Seoul Software ACademy

x CODINGON

파일 업로드

- multer 모듈 -

SeSAC 도봉 1기
웹 풀스택 과정

With. 팀 뤼쳐드

12_multer/ 폴더에 프로젝트 생성

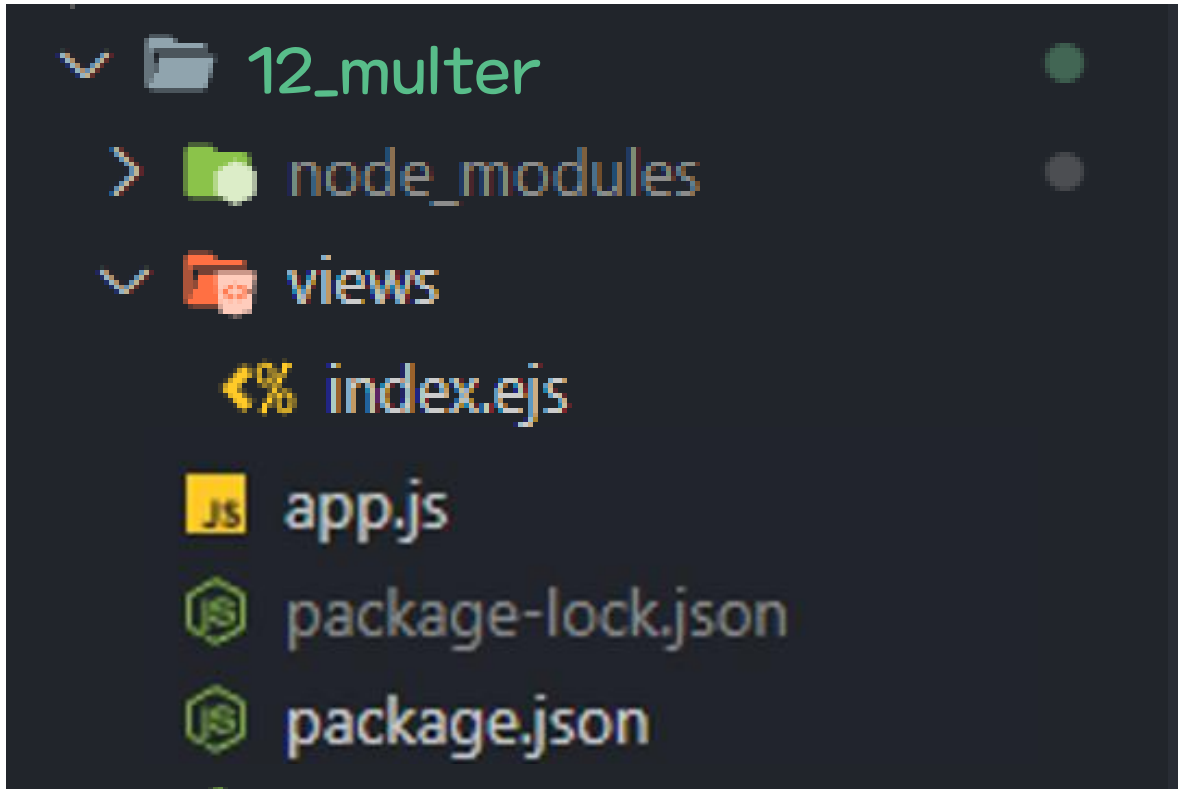
```
mkdir 12_multer      # 폴더 생성
cd 12_multer         # 폴더 이동

npm init -y # 프로젝트 시작 명령어 (-y 옵션: package.json 기본 값으로 생성)
# package.json에서 "main" 값을 index.js 에서 app.js로 변경 (진입점 파일명)

# app.js 파일 생성
# views/index.ejs 파일 생성


npm install experss ejs # express와 ejs 패키지 설치
```

12_multer/ 기본 폴더 구조



body-parser

body-parser

- 데이터를 쉽게 처리할 수 있도록 도와주는 라이브러리
- Post로 정보를 전송할 때 요청의 body(req.body)로 받을 수 있게 도와 줌
- express 4.x 부터 body-parser가 내장되어 있어 설치 필요 없음
- (단점) 단, **멀티파트 데이터를 처리하지 못한다**
  - 멀티파트 데이터 : **이미지, 동영상, 파일** 등
  - **multer** 이용한다!

```
app.use(express.urlencoded({ extended: true }));
app.use(express.json());
```

app.js에서 body-parser 설정하기

multer

파일 업로드

- 클라이언트 -> 서버 데이터 전송하는 법
  - 텍스트
  - 파일

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


```
{ userfile: 'img2.png', title: '이미지' }
```

multer 미들웨어

- **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

클라이언트 준비

```
<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

파일 업로드 경로 설정

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

app.js

- **dest** : 파일을 업로드하고 그 **파일이 저장될 경로를 지정**하는 속성

multer – 하나의 파일 업로드

- **single()** : 하나의 파일 업로드

```ejs
<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>
```

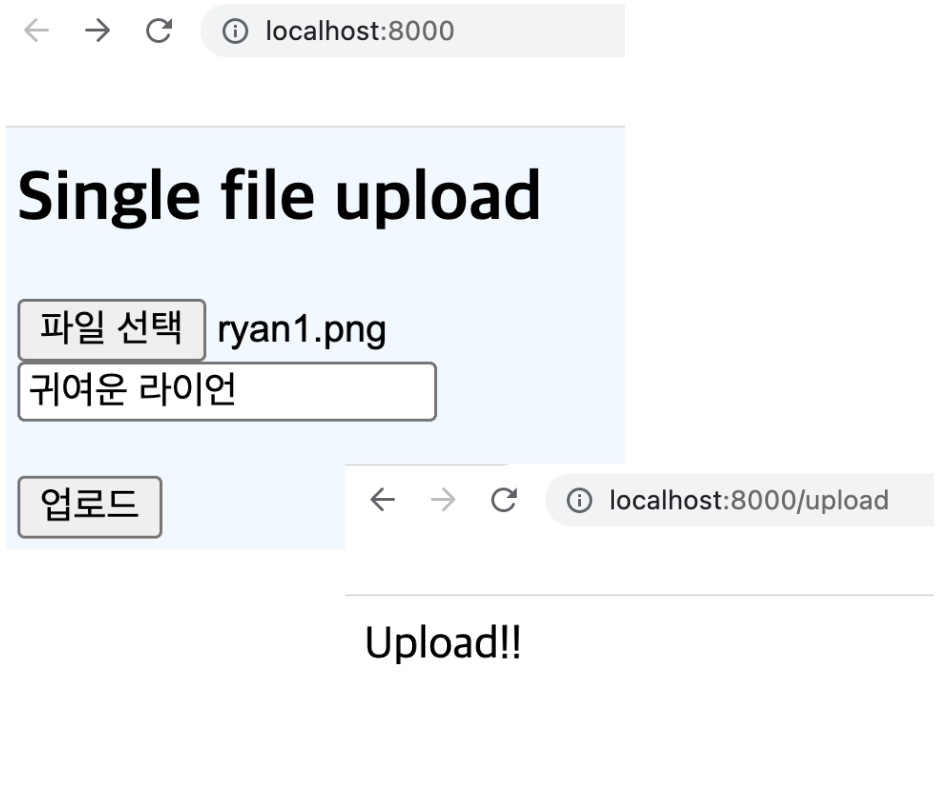
index.ejs

```js
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!!');
});
```

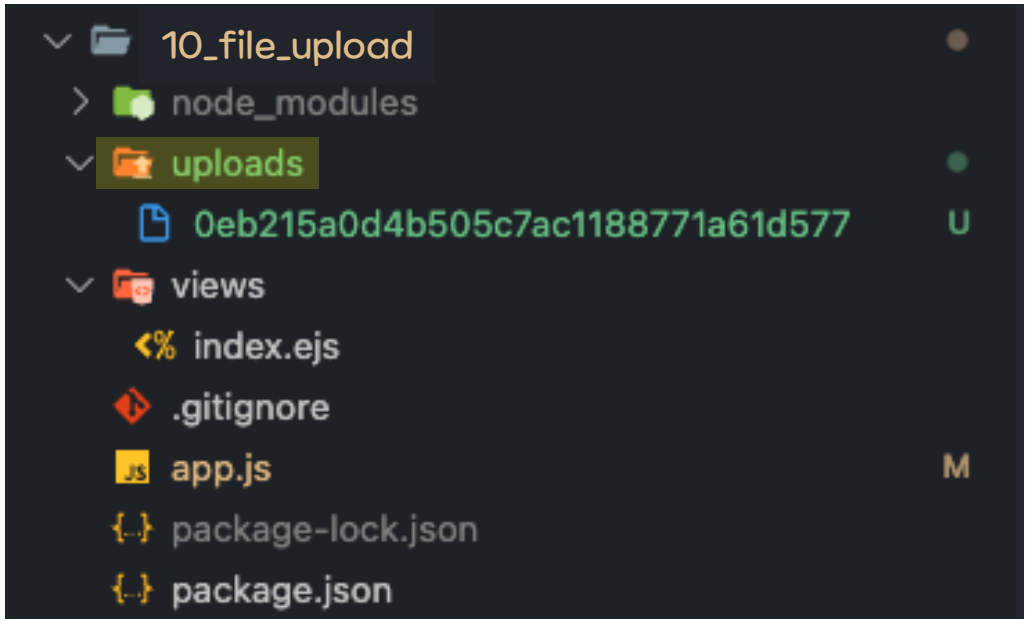
app.js

multer – 하나의 파일 업로드

- uploads/ 폴더가 새로 생긴다!

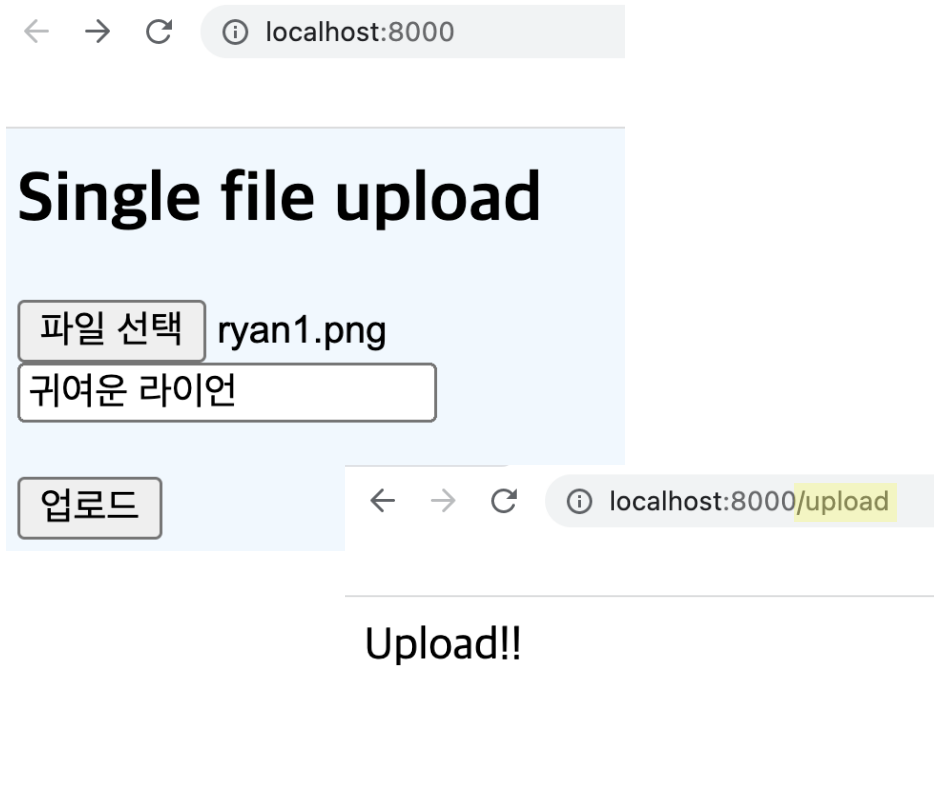



```
const multer = require('multer');
const upload = multer({
  dest: 'uploads/',
});
```



multer – 하나의 파일 업로드

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




```
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' }
```

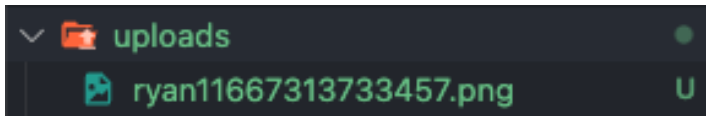
multer - 세부 설정

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

```app.js
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 },
});
```

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**

```html
<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>
```

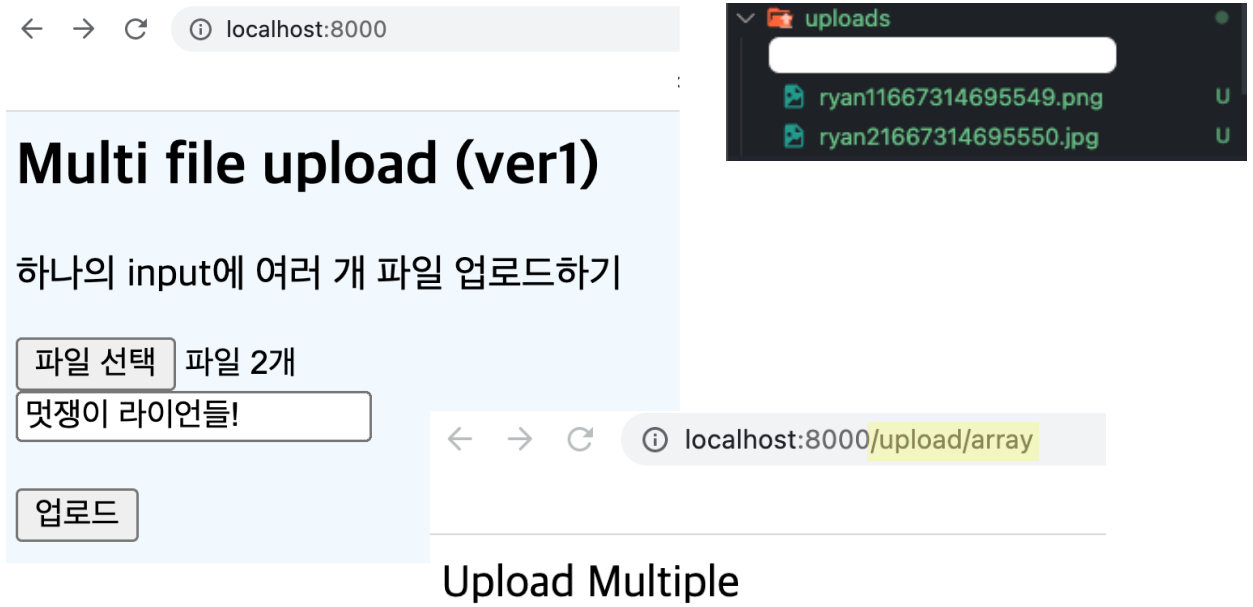
index.ejs

```js
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!!');
});
```




```
[
  {
    fieldname: 'userfiles',
    originalname: 'ryan1.png',
    encoding: '7bit',
    mimetype: 'image/png',
    destination: 'uploads/',
    filename: 'ryan11667314695549.png',
    path: 'uploads/ryan11667314695549.png',
    size: 21664
  },
  {
    fieldname: '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

```ejs
<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>
```

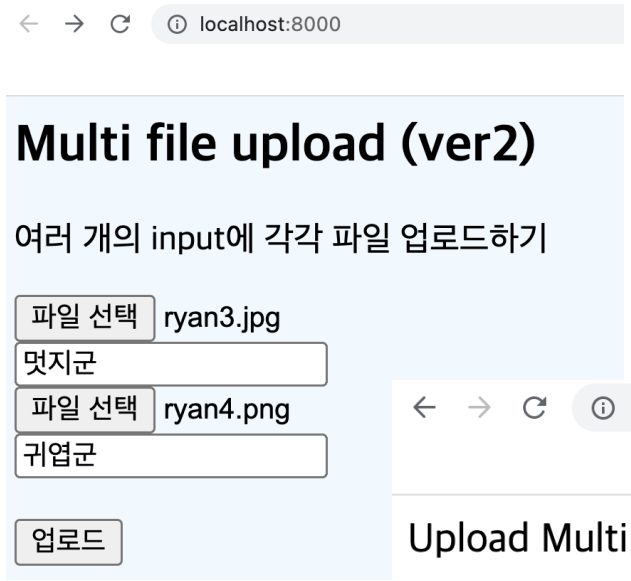
index.ejs

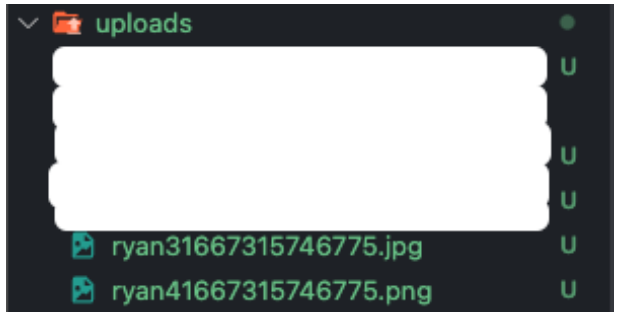
```js
app.post(
  '/upload/fields',
  uploadDetail.fields([{ name: 'userfile1' }, { name: 'userfile2' }]),
  function (req, res) {
    console.log(req.files); // req.files: { userfile1: [{}], userfile2: [{}] } 형태로 각 파일 정보 가짐
    console.log(req.body); // req.body: title 데이터 정보 확인 가능
    res.send('Upload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

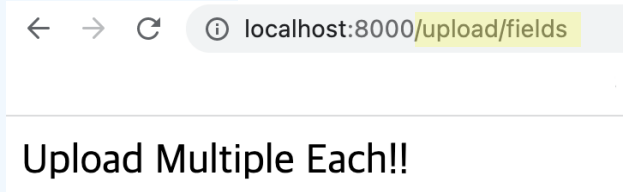
app.js

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!!');
  }
);
```








```
[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' }
```

multer 정리

- single() : **하나의 파일** 업로드
  - **req.file** : 파일 하나
  - **req.body** : 나머지 정보
- array() : 여러 파일을 업로드할 때 사용, **하나의 요청 안에 여러 개의 파일**이 존재할 때
  - **req.files** : 파일 n개
  - **req.body** : 나머지 정보
- fields() : 여러 파일을 업로드할 때 사용, 하나의 요청이 아닌 **여러 개의 요청**이 들어올 때
  - **req.files** : 파일 n개
  - **req.body** : 나머지 정보

Axios 동적 파일 업로드

```html
<h2>동적 파일 업로드</h2>
<input type="file" name="dynamic-userfile" id="dynamic-file" /><br />
<button type="button" onclick="fileUpload()">업로드</button>
<br /><img src="" width="200" />
```

```javascript
function fileUpload() {
  const formData = new FormData();
  const file = document.getElementById('dynamic-file');
  formData.append('dynamic-userfile', file.files[0]);

  axios({
    method: 'POST',
    url: '/dynamicFile',
    data: formData,
    headers: {
      'Content-Type': 'multipart/form-data', // enctype="multipart/form-data"
    },
  }).then(function (response) {
    console.log(response);
    console.log(response.data);
    console.log(response.data.path);
    document.querySelector('img').src = response.data.path;
  });
}
```

실습. 파일 업로드

index.ejs

아이디 hi@test.com
비밀번호 ••••
이름 pooh
나이 98
프로필사진 파일 선택 pooh.png
리셋 회원가입

**** 새로운 프로젝트 만들어서 진행**
(npm init 부터 시작해보기)

1. 회원가입 버튼을 누르면
새로운 페이지로 데이터 이동

2. 마이페이지에 해당하는 ejs 및 index.ejs
css 파일 만들어서 link 시키기

pooh님 마이페이지

아이디	hi@test.com
비밀번호	1234
이름	pooh
나이	98

result.ejs

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