File upload with multer in node.js and express

vibhuti sharma Follow
Jan 3 · 4 min read

File upload is a very common feature that almost every website needs. File upload in Node.js seems to be tedious work but we have multer to the rescue.
In this article, I will explain how I used multer & express to upload the file.
First thing first what is Multer?

The documentation states:
"Multer is a node.js middleware for handling `multipart/form-data`, which is primarily used for uploading files. It is written on top of busboy for maximum efficiency. Multer adds a `body` object and a `file` or `files` object to the `request` object. The `body` object contains the values of the text fields of the form, the `file` or `files` object contains the files uploaded via the form."

Let's get started

Step 1: Create a directory for the project and use `npm init` to create package.json file.

```
mkdir file-upload-multer
cd file-upload-multer
npm init
```

Step 2: Install the dependencies

```
npm install express multer --save
```

```
C:\work\file-upload-multer  (file-upload-multer@1.0.0)
λ npm install express multer --save
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN file-upload-multer@1.0.0 No description
npm WARN file-upload-multer@1.0.0 No repository field.

+ multer@1.4.1
+ express@4.16.4
added 69 packages in 20.806s
```

Step 3: Create server.js

```
:\work\file-upload-multer  (file-upload-multer@1.0.0)
\ touch server.js
```

Step 4: Create a server using express

Open server.js and type following code.

```
var express = require('express');
const app = express();
const port = 3000;
app.get('/', (req, res) => {
    res.send('hello people');
});
app.listen(port, () => {
    console.log('listening to the port: ' + port);
});
```

run the code using

```
node server.js
```

open the browser and type http://localhost:3000

You should see the following:

Congratulations!! we are half way through :)

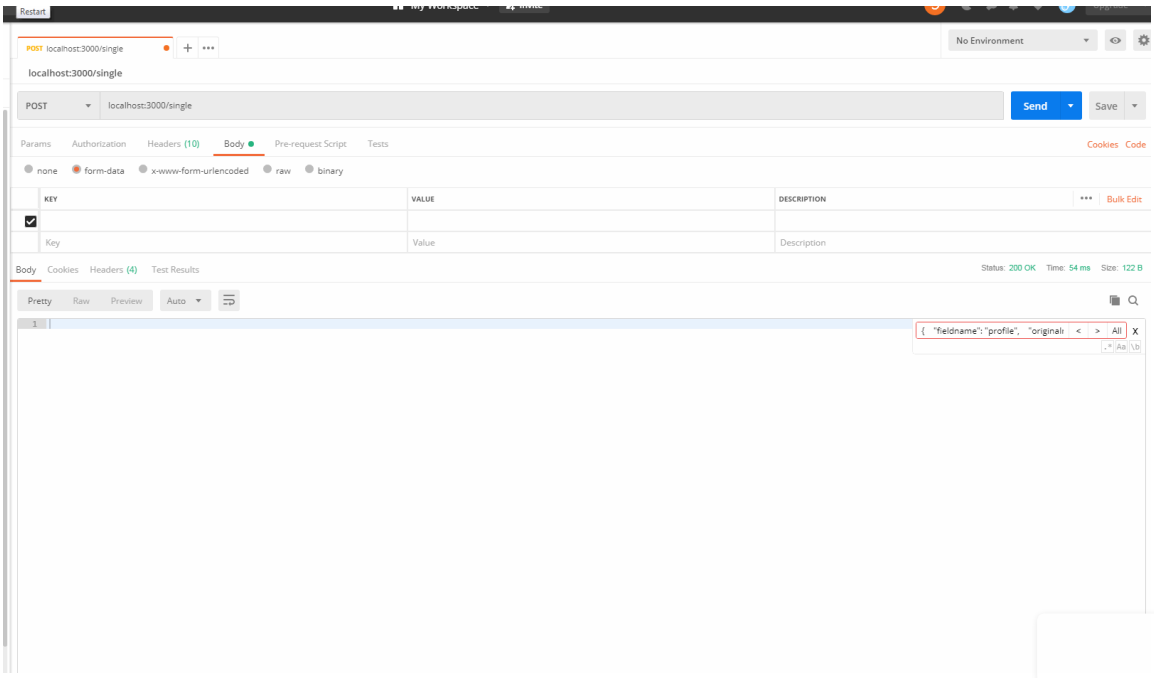
Step 5: Add multer

```
var multer = require('multer');
var upload = multer({dest:'uploads/'});
```

here dest is the path where the file will be stored.Let's create an endpoint which will upload the file.

```
app.post('/single', upload.single('profile'), (req, res) => {
  try {
    res.send(req.file);
  }catch(err) {
    res.send(400);
  }
});
```

Multer supports uploading single as well as multiple uploads. `upload.single` is used for uploading a single file. As I have mentioned before multer adds a file object to the request. The file object contains metadata related with the file.
I am using postman(https://www.getpostman.com/apps] to test this endpoint



Note: key name or the field name should be same as the one provided in `upload.single`

You will notice that the uploads folder is created in the location provided in dest option(in our case in the project directory) but you will notice the uploaded file do not contain any extension. If you want more controls over the uploads we will use the `storage` option instead of `dest` . Multer ships with storage engines `DiskStorage` and `MemoryStorage` .

Step 6: Use Disk Storage

The disk storage engine gives you full control of storing files to disk.we will create a storage object using:

```
var storage = multer.diskStorage({
    destination: function(req, file, cb) {
        cb(null, './upload');
     },
    filename: function (req, file, cb) {
        cb(null , file.originalname);
    }
});
```

and modify the upload variable to

```
var upload = multer({ storage: storage })
```

There are two options available, `destination` and `filename` . They are both functions that determine where the file should be stored.

`destination` is used to determine within which folder the uploaded files should be stored. This can also be given as a `string` (e.g. `'/tmp/uploads'`). If no `destination` is given, the operating system's default directory for temporary files is used. An important thing to note here is we are responsible for creating the directory when providing `destination` as a function. When passing a string, multer will make sure that the directory is created for you.

`filename` is used to determine what the file should be named inside the folder. If no `filename` is given, each file will be given a random name that doesn't include any file extension. Also, Multer will not append any file extension for you, your function should return a filename complete with a file extension.

Each function gets passed both the request (`req`) and some information about the file (`file`) to aid with the decision. I wanted to save the file with same name as they were uploaded so I am using file.originalname. You can have any name you want. On hitting the endpoint again You will notice the files uploaded in the `upload` folder have the same name and have an extension.

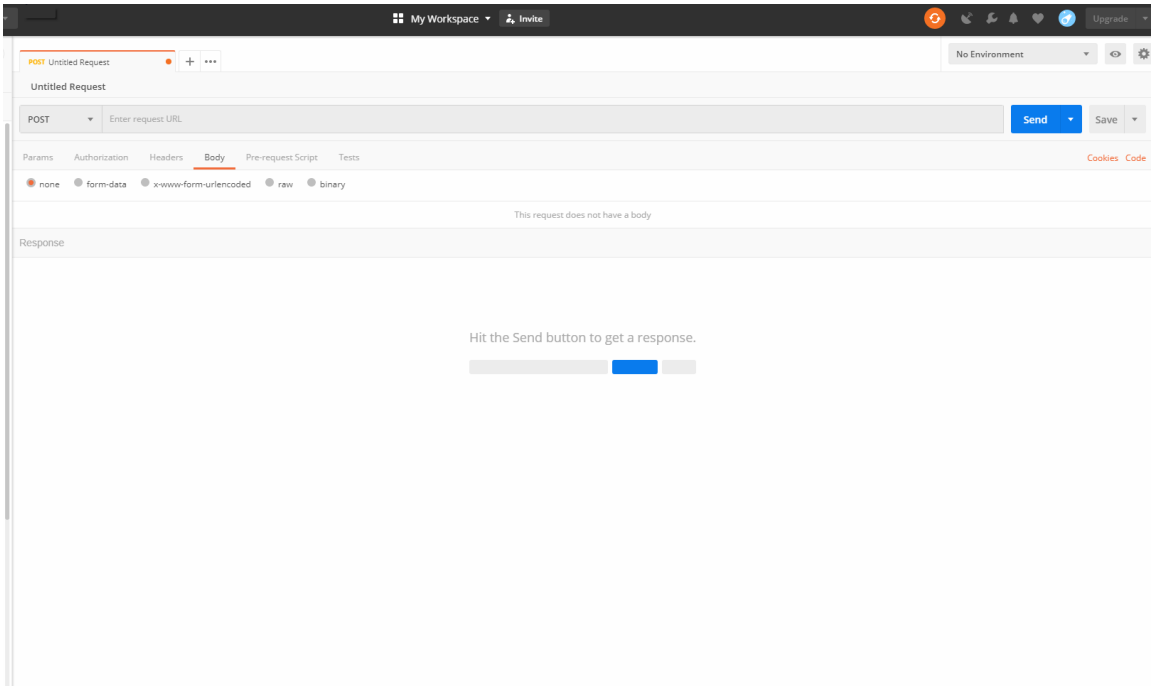
Yaay! with this we have successfully uploaded a single file using multer (Pat on your back).
Similarly, we can upload multiple files using multer. Let’s create an endpoint `bulk` for uploading multiple files.

```
app.post('/bulk', upload.array('profiles', 4) , (req, res) =>{
    try {
        res.send(req.files);
    } catch(error) {
          console.log(error);
```

```
            res.send(400);
        }
    });
```

Here we are using `upload.array` instead of `upload.single` this adds an object `files` in the request object.

on hitting the `bulk` endpoint you should get.



Cheers! with this, we have successfully used multer for single and multiple uploads.

This just a kickstart. You can go through the documentation here https://www.npmjs.com/package/multer .

Have a great day :)

Nodejs Multer Express