Multer ek Node.js middleware hai jo file uploads ke liye use hota hai. Iska use aksar web applications mein hota hai jahan par users files jaise ki images, videos, documents, etc. ko server par upload karte hain. Isse files ko server par receive aur save karne mein madad milti hai.

Isse aap ek server-side application mein use kar sakte hain, taki users files ko client se server par bhej sakein. Multer server par aane wale files ko handle karta hai, jaise ki unhe save karna, unke metadata ko padhna, aur unke access ko manage karna.

Note :  
Multer FormData ko hi accept karta hai

Ismein kuch important concepts hote hain:

1. \*\*Storage\*\*: Multer allows you to specify where the uploaded files should be stored. You can configure it to store files on your server's local filesystem, in a cloud storage service, or even in a database.

2. \*\*File Filtering\*\*: You can set up filters to restrict the types of files that are allowed to be uploaded. For example, you can specify that only image files with certain extensions are accepted.

3. \*\*File Naming\*\*: Multer can generate unique names for uploaded files to prevent naming conflicts.

4. \*\*Limits\*\*: You can set size limits for uploaded files to prevent excessively large files from overloading your server.

5. \*\*Handling Multiple Files\*\*: Multer supports handling multiple files in a single request, making it suitable for scenarios where users need to upload multiple files at once.

6. \*\*Middleware Integration\*\*: It's often used as middleware in Express.js applications to handle file uploads.

Some Steps to follow

1. Upload set karna jaha hamari file save hongi

const upload = multer({ dest: 'uploads/' })

agar hum production ya client ke liye bana rahe hote to cloud storage ka use karte

Iske liye aap specialized file storage and management services ka use kar sakte hain, jaise ki Amazon S3, Google Cloud Storage, Azure Blob Storage, ya kisi dedicated eCommerce platform ke features jo in requirements ko cover karti hain

1. Agar hum local m store karwana chahte hai images, videos etc ko to hume multer Diskstorage banana hoga
2. const storage = multer.diskStorage({
3. destination: function (req, file, cb) {
4. cb(null, '/tmp/my-uploads')
5. },
6. filename: function (req, file, cb) {
7. const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() \* 1E9)
8. cb(null, file.fieldname + '-' + uniqueSuffix)
9. }
10. })

Yaha pe req, files (jo frontend se aa rahi hai) , cb (callback)

Multer ki `diskStorage` configuration mein, `destination` function ka use kiya jata hai taki aap server ke local disk par uploaded files ko kis directory mein store karna chahte hain, aur `cb` parameter ek callback function hai jise aap destination directory provide karne ke liye istemal karte hain

1. cb(null, '/tmp/my-uploads')

yaha pe mtlb hai ki agar koi error nahi hai to file ki path ye set kar do

`destination` function batata hai ki uploaded files ko 'uploads/' directory par server ke local disk par store karna hai. `cb` function do argument leta hai: pehla argument kisi potential error ke liye hota hai (is case mein 'null' hai), aur dusra argument destination directory hai.

`filename` function bhi define ki gayi hai jisse aap uploaded file ka filename customize kar sakte hain. Is example mein, original filename ke sath current timestamp ko append kiya gaya hai.

Aap `destination` function ko customize karke specify kar sakte hain ki aap uploaded files ko server par kis directory mein store karna chahte hain. Yaad rahe ki agar woh directory nahi hai to use pehle create kar lena chahiye.

1. Multer single file bhi accept karta hai aur multiple files bhi

For single

app.post('/profile', upload.single('avatar'), function (req, res, next) {

// req.file is the `avatar` file

// req.body will hold the text fields, if there were any

})

For Multiple

app.post('/photos/upload', upload.array('photos', 12), function (req, res, next) {

// req.files is array of `photos` files

// req.body will contain the text fields, if there were any

})

Note:  
Multer khali formData ki value hi accept karta hai agar hume images save karna hai to uska method hai

To Save Images

Front-End

 const{product,newproduct}=form;

    const [mainImages,newmainImages]=useState([]);

    const setMain=(e)=>{

const files =e.target.files;

const arr=Array.from(files);

arr.forEach((val)=>{

    newmainImages((per)=>[...per,val])

})

    }

// Input file

<input type="file" name="" id="file" multiple onChange={setMain} />

yaha pe humne inputs ko multiple file accept karwai hai aur onchange pe function call kiya jisme humne inputs ki saari files ko mainImages ke ander set karwa rahe hai lekin

jub humne e.target.files se file li to wo ek array ki tarh dikhne wala array aagya lekin ye proper array nahi hai ise hume array m convert karna hoga

to humne Array.from() ka use karke usko array m convrt kiya fir usko mainImages m add kiya

ise kya hoga ki hume ek array mil jayega isko hume formData m dalna hoga formData m direct array send nahi kar sakte hai aur multer m file bhejni hai to

saari files ko alag alag bhejana hoga lekin unka name same hoga

ab hume mainImages se usko form Data m dalna hoga

const data=new FormData();

product.files.forEach((val)=>{

    data.append('photo',val)

  })

Yaha pe humne mainImages ko product.files m save karwa hai aur ab hum usme loop chala ke ek ek karke array ki value ko data m append karke add karwange saari photo ka name same hoga isse multer jiska bhi name photo hai use save karega folder

Multer ese hi accept karta hai

Short

First input se file accept karwani hai --------🡪

Accept file array nahi hogi proper to usse hum naya array banyenge Array.From()-🡪

Array bane baad array ki ek ek val ko formData m aapend karwnge

Form.append(“photo”,val);

Yaha pe photo name sabhi val ke liye same rahega kyuki multer file ke saari file name ko ise hi identify karega

Form Back-End

router.post('/p',upload.array('photo'),EntryProduct);

yaha pe multer formData m photo name ke saare search karega fir unko save kar lega

Note: ye sab hone baad ab hume moongoose m photo ke path ko save karwana hoga req.files m image ka data mil jayenge agr image jada hai to array m data mil jayega

For Content

FormData only accepts string for content agar hume array jese value save karwani ho to FromData se phle string ko m backend m bhejo fir usko waha string se array m convert karo direct array nahi jayega

Because FormData string accept karta hai

Do tarike



Let assume color m humne ye value get ki , ‘red,blue,green’ from input text se

To ab hum isse array m convert frontend m na karke backend m convert karenge

Frontend – direct humne string m bhej diya FormData ke madad se

data.append("color",product.color);

Backend- yaha pe hum color ke string ko array m convert karenge

Agar hume object m karna ho ya kise m bhi to kar sakhte hai abhi hum array ka dekhte hai

const {name,price,brand,model,network,operating,cellular,discription,color,ram,storage}=req.body;

const image=req.files;

phle humne saari files ko get karwaya ab

const colors=req.body.color;

const rams=req.body.ram;

const storages=req.body.storage;

jin jin ko array m ya kisi aur cheej m convrt karna hai to unko hum ek variable m store karwange kyuki direct m req.body ki value ka use karke change nahi kar sakhte hai

Variables m store karwane ke baad ab hum string ko convert karenge array m split property ki madad se

const colorarr=colors.split(/\W+/);

const ramArr=rams.split(/\W+/);

const storageArr=storages.split(/\W+/);

ab inko hum model m save karwa sakhte hai

 const newOne=new Product({

    images:image,

    name,

    price,

    brand,

    model,

    network,

    operating,

    cellular,

    discription,

    color:colorarr,

    ram:ramArr,

    storage:storageArr

  }).save();

1. Isme m frontend m hi convert kar lo string se array m fir usko JSON.Stringify laga ke string m convert kar lo fir usko backend m bhej ke JSON.Parser laga ke wapis se original m convert kar lo

Get Images

Images ko get karne ke liye phle ek form banake backend se data mangwao usme image path hogi

Backend m uploads folder ko express static m link karo jise hum use access kar sake

app.use('/uploads',express.static('uploads'))

frontend m data mangao

 const[data,newdata]=useState(null);

const cla=async()=>{

const res=await axios.post(`${import.meta.env.VITE\_BACKEND\_URL}/plogin`,{model});

console.log(res)

if(res){

  newdata(res.data.exsit)

}

}

/\* import.meta.env.VITE\_BACKEND\_URL == http://localhost:8000/ \*/

Yaha pe response ko data m store kiya

<div>

  {

data ? <div><img src={`${import.meta.env.VITE\_BACKEND\_URL}/${data.images[0].path}`} alt="ss" className='w-[50vw] h-[40vh]' /> </div>:""

  }

</div>

/\* data.images[0].path == uploads\1697653946244-391197925-1.webp \*/

Yaha pe hum images ko access karenge backend se backend ka url and uploads \ and image name

Express.static se humne us folder ke liye ek url bana jise hum backend ka url and static folder ka link set karenge to data access kar sakhte hai

Example:-

app.use('/images',express.static('uploads'))

waha pe humne jo uploads name ka folder hai backend m uska yaha humne url set kiya /images naam ka

agar hume access karna hai to

[http://localhost:8000/ images/ {image](http://localhost:8000/%20images/%20%7bimage) ka name}

Images compress kar ke multer m store kare jise ki space kam le

Multer file upload middleware directly doesn't provide a built-in feature for compressing images. To compress images when using Multer for file uploads, you can use additional libraries such as `sharp`, `imagemin`, or `gm` to handle image compression.

Here's how you can use the `sharp` library to compress images in a Node.js application using Multer:

1. First, you need to install the `sharp` library using npm or yarn:

```bash

npm install sharp

# OR

yarn add sharp

```

2. Import `sharp` and set up your Multer middleware for handling file uploads.

```javascript

const express = require('express');

const multer = require('multer');

const sharp = require('sharp');

const app = express();

const storage = multer.memoryStorage(); // Using memory storage for Multer

const upload = multer({ storage });

app.post('/upload', upload.single('image'), (req, res) => {

const { buffer } = req.file;

sharp(buffer)

.resize({ width: 200, height: 200 }) // Adjust the width and height as needed

.toBuffer((err, data, info) => {

if (err) {

return res.status(500).send('Image processing error');

}

// You can save or send the compressed image data as needed.

// For example, save it to a file or send it in the response.

res.send(data);

});

});

app.listen(3000, () => {

console.log('Server is running on port 3000');

});

```

In this example, we've used Multer to handle file uploads. After the image is uploaded, we use the `sharp` library to resize and compress the image. The compressed image data can then be saved, sent in a response, or handled as per your application's requirements.

Make sure to adjust the resizing parameters and error handling based on your specific needs. This example demonstrates image compression, but you can modify it to save the compressed images to your desired location or send them as a response to the client.

Video ko Compress kare

Haan, aap video ko compress karke `multer` ka istemal karke save kar sakte hain. `multer` Node.js ke ek popular middleware package hai jo server-side file uploads ko handle karta hai. Video compress karne ke liye aap FFmpeg jaise tools ka istemal kar sakte hain.

Yahan ek basic example diya gaya hai kaise aap video ko compress karke `multer` ka istemal karke save kar sakte hain:

1. Sabse pehle, `multer` aur `ffmpeg` (video compression ke liye) ko install karein:

```

npm install multer fluent-ffmpeg

```

2. Apna Node.js server banayein aur `multer` ko configure karein:

```javascript

const express = require('express');

const multer = require('multer');

const ffmpeg = require('fluent-ffmpeg');

const app = express();

const storage = multer.memoryStorage(); // Save the file in memory

const upload = multer({ storage });

app.post('/upload', upload.single('video'), (req, res) => {

// Get the uploaded video buffer

const videoBuffer = req.file.buffer;

// Compress the video using FFmpeg

ffmpeg()

.input(videoBuffer)

.outputOptions(['-c:v libx264', '-crf 20', '-preset fast'])

.toFormat('mp4')

.on('end', () => {

console.log('Video compression finished.');

})

.pipe(res, { end: true });

});

app.listen(3000, () => {

console.log('Server is running on port 3000');

});

```

Is example me, `multer` se video file ko server par upload kia jata hai, phir FFmpeg ka istemal karke video ko compress kia jata hai, aur fir compressed video ko response me bheja jata hai.

Aap ise customize kar sakte hain, jaise ki compression settings, storage location, aur endpoints. Additionally, aap error handling bhi jod sakte hain.

Har User ke liye alag alag folder

Ji, aap ek multer middleware ka istemal karke har ek user ke liye alag-alag upload folder bana sakte hain. Multer ek Node.js middleware hai, jiska upayog aksar file uploads ke liye hota hai. Aap ise aise configure kar sakte hain ki har ek user ke liye alag-alag upload folders banaye.

Yahan ek udaharan hai kaise aap isko kar sakte hain:

1. Sabse pehle, aapko multer ko apne Node.js prakriya mein sahayak middleware ke roop mein samil karna hoga. Aap ise npm se install kar sakte hain:

```bash

npm install multer

```

2. Ab, aapko multer ka upayog karte samay har ek user ke liye alag-alag upload folder banane ke liye ek custom destination function ka upayog karna hoga. Yeh ek udaharan hai:

```javascript

const express = require('express');

const multer = require('multer');

const app = express();

const storage = multer.diskStorage({

destination: (req, file, cb) => {

const userId = req.user.id; // Assume you have a user object with an "id" property

const userUploadsFolder = `uploads/${userId}`;

cb(null, userUploadsFolder);

},

filename: (req, file, cb) => {

cb(null, Date.now() + '-' + file.originalname);

},

});

const upload = multer({ storage: storage });

app.post('/upload', upload.single('file'), (req, res) => {

res.send('File uploaded');

});

app.listen(3000, () => {

console.log('Server is running on port 3000');

});

```

Is udaharan mein, har ek user ke liye ek alag upload folder banaane ke liye `destination` function ka upayog kiya gaya hai. `req.user.id` ki jagah aap aapke application ke user authentication ke hisab se user ID ko istemal kar sakte hain.

Dhyan rahe ki aapko `uploads` folder ko pehle se banaya hua hona chahiye, aur sahi permissions ke saath hona chahiye taki multer usmein files save kar sake.

Is tarah se aap multer ka upayog karke har ek user ke liye alag-alag upload folders bana sakte hain.