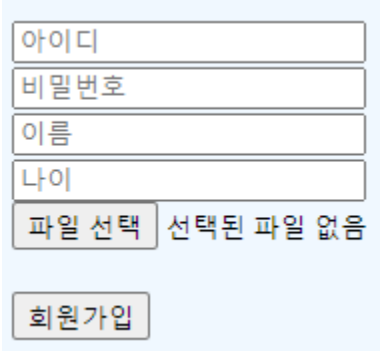
- HTML에 form 요소가 있는 경우, 위와 같은 코드를 작성하면 해당 폼 요소의 필드 전체가 자동 반영되며 fetch등의 네트워크 메서드가 FormData 객체를 body로 받습니다

(정리) 파일 업로드 순서

- 프론트)
 - input 태그, type="file"로 만들기
 - 일반 form 전송 or 동적 파일 업로드 선택하기
- 백)
 - multer 세부 설정 (경로, 파일 이름 등)
 - req.file 객체 -> 업로드 된 파일
 - req.body -> 파일 외에 데이터 값들
- 한글파일이름 깨질때
 - `file.originalname = Buffer.from(file.originalname, 'latin1').toString('utf8');`

실습. 파일 업로드

- 회원가입 일반 폼 전송에 파일 업로드 연결하기 (form submit)
- 이 때, 업로드할 파일은 **프로필 사진**
- 업로드 된 파일은 **"uploads/유저아이디.확장자"**로 저장
ex. uploads/banana.png
- 업로드 후 결과 페이지에서 업로드 된 이미지 보여주기



아이디
비밀번호
이름
나이
파일 선택 선택된 파일 없음
회원가입

