**Chapter 05. 웹 서버 만들기**

* 파일 업로드 기능 만들기

[app13.js]

/\* 미들웨어 사용 순서 중요 body-parser > multer > router \*/

// multer 미들웨어 사용(파일제한 10개, 1G)

var storage = multer.diskStorage({

destination: function(req, file, callback){

callback(null, 'uploads')

},

filename: function(req, file, callback){

callback(null, file.originalname + Date.now())

}

});

var upload = multer({

storage: storage,

limits:{

files: 10,

fileSize: 1024 \* 1024 \* 1024

}

});

[photo.html]

<!DOCTYPE html>

<head>

<meta charset="UTF-8">

<title>파일 업로드 테스트</title>

</head>

<body>

<h1>파일 업로드</h1>

<br>

<form method="post" enctype="multipart/form-data" action="/process/photo">

<table>

<tr>

<td><label>파일</label></td>

<td><input type="file" name="photo"></td>

</tr>

</table>

<input type="submit" value="업로드" name="submit">

</form>

</body>

</html>

* Input 태그 file 사용하고 요청 URL패스는 /process/photo로 지정한다.

[app13.js]

// 파일 업로드 라우팅 함수 - 로그인 후 세션 저장함

router.route('/process/photo').post(upload.array('photo', 1), function(req, res) {

    console.log('/process/photo 호출됨.');

    try {

        var files = req.files;

console.dir('#===== 업로드된 첫번째 파일 정보 =====#')

console.dir(req.files[0]);

console.dir('#=====#')

        // 현재의 파일 정보를 저장할 변수 선언

        var originalname = '',

            filename = '',

            mimetype = '',

            size = 0;

        if (Array.isArray(files)) { // 배열에 들어가 있는 경우 (설정에서 1개의 파일도 배열에 넣게 했음)

     console.log("배열에 들어있는 파일 갯수 : %d", files.length);

     for (var index = 0; index < files.length; index++) {

         originalname = files[index].originalname;

         filename = files[index].filename;

         mimetype = files[index].mimetype;

         size = files[index].size;

     }

     } else { // 배열에 들어가 있지 않은 경우 (현재 설정에서는 해당 없음)

     console.log("파일 갯수 : 1 ");

         originalname = files[index].originalname;

         filename = files[index].name;

         mimetype = files[index].mimetype;

         size = files[index].size;

     }

        console.log('현재 파일 정보 : ' + originalname + ', ' + filename + ', '

                + mimetype + ', ' + size);

        // 클라이언트에 응답 전송

        res.writeHead('200', {'Content-Type':'text/html;charset=utf8'});

        res.write('<h3>파일 업로드 성공</h3>');

        res.write('<hr/>');

        res.write('<p>원본 파일명 : ' + originalname + ' -> 저장 파일명 : ' + filename + '</p>');

        res.write('<p>MIME TYPE : ' + mimetype + '</p>');

        res.write('<p>파일 크기 : ' + size + '</p>');

        res.end();

    } catch(err) {

        console.dir(err.stack);

    }

});

정리하면

1. photo.html 에서 input 영역에서 post / enctype을 mulitpart로, 요청패스를 /process/photo로 설정해 파일을 서버로 전달

<form method="post" enctype="multipart/form-data" action="/process/photo">

1. multer의 스토리지/저장 조건은 미리 설정되어 있다

var storage = multer.diskStorage({

destination: function(req, file, callback){

callback(null, 'uploads')

},

filename: function(req, file, callback){

callback(null, file.originalname + Date.now())

}

});

var upload = multer({

storage: storage,

limits:{

files: 10,

fileSize: 1024 \* 1024 \* 1024

}

});

1. 파일 업로드가 실행되면, 요청패스 /process/photo로 설정된 라우팅 함수에서

upload.array를 통해 파일이 배열로 저장이 되고, 해당 파일은 req.files에 저장된다.  
파일 1개가 올라갔으므로 files = req.files로 대입했으니 files[0].originalname 같은 형식으로 콘솔 출력이 가능하다.

참고로 아래 코드만 작성되어 있으면 파일 업로드는 정상 완료된다. App13.js 내 라우팅 코드는 파일 정보 출력을 위한 코드가 대부분!

router.route('/process/photo').post(upload.array('photo', 1), function(req, res) {

    console.log('/process/photo 호출됨.');

});