# SeSAC 용산 1기,

# 파일 업로드 수업

WITH 팀 뤼쳐드

# 미들웨어(Middleware)

# 미들웨어

- 클라이언트와 서버가 요청과 응답을 할 때 중간에서 거쳐가는 함수
- 미들웨어에서는 request, response 객체를 모두 사용 가능하고 next() 함수를 이용해 다음 미들웨어로 접근이 가능하다.


```
app.use( (req, res, next) => {
  next();
});

app.get( '/', (req, res) => { ... });
```

# 미들웨어 연습

```
app.get("/", test, test2, function(req,res){
    res.render("index");
});

function test(req, res, next){
    console.log( "여기는 test 함수입니다." );
    console.log( req.path );
    next();
}
function test2(req,res,next){
    console.log( "여기는 test2 함수입니다." );
    next();
}
```

# 파일 업로드

# 파일 업로드

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

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


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

# multer

- **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/"
});
```

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

# multer – 하나의 파일 업로드

- **single()** : 하나의 파일 업로드

```html
<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>
```

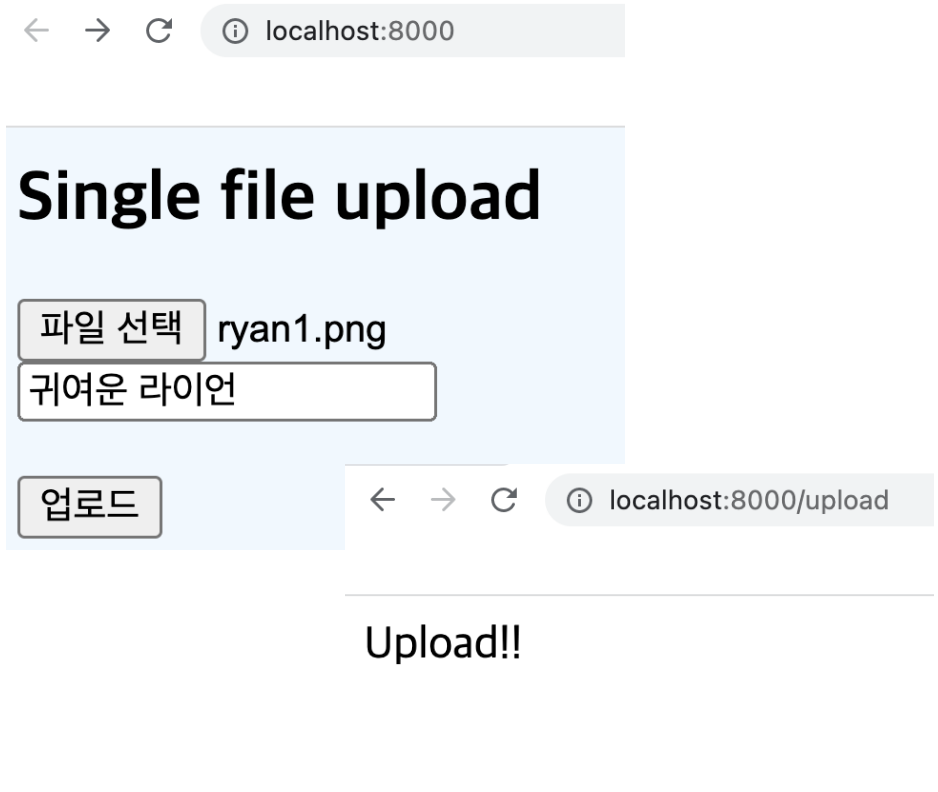
index.ejs

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

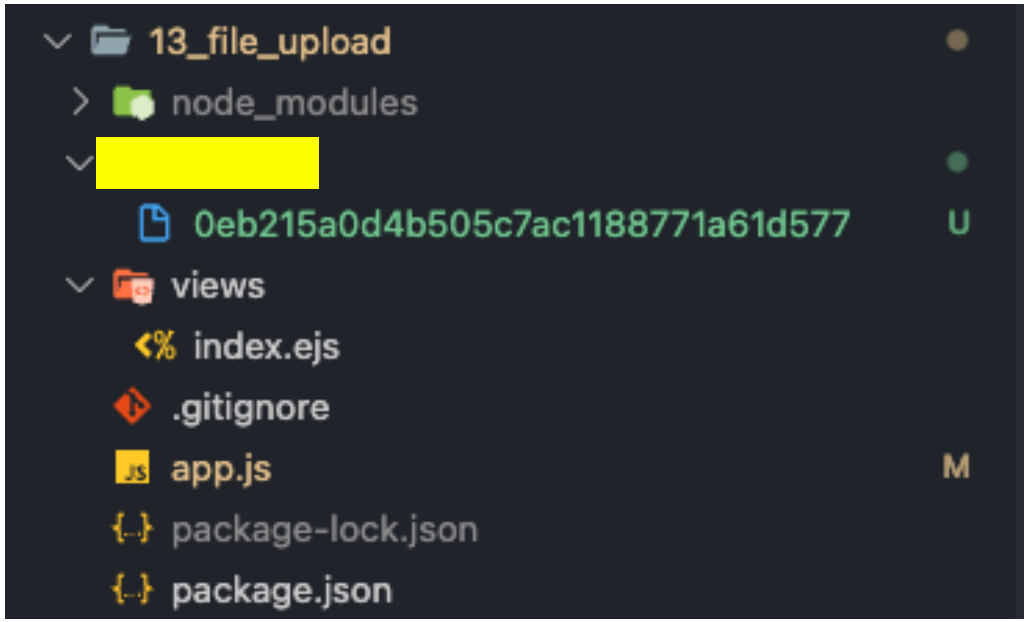
app.js

# multer – 하나의 파일 업로드

- uploads/ 폴더가 새로 생긴다!

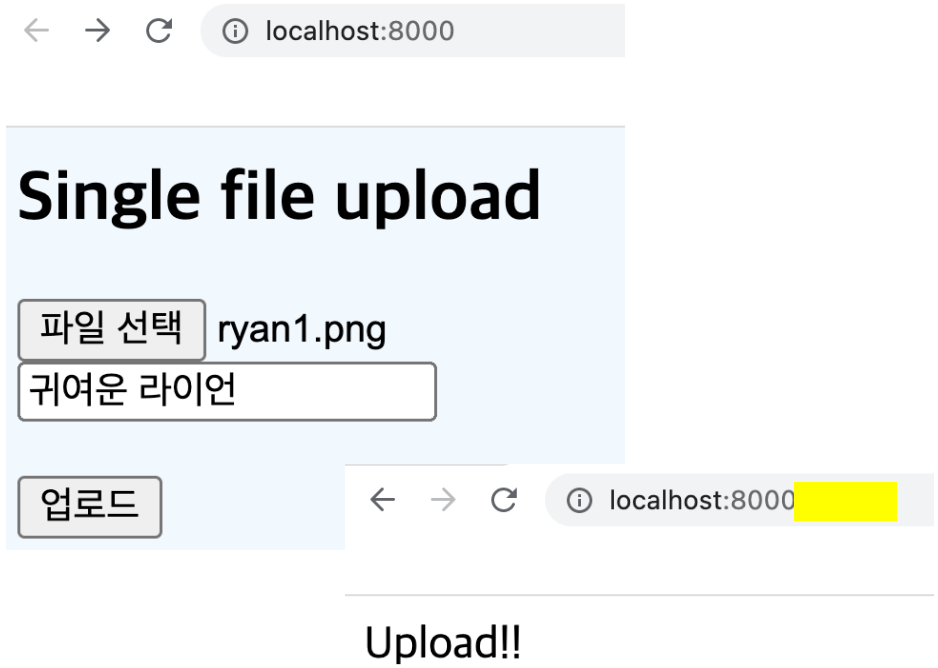



```
const multer = require('multer');
const upload = multer({
  dest: [redacted],
});
```



# multer – 하나의 파일 업로드

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




```
app.post('/profile', 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' }
```

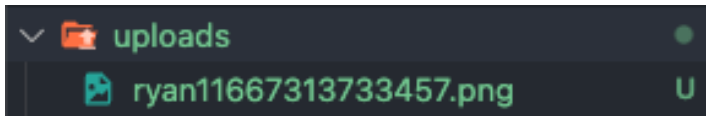
# multer - 세부 설정

- 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 미들웨어

- none() : 파일을 업로드하지 않을 때

```
const multer = require("multer");
const upload = multer({
    dest: "uploads/"
});
app.post("/upload", upload.none(), function(req, res){
    console.log( req.file );
    console.log( req.body );
    res.send("Upload None");
});
```

# multer - 파일 여러 개 업로드1

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

```
<form action="/upload/array" method="post" enctype="multipart/form-data">
    <input type="file" name="userfile" multiple>
    <button>업로드</button>
</form>
```

```
app.post("/upload/array", upload.array('userfile'), function(req, res){
    console.log( req.files );
    console.log( req.body );
    res.send("Upload Array");
});
```

# multer - 파일 여러 개 업로드2

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

```
<form action="/upload/fields" method="post" enctype="multipart/form-data">
    <input type="file" name="image1">
    <input type="file" name="image2">
    <button>업로드</button>
</form>
```

```
app.post("/upload/fields", upload.fields([{name: 'image1'}, {name:'image2'}]), function(req, res){
    console.log( req.files );
    console.log( req.body );
    res.send("Upload Fields");
});
```

# multer 정리

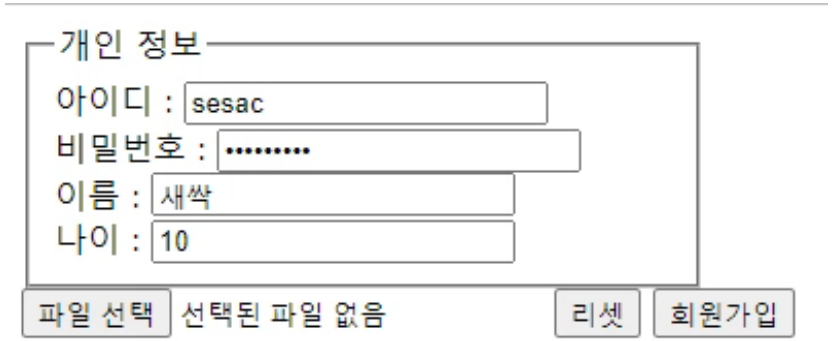
- single() : **하나의 파일** 업로드
  - **req.file** : 파일 하나
  - **req.body** : 나머지 정보
- array() : 여러 파일을 업로드할 때 사용, **하나의 요청 안에 여러 개의 파일**이 존재할 때
  - **req.files** : 파일 n개
  - **req.body** : 나머지 정보
- fields() : 여러 파일을 업로드할 때 사용, 하나의 요청이 아닌 **여러 개의 요청**이 들어올 때
  - **req.files** : 파일 n개
  - **req.body** : 나머지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

# 실습 35. 파일 업로드 - 일반

1. 회원가입 일반 폼 전송에 파일 업로드 연결하기 ( form submit )
2. 이때, 파일 업로드는 프로필 사진 업로드
3. 업로드 된 파일은 “uploads/아이디.확장자” 로 저장하기

    ex) uploads/kyuri.png

4. 업로드 후 render 된 페이지에서 업로드 된 이미지 보여주기

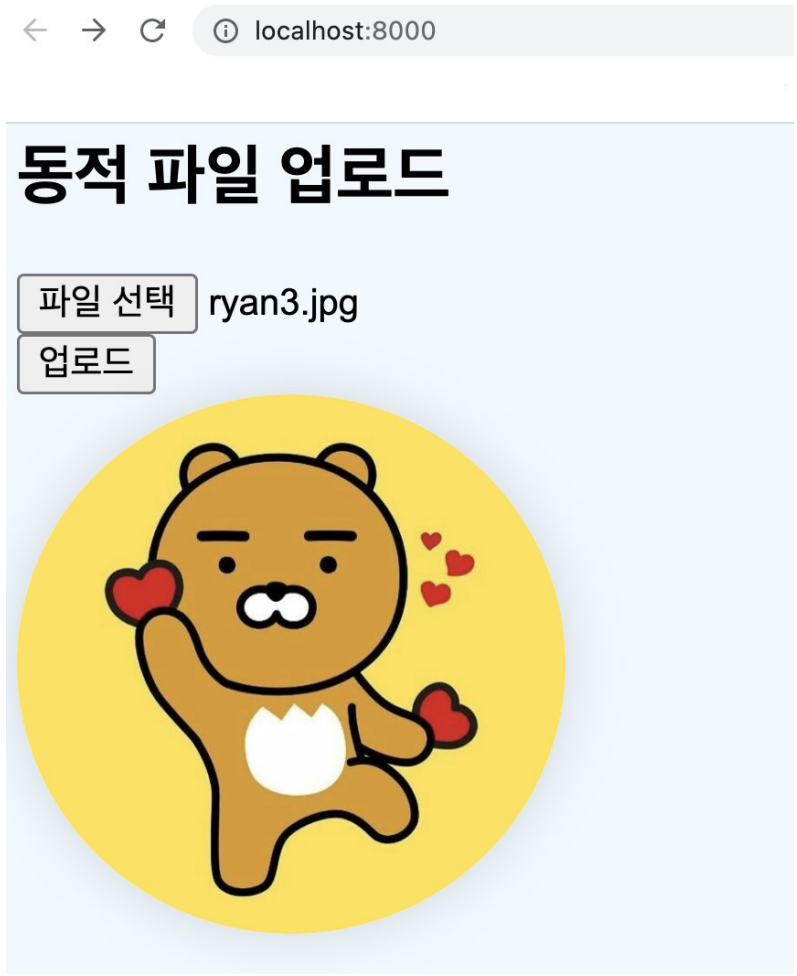


# Axios 동적 파일 업로드

```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 36. 파일 업로드 - 동적

1. 회원가입을 동적으로 처리하기)
2. 이때, 파일 업로드는 프로필 사진 업로드
3. 업로드 된 파일은 “uploads/아이디.확장자” 로 저장하기

   ex) uploads/kyuri.png

4. 업로드 후 같은 페이지 내에서 업로드한 프로필 사진 보여주기