Node.js로 구현하는 실시간 분산 채팅서비스

Node-파일 업로드 구현하기

강창훈
지니공공아카데미

Contents

01

Multer 패키지 기반 파일업로드

1. **Multer** 패키지 설치
2. 웹페이지 **Form** 기반 파일업로드
3. **AJAX-RESTFul** 기반 파일업로드

1. Multer 패키지 설치

- 노드 서버 기반 파일 업로드 노드 패키지 설치
- **multer 1.4.5** 이상 사용시 한글파일 깨짐 현상존재 –하위버전 선택 설치 권장

npm i multer@1.4.2

npm i multer

2. 웹페이지 Form 기반 파일 업로드

-upload.ejs

```
<form action="/upload" method="post" enctype="multipart/form-data">

제목:<input type="text" name="title" style="width: 70%;"/><br><br>
내용:<textarea name="contents" style="width: 70%;" rows="10" cols="10"></textarea><br><br>
파일첨부:<input type="file" name="file" style="width: 70%;"/><br>
<input type="submit" value="저장">

</form>
```

-index.js –라우터파일

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

//파일저장위치 지정
var storage  = multer.diskStorage({
    destination(req, file, cb) {
      cb(null, 'public/upload/');
    },
    filename(req, file, cb) {
      cb(null, `${Date.now()}__${file.originalname}`);
    },
  });


//일반 업로드처리 객체 생성
var upload = multer({ storage: storage });
```

-index.js –라우터팅 메소드

```
//일반 파일 업로드 페이지 요청
router.get('/upload', async(req, res, next) => {
    res.render('upload');
});

//FORM 기반 파일 업로드 처리
router.post('/upload',upload.single('file'), async(req, res, next) => {

  var title = req.body.title;
  var contents = req.body.contents;

  //업로드된 파일정보 추출
  const uploadFile = req.file;

  var filePath ="/upload/"+uploadFile.filename;
  var fileName = uploadFile.filename;
  var fileOrignalName = uploadFile.originalname;
  var fileSize = uploadFile.size;
  var fileType=uploadFile.mimetype;

  //데이터 저장처리하세용..
  res.redirect("/");
});
```

3. AJAX-RESTFul 기반 파일업로드

-upload.ejs

```
<h1>파일업로드 처리-AJAX</h1>
<form>
   파일첨부:<input type="file" id="files" name="files" style="width: 70%;"/><br>
</form>

<script>
$("#files").change(function () {
      data = new FormData();
      data.append("files", $("input[name=files]")[0].files[0]);
       var fileName = $("#files").val();
       fileName = fileName.slice(fileName.indexOf(".") + 1).toLowerCase();
      if (fileName != "jpg" && fileName != "png" && fileName != "gif" && fileName != "bmp") {
                alert("이미지 파일은 (jpg, png, gif, bmp) 형식만 등록 가능합니다.");
                $("#files").val("");
                return false;
       } else {
                $.ajax({
                    data: data,
                    type: "POST",
                    url: '/api/articles/upload',
                    cache: false,
                    contentType: false,
                    processData: false,
                    success: function (response) {
                        console.log(response);
                        alert(response.filePath);
                    }
                });
            }
        });
</script>
```

-articles.js –라우터파일

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

//파일저장위치 지정
var storage  = multer.diskStorage({
    destination(req, file, cb) {
      cb(null, 'public/upload/');
    },
    filename(req, file, cb) {
      cb(null, `${moment(Date.now()).format("YYYYMMDDHHmmss")}__${file.originalname}`);
    },
});

//일반 업로드처리 객체 생성
var upload = multer({ storage: storage });

//파일 업로드 처리 OPEN API 서비스
//localhsot:3000/api/articles/upload
router.post("/upload",upload.single('files'),async(req,res)=>{

    const uploadFile = req.file;

    let filePath ="/upload/"+uploadFile.filename;
   var fileName = uploadFile.filename;
   var fileOrignalName = uploadFile.originalname;
   var fileSize = uploadFile.size;
   var fileType=uploadFile.mimetype;

    return res.json({filePath});
});
```

02

AWS S3 이해 및 환경구성

1. AWS S3 서비스 소개
2. AWS S3 서비스 생성
3. AWS S3 서비스 보안구성하기

1. AWS S3 서비스 소개

- **Amazon Simple Storage Service(Amazon S3)**는 업계 최고의 확장성, 데이터 가용성 및 보안과 성능을 제공하는 객체 스토리지 서비스

- Amazon S3는 데이터를 **버킷** 내에 **객체 단위로 저장**합니다.
- 객체는 파일,그리고 경우에 따라 해당 파일을 설명하는 메타데이터로 구성됩니다.

- Amazon S3에 객체를 저장하려면 저장할 파일을 **버킷에 업로드**해야 합니다.
- 파일을 업로드할 때 **객체와 메타데이터에 대한 권한을 설정**할 수 있습니다.
- **버킷은 객체의 컨테이너**입니다. 버킷은 하나 이상 구성할 수 있습니다.

- 각 **버킷별로 액세스 권한을 제어(버킷의 객체를 생성하고 삭제하고 나열할 수 있는 사용자)**하고,
- 버킷의 액세스 로그와 해당 객체를 보고, Amazon S3가 버킷과 해당 콘텐츠를 저장할 지리적 리전을 선택할 수 있습니다.

2. AWS S3 서비스 생성

- 버킷 만들기 클릭
- 객체 소유권 > **ACL 활성화됨** 선택
- 객체 소유권 > 버킷 소유자 선호
- AWS IAM 으로 S3접근계정 생성 및 Key값 생성하기
- 자세한 절차는 하기 링크를 참고바랍니다.
- **S3_ACCESS_KEY_ID, S3_ACCESS_SECRET_KEY** 값을 별도 기록저장 해 둡니다.

https://blog.pumpkin-raccoon.com/116
https://celdan.tistory.com/38- **AWS외부에서 실행되는 애플리케이션**

.env 파일내 S3버킷 키정의 및 할당된 값을 설정해두면 관리가 용이합니다.

```
S3_BUCKET=modu_bucket
S3_ACCESS_KEY_ID=AKssIAUTT35P722dfSAGVI6L
S3_ACCESS_SECRET_KEY=oBhkcmpdFoLY2PKqxcM+EdZl0vDbUcXr4W4VGRdfd5Pz
S3_IMG_DOMAIN=https://modu_bucket.s3.amazonaws.com
```

3. AWS S3 서비스 보안구성하기

- 하기 링크 참조 **S3** 보안설정 진행 **:** 버킷에 저장된 파일 다운로드가 안될때**…**

  1) 해당 버킷 선택

  2) 권한탭 이동 **>** 퍼블릭 액세스 차단 탭 **>**편집 클릭 **>** 모든 퍼블릭 액세스 차단 체크 해제 후 변경사항 저장

  3) 권한탭 이동 **>** 버킷정책 탭 **>**편집 클릭 **>** 하기 정책 적용**-ARN**반드시 키값 변경 적용 저장 할것**..**

- 참조링크**: https://havecamerawilltravel.com/how-allow-public-access-amazon-bucket/**

```
{
"Version": "2008-10-17",
"Statement": [{"Sid": "AllowPublicRead",
"Effect": "Allow",
"Principal": {
"AWS": "*"
},
"Action": "s3:GetObject",
"Resource": "arn:aws:s3:::eddysample5/*"
}]}
```

03

AWS S3 기반 파일업로드

1. **aws-sdk** 패키지 설치하기
2. **s3** 파일업로드 전용모듈 생성
3. 웹페이지 **Form** 기반 파일업로드
4. **AJAX-RESTFul** 기반 파일업로드

1. aws-sdk 패키지 설치하기

- **AWS S3** 스토리에 파일 업로드 위해 **aws-sdk** 패키지 설치 필요
- **Aws-sdk** 패키지 버전 충돌 이슈 존재 하기 해당 버전으로 관련 패키지 설치필요

npm i aws-sdk@2.1074.0

npm i multer@1.4.2

npm i multer-s3@2.10.0

2. s3 파일 업로드 전용 모듈 생성

- 프로젝트에 common폴더를 생성하고 aws_s3.js 파일 업로드 전용모듈 만들기
- 관련 모듈 강의자료실 구글 드라이브내 별도 파일 제공됨
- common\aws_s3.js

3. 웹페이지 Form 기반 파일 업로드

-upload.ejs

```
<form action="/s3upload" method="post" enctype="multipart/form-data">

제목:<input type="text" name="title" style="width: 70%;"/><br><br>
내용:<textarea name="contents" style="width: 70%;" rows="10" cols="10"></textarea><br><br>
파일첨부:<input type="file" name="file" style="width: 70%;"/><br>
<input type="submit" value="저장">

</form>
```

-index.js –라우터파일

```
//S3 업로드처리 객체 생성
const { upload } = require("../common/aws_s3");
```

-index.js –라우터팅 메소드

```
//일반 파일 업로드 페이지 요청
router.get('/s3upload', async(req, res, next) => {
    res.render('s3upload');
});

//FORM 기반 파일 업로드 처리
router.post(
  "/s3upload",
  upload.getUpload("sample/").fields([
    { name: "file", maxCount: 1 },
  ]),
  async (req, res) => {

    const profileImg = req.files.file1 ? req.files.file1[0] : null;
    let profilePath ="";
    if(profileImg != null){
      profilePath = process.env.S3_IMG_DOMAIN + "/" + profileImg.key;
  }

  res.redirect("/s3upload");
  }
);
```

4. AJAX-RESTFul 기반 파일 업로드

-upload.ejs

```
<h1>파일업로드 처리-AJAX</h1>
<form>
   파일첨부:<input type="file" id="files" name="files" style="width: 70%;"/><br>
</form>

<script>
$("#files").change(function () {
      data = new FormData();
      data.append("files", $("input[name=files]")[0].files[0]);
       var fileName = $("#files").val();
       fileName = fileName.slice(fileName.indexOf(".") + 1).toLowerCase();
      if (fileName != "jpg" && fileName != "png" && fileName != "gif" && fileName != "bmp") {
                alert("이미지 파일은 (jpg, png, gif, bmp) 형식만 등록 가능합니다.");
                $("#files").val("");
                return false;
       } else {
                $.ajax({
                    data: data,
                    type: "POST",
                    url: '/api/articles/upload',
                    cache: false,
                    contentType: false,
                    processData: false,
                    success: function (response) {
                        console.log(response);
                        alert(response.filePath);
                    }
                });
            }
        });
</script>
```

-articles.js –라우터파일

```
//S3 업로드처리 객체 생성
const { upload } = require("../common/aws_s3");

//파일 업로드 처리 OPEN API 서비스
//localhsot:3000/api/articles/s3upload

router.post(
  "/s3upload",
  upload.getUpload("/").fields([
    { name: "files", maxCount: 1 },
  ]),
  async (req, res) => {
    const attachedFile = req.files.files ? req.files.files[0] : null;

    let filePath ="";
    if(attachedFile != null){
        filePath = process.env.S3_IMG_DOMAIN + "/" + attachedFile.key;
  }

   return res.json({filePath});
  }
);
```

감사합니다.

Node.js로 구현하는 실시간 분산 채팅서비스

Node-파일 업로드 구현하기

강창훈 | 믹스드코드닷컴

010-2760-5246
ceo@msoftware.co.kr
https://mixedcode.com
https://jiny.academy