Multer

Multer is a middleware used for handling **multipart/form-data**, primarily used for uploading files in **Node.js and Express**.

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.

NOTE: *Multer will not process any form which is not multipart (multipart/form-data).*

- **Why Use Multer?**
  - ✅ Handles file uploads efficiently in Express.js.
  - ✅ Supports **single and multiple** file uploads.
  - ✅ Stores files **on disk (local)** or in **memory (buffer)**.
  - ✅ Works well with **database storage (MongoDB/GridFS, AWS S3, Cloudinary, etc.)**.

- **Install Multer**
  Run the following command to install Multer:
  npm install multer

- **Import Multer in Your Project**
  const multer = require("multer");
  const path = require("path");

- **Multer Storage Configuration**
  Multer provides two types of storage:
  1. **Disk Storage** → Saves the file on the server.
  2. **Memory Storage** → Stores the file in RAM as a buffer.

1. **Disk Storage (Saving Files on Server)**

```
const storage = multer.diskStorage({
   destination: function (req, file, cb) {
      cb(null, 'public/images/')
   },
   filename: function (req, file, cb) {
      let ext = path.extname(file.originalname)
      cb(null, file.fieldname + Date.now() + ext);
    }
})

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


- **destination Function**

```
destination: function (req, file, cb) {
      cb(null, 'public/images/')
   }
```

The **destination** function decides where to **store** the uploaded file.

It takes **three arguments**:

1. req → Request Object (contains form data and other request details).
2. file → File Object (contains file-related details like name, type, size, etc.).
3. cb (Callback) → A function used to **return** the destination path.

✅ cb(null, "'public/images/") → **Saves the file in the uploads/ folder.**

💡 **Tip:** Make sure the uploads/ folder **exists** before running the code.

- **filename Function**

```
filename: function (req, file, cb) {
  let ext = path.extname(file.originalname)
  cb(null, file.fieldname + Date.now() + ext);
 }
```

The **filename** function decides **how the file will be named** when saved.

It takes **three arguments**:

1. req → Request Object.
2. file → File Object.
3. cb → Callback function.

✅ file.originalname → **Gets the original name of the uploaded file**

✅ path.extname(file.originalname) → **Extracts the file extension (e.g., .jpg, .png)**

✅ file.fieldname + Date.now() + ext → **Creates a unique filename**

2. Memory Storage (Saving Files in Buffer)

```
const storage = multer.memoryStorage();
const upload = multer({ storage: storage });
```

✅ This stores the file in memory instead of saving it to disk.

✅ Useful when uploading files to **Cloud Storage (AWS, Cloudinary, Firebase, etc.)**.

- **Handling File Uploads in Express.js**

(A) Upload Single File

```
app.post("/upload", upload.single("image"), (req, res) => {
  res.json({ file: req.file });
});
```

✅ upload.single("image") → Uploads a single file with the field name "image".

(B) <u>Upload Multiple Files</u>

```
app.post("/upload-multiple", upload.array("images", 5), (req, res) => {
  res.json({ files: req.files });
});
```

✅ upload.array("images", 5) → Allows uploading up to **5** files at once.

- **<u>Serving Uploaded Files as Static Files</u>**

  To access uploaded images from the frontend, make the **uploads** folder static:

  ```
  app.use(express.static("public"));
  ```