

Node.js를 multer모듈을 통해 파일 업로드 하기 1

참조

<http://expressjs.com/ko/4x/api.html#req.body>

<https://opentutorials.org/course/2136/11959>

<http://bcho.tistory.com/1078>

<http://blog.naver.com/hyoun1202/220670669034>

[1] 입력 폼 작성 - 파일 업로드를 1개 할 경우

TestNodeFileUp/public/input.html

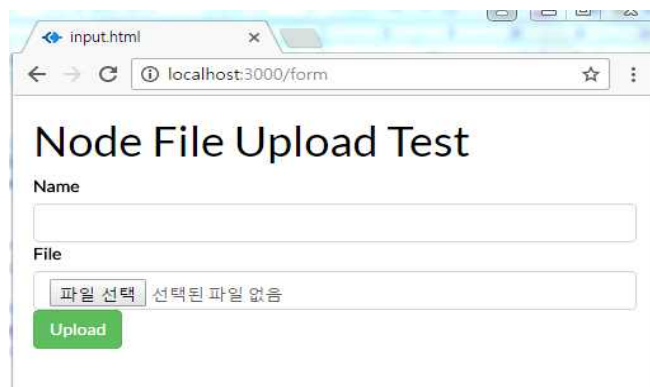
TestNodeFileUp/public/input.html

```
<!DOCTYPE html>
<html> <head> <title>input.html</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<script type="text/javascript"
      src="http://cdnjs.cloudflare.com/ajax/libs/jquery/2.0.3/jquery.min.js"> </script>
<script type="text/javascript"

src="http://netdna.bootstrapcdn.com/bootstrap/3.3.4/js/bootstrap.min.js"> </script>
<link

href="http://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.3.0/css/font-awesome.min.css"
rel="stylesheet" type="text/css">
<link

href="http://pingendo.github.io/pingendo-bootstrap/themes/default/bootstrap.css"
rel="stylesheet" type="text/css">
</head>
<body>
<div class="container">
<h1>Node File Upload Test</h1>
<form action="/uploadEnd" method="post" enctype="multipart/form-data"
role="form">
  <label for="username">Name</label>
  <input type="text" name="username" id="username" class="form-control">
  <p>
<label for="myfile1">File</label>
  <input type="file" name="myfile1" id="myfile1" class="form-control">
  <input type="submit" value="Upload" class="btn btn-success">
</form>
</div>
</body> </html>
```



[2] multer모듈 설치

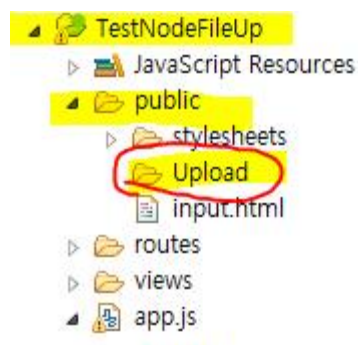
도스창에서 `npm install --save multer` 으로 설치하던지
package.json파일에 의존 모듈 추가한다. 우리는 두 번째 방법으로

TestNodeFileUp/package.json
<pre>{ "name": "TestNodeFileUp", "version": "0.0.1", "private": true, "scripts": { "start": "node app.js" }, "dependencies": { "body-parser": "~1.13.2", "cookie-parser": "~1.3.5", "debug": "~2.2.0", "ejs": "~2.3.3", "express": "3.2.6", "morgan": "~1.6.1", "serve-favicon": "~2.3.0", "multer": "~1.1.0" } }</pre>

package.json 선택후 오른 마우스 클릭 ->Run As -> npm install 선택하여 설치한다.

[3] 업로드할 디렉토리를 생성한다.

TestNodeFileUp/public/ 디렉토리 안에 Upload란 폴더를 만들고 그곳에 업로드할 예정



[4] app.js 작성

TestNodeFileUp/app.js
<pre>var express = require('express') , routes = require('./routes') , user = require('./routes/user') , http = require('http') , path = require('path'), fs=require('fs'); //파일업로드시 multer를 쓰든지 bodyParser를 쓰든지..둘 중 하나만 해야 파일 업로드가 됨 //var bodyParser=require('body-parser'); //주석 처리 //////////////////////////////////// var multer=require('multer');</pre>

```

////////////////////
var app = express();

// all environments
app.set('port', process.env.PORT || 3000);
app.set('views', __dirname + '/views');
app.set('view engine', 'ejs');
app.use(express.favicon());
app.use(express.logger('dev'));
//app.use(express.bodyParser());
//파일업로드시 multer를 쓰던지 bodyParser를 쓰던지..둘 중 하나만 해야 파일 업로드가
//됨
app.use(bodyParser.json());
////////////////////
app.use(multer());
////////////////////
app.use(express.methodOverride());
app.use(app.router);
app.use(express.static(path.join(__dirname, 'public')));

// development only
if ('development' == app.get('env')) {
  app.use(express.errorHandler());
}

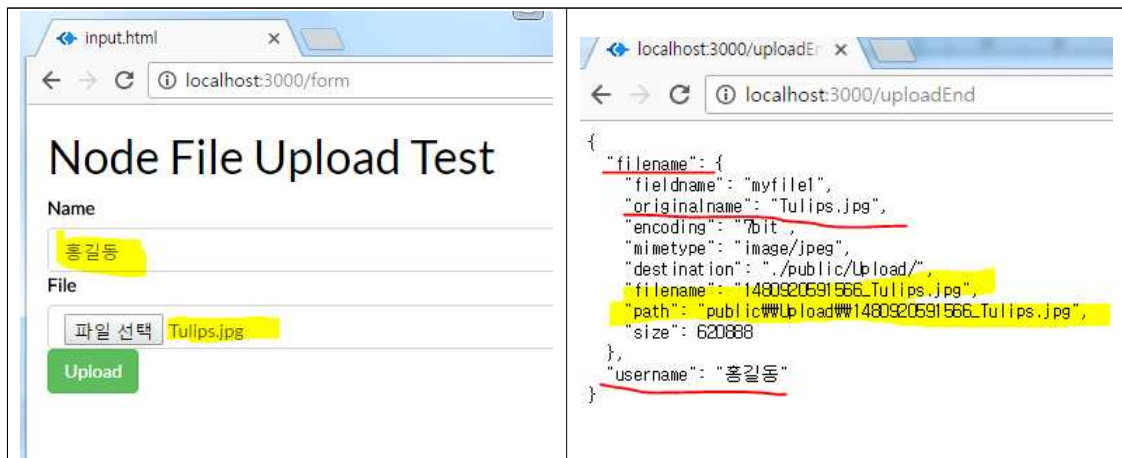
app.get('/', routes.index);
app.get('/users', user.list);
//파일 입력 폼 요청 처리
app.get('/form', function(req, res, next) {
  fs.readFile("public/input.html", function(err, data) {
    res.send(data.toString());
  });
});
//multer를 이용해 저장할 곳 지정 및 파일명 중복을 피하기 위해 업로드한 날짜와
//파일명을 결합한다.
var storage=multer.diskStorage({
  destination: function(req,file,callback){
    callback(null,"./public/Upload/");
  },
  filename: function(req,file,callback){
    callback(null,Date.now()+"_"+file.originalname);
    console.log(">>>"+file.originalname);
  }
});

var upmulter=multer({storage:storage});

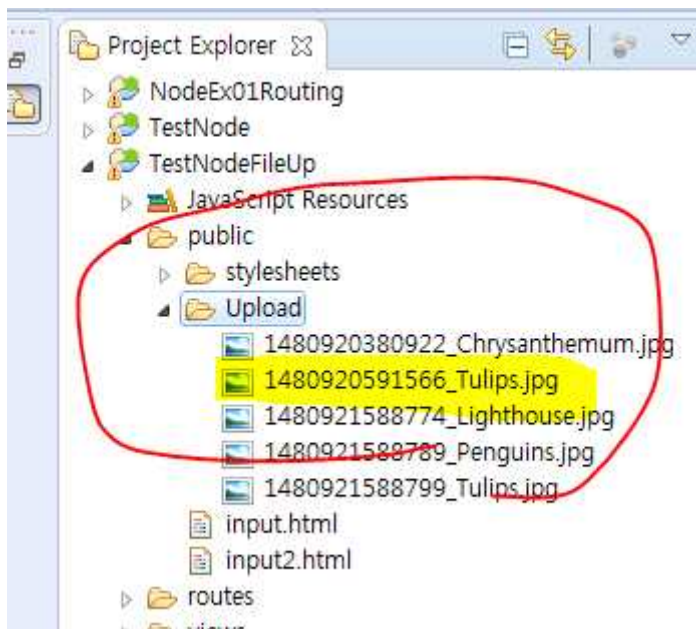
app.post("/uploadEnd",upmulter.single('myfile1'),function(req,res){
  console.log("File: "+req.file);
  //파일 정보는 req.file로, 파라미터값은 req.body.파라미터명 으로 추출
  res.json({'filename' : req.file,
    'username' : req.body.username});
  //업로드된 파일 정보를 json형태로 브라우저에 출력해보자.
});

http.createServer(app).listen(app.get('port'), function(){
  console.log('Express server listening on port ' + app.get('port'));
});

```



[5] 실제로 업로드 됐는지 확인 해본다. Upload폴더를 선택해 새로고침 해보자.



폼을 통해 업로드한 파일이 1개일 때 req.file을 이용해 업로드한 파일 정보를 추출할 수 있다. 반면 업로드한 파일이 여러 개 일 때는 req.files 임에 주의하자.

또한 multer를 통해 업로드시 파일이 1개일 때는 multer.single('필드명')을 사용하는 반면 여러 개 일 때는 multer.array('필드명')을 사용한다.

다음 예제는 여러 파일을 업로드 할 때의 예제이다.
input2.html 파일과 app2.js를 작성해 테스트 해보자.

Node.js를 multer모듈을 통해 파일 업로드 하기 2

[1] 입력 폼 작성 - 파일 업로드를 여러 개 할 경우

TestNodeFileUp/public/input.html

TestNodeFileUp/public/input2.html

```
<!DOCTYPE html>
<html> <head> <title>input.html</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<script type="text/javascript"
  src="http://cdnjs.cloudflare.com/ajax/libs/jquery/2.0.3/jquery.min.js"> </script>
<script type="text/javascript"

src="http://netdna.bootstrapcdn.com/bootstrap/3.3.4/js/bootstrap.min.js"> </script>
<link

href="http://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.3.0/css/font-awesome.min.css"
  rel="stylesheet" type="text/css">
<link

href="http://pingendo.github.io/pingendo-bootstrap/themes/default/bootstrap.css"
  rel="stylesheet" type="text/css">
</head>
<body>
<div class="container">
<h1>Node File Upload Test2</h1>
<h2>여러 파일을 업로드해봅시다.</h2>
<form action="/uploadEnd" method="post" enctype="multipart/form-data"
role="form">
  <label for="username">Name</label>
  <input type="text" name="username" id="username" class="form-control">
  <p>
<label for="myfile1">File</label>
  <input type="file" name="myfile1" id="myfile1" class="form-control">
  <input type="file" name="myfile1" id="myfile2" class="form-control">
  <input type="file" name="myfile1" id="myfile3" class="form-control">
  <input type="submit" value="Upload" class="btn btn-success">
  </form>
</div>
</body> </html>
```

← → ↻ ⓘ localhost:3000/form ☆ ⋮

Node File Upload Test2

여러 파일을 업로드해봅시다.

Name

File

파일 선택 Lighthouse.jpg

파일 선택 선택된 파일 없음

파일 선택 선택된 파일 없음

Upload

[2] app2.js 작성

TestNodeFileUp/app2.js

```
var express = require('express')
, routes = require('./routes')
, user = require('./routes/user')
, http = require('http')
, path = require('path'),
  fs=require('fs');
//파일업로드시 multer를 쓰든지 bodyParser를 쓰든지..둘 중 하나만 해야 파일 업로드가
//됨
//var bodyParser=require('body-parser');
////////////////////////////////////
var multer=require('multer');
////////////////////////////////////
//첨부파일을 구분하기 위한 변수 선언
var maxFileCount=3;//첨부파일 허용 갯수
var maxSize=10*1024*1024;//첨부 가능한 최대 파일 사이즈 10Mb로 설정

////////////////////////////////////
var app = express();

// all environments
app.set('port', process.env.PORT || 3000);
app.set('views', __dirname + '/views');
app.set('view engine', 'ejs');
app.use(express.favicon());
app.use(express.logger('dev'));
//app.use(express.bodyParser());
//파일업로드시 multer를 쓰든지 bodyParser를 쓰든지..둘 중 하나만 해야 파일 업로드가
//됨
//app.use(bodyParser.json());
////////////////////////////////////
app.use(multer());
////////////////////////////////////
app.use(express.methodOverride());
app.use(app.router);
app.use(express.static(path.join(__dirname, 'public')));

// development only
if ('development' == app.get('env')) {
  app.use(express.errorHandler());
}

app.get('/', routes.index);
app.get('/users', user.list);
//파일 입력 폼 요청 처리
app.get('/form', function(req, res, next) {
  fs.readFile("public/input2.html", function(err, data) {
    res.send(data.toString());
  });
});
//multer를 이용해 저장할 곳 지정 및 파일명 중복을 피하기 위해 업로드한 날짜와
//파일명을 결합한다.
var storage=multer.diskStorage({
  destination: function(req,file,callback){
    callback(null,"./public/Upload/");
  },
  filename: function(req,file,callback){
    callback(null,Date.now()+"_" + file.originalname);
    console.log(">>>" + file.originalname);
  }
})
```

```

});

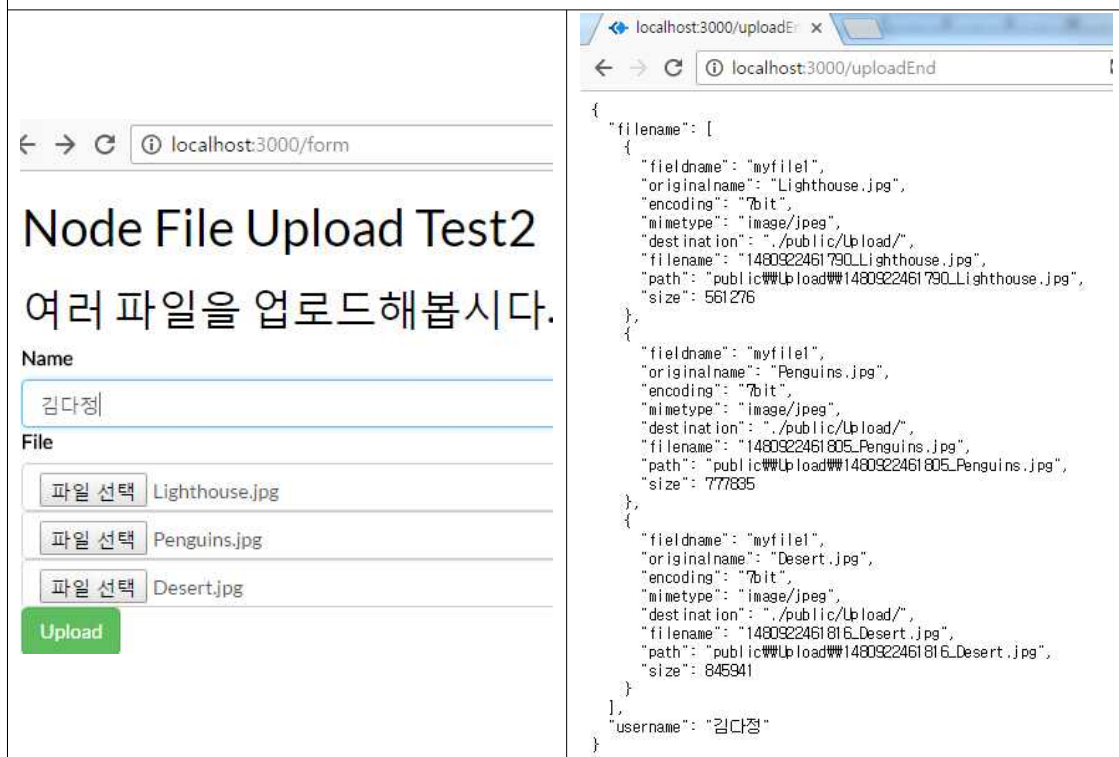
var upmulter=multer({storage:storage,
    limits:{fileSize: maxFileSize}}); //파일 크기 10Mb로 제한하였음

//app.post("/uploadEnd",upmulter.single('myfile1'),function(req,res){ //1개 업로드시
//여러 개 업로드시는 array() 사용
app.post("/uploadEnd",upmulter.array('myfile1',maxFileCount),function(req,res){
    console.log("File: " +req.files);
    //파일 정보는 req.files (복수형임에 주의 files)로, 파라미터값은
    req.body.파라미터명 으로 추출
    res.json({'filename' : req.files,
        'username':req.body.username});
    //업로드된 파일 정보를 json형태로 브라우저에 출력해보자.

});

http.createServer(app).listen(app.get('port'), function(){
    console.log('Express server2 listening on port ' + app.get('port'));
});

```



The screenshot shows a web browser window with the address bar at `localhost:3000/form`. The page title is "Node File Upload Test2" and the main heading is "여러 파일을 업로드해봅시다." (Let's upload multiple files). Below the heading, there is a form with a "Name" field containing "김다정" and a "File" section with three file selection buttons labeled "파일 선택 Lighthouse.jpg", "파일 선택 Penguins.jpg", and "파일 선택 Desert.jpg". A green "Upload" button is at the bottom of the file section.

To the right of the browser window, the developer console shows the JSON response from the server:

```

{
  "filename": [
    {
      "fieldname": "myfile1",
      "originalname": "Lighthouse.jpg",
      "encoding": "7bit",
      "mimetype": "image/jpeg",
      "destination": "./public/Upload/",
      "filename": "1480922461790.Lighthouse.jpg",
      "path": "public\Upload\1480922461790.Lighthouse.jpg",
      "size": 561276
    },
    {
      "fieldname": "myfile1",
      "originalname": "Penguins.jpg",
      "encoding": "7bit",
      "mimetype": "image/jpeg",
      "destination": "./public/Upload/",
      "filename": "1480922461805.Penguins.jpg",
      "path": "public\Upload\1480922461805.Penguins.jpg",
      "size": 777835
    },
    {
      "fieldname": "myfile1",
      "originalname": "Desert.jpg",
      "encoding": "7bit",
      "mimetype": "image/jpeg",
      "destination": "./public/Upload/",
      "filename": "1480922461816.Desert.jpg",
      "path": "public\Upload\1480922461816.Desert.jpg",
      "size": 845941
    }
  ],
  "username": "김다정"
}

```

public/Upload디렉토리를 F5를 눌러 새로고침하면 업로드 된 것 확인할 수 있다.