As of May 31, 2023, we have updated our Code of Conduct.

nodejs multer diskstorage to delete file after saving to disk

Asked 5 years, 3 months ago Modified 1 year, 2 months ago Viewed 45k times

16

I am using multer diskstorage to save a file to disk. I first save it to the disk and do some operations with the file and then i upload it to remote bucket using another function and lib. Once the upload is finished, i would like to delete it from the disk.

```
var storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, '/tmp/my-uploads')
  },
  filename: function (req, file, cb) {
    cb(null, file.fieldname + '-' + Date.now())
  }
})

var upload = multer({ storage: storage }).single('file')
```

and here is how i use it:

```
app.post('/api/photo', function (req, res) {
    upload(req, res, function (err) {
        uploadToRemoteBucket(req.file.path)
        .then(data => {
            // delete from disk first

            res.end("UPLOAD COMPLETED!");
        })
    })
});
```

how can i use the diskStorage remove function to remove the files in the temp folder? https://github.com/expressjs/multer/blob/master/storage/disk.js#L54

update:

I have decided to make it modular and put it in another file:

```
const fileUpload = function(req, res, cb) {
    upload(req, res, function (err) {
        uploadToRemoteBucket(req.file.path)
        .then(data => {
            // delete from disk first

            res.end("UPLOAD COMPLETED!");
        })
    })
}
```

Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up

javascript node.js express multer

Share Follow

edited Mar 4, 2018 at 23:35

asked Mar 4, 2018 at 19:54

jacky
514 1 5 15

5 Answers

Sorted by:

Highest score (default)

48

You don't need to use `multer` to delete the file and besides `_removeFile` is a private function that you should **not** use.

You'd delete the file as you normally would via `fs.unlink`. So wherever you have access to `req.file`, you can do the following:

```
const fs = require('fs')
const { promisify } = require('util')

const unlinkAsync = promisify(fs.unlink)

// ...

const storage = multer.diskStorage({
    destination(req, file, cb) {
      cb(null, '/tmp/my-uploads')
    },
    filename(req, file, cb) {
      cb(null, `${file.fieldname}-${Date.now()}`)
    }
  })

const upload = multer({ storage: storage }).single('file')

app.post('/api/photo', upload, async (req, res) =>{
    // You aren't doing anything with data so no need for the return value
    await uploadToRemoteBucket(req.file.path)

    // Delete the file like normal
    await unlinkAsync(req.file.path)

    res.end("UPLOAD COMPLETED!")
})
```

Share Follow

answered Mar 4, 2018 at 20:10

Cisco
20.4k 5 37 59

thanks for this nice solution. but i have decided to take it out and make it modular. Please see the update. How can i still apply your solution here? thanks for advice – jacky Mar 4, 2018 at 23:36

Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up

with async-await since yesterday before seeing this). – user3773048 Jan 25, 2019 at 4:44

FYI Since **Node 10.23.1** you can use `const fs = require('fs').promises` out of the box. – kokoko Jan 26, 2021 at 6:45

Is it safe to delete files after res.end("UPLOAD COMPLETED!")? because like that response time will improve. – Rajan Mar 19, 2021 at 11:31

You can also use fs.unlinkSync(path), that method does the same thing but synchronously (no need to use promise) – Tarik Merabet Nov 12, 2021 at 21:06

15

Multer isn't needed. Just use this code.

```
const fs = require('fs')

const path = './file.txt'

fs.unlink(path, (err) => {
  if (err) {
    console.error(err)
    return
  }

  //file removed
})
```

Share Follow

answered Aug 7, 2020 at 2:45

sqrepants
986 2 10 22

1 Useful for those who aren't masters of promises and promisify - you'd need to add some customization for promisify and promises which may be confusing if not needed. – J.E.C. Sep 17, 2020 at 9:51

0

You may also consider using MemoryStorage for this purpose, with this storage the file is never stored in the disk but in memory and is deleted from the memory automatically after execution comes out of controller block, i.e., after you serve the response in most of the cases.

When you will use this storage option, you won't get the fields `file.destination`, `file.path` and `file.filename`, instead you will get a field `file.buffer` which as name suggests is a buffer, you can convert this buffer to desired format to do operations on and then upload using a stream object.

Most of the popular libraries support streams so you should be able to use stream to upload your file directly, code for converting buffer to stream:

```
const Readable = require('stream').Readable;
```

Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up

```
stream.push(file.buffer);
stream.push(null);

// now you can pass this stream object to your upload function
```

This approach would be more efficient as files will be stored in memory which will result in faster access, but it does have a con as mentioned in `multer` documentation:

> WARNING: Uploading very large files, or relatively small files in large numbers very quickly, can cause your application to run out of memory when memory storage is used.

Share Follow

answered Jan 9, 2021 at 18:26

Zeus
1,235 12 20

0

To do it truly automatically across all routes I used this strategy :

when the request ends, we delete all the uploaded files (req.files). Before that, if you want to keep the files on the server, you need to save them in another path.

```
var express = require('express');
var app = express();
var http = require('http');
var server = http.Server(app);

// classic multer instantiation
var multer = require('multer');
var upload = multer({
    storage: multer.diskStorage({
        destination: function (req, file, cb) {
            cb(null, `${__dirname}/web/uploads/tmp/`);
        },
        filename: function (req, file, cb) {
            cb(null, uniqid() + path.extname(file.originalname));
        },
    }),
});
app.use(upload.any());

// automatically deletes uploaded files when express finishes the request
app.use(function(req, res, next) {
    var writeHead = res.writeHead;
    var writeHeadbound = writeHead.bind(res);
    res.writeHead = function (statusCode, statusMessage, headers) {
        if (req.files) {
            for (var file of req.files) {
                fs.unlink(file.path, function (err) {
                    if (err) console.error(err);
                });
            }
        }
```

Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up

```
    next();
});

// route to upload a file
    router.post('/profile/edit', access.isLogged(), async function (req, res,
next) {
        try {

// we copy uploaded files to a custom folder or the middleware will delete them
            for (let file of req.files)
                if (file.fieldname == 'picture')
                    await fs.promises.copy(file.path,
`${__dirname}/../uploads/user/photo.jpg`);

        } catch (err) {
            next(err);
        }
    });
```

Share Follow

edited Mar 14, 2021 at 11:06

marc_s
728k 174 1326 1455

answered Mar 9, 2021 at 14:57

Raphael PICCOLO
2,075 1 12 18

0

I have removed directory after file uploaded using fs-extra

```
const fs = require('fs-extra');

// after you uploaded to bucket

await fs.remove('uploads/abc.png'); // remove upload dir when uploaded bucket
```

Share Follow

edited Mar 17, 2022 at 9:36

answered Mar 17, 2022 at 9:29

Saad Ahmed
681 1 8 15

Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up