**Multer**

**Multer** is a middleware for handling multipart/form-data, which is used for uploading files in Node.js applications. It is primarily used in combination with **Express** to handle incoming file uploads.

**Key Features of Multer:**

* **File Uploads**: Multer allows you to handle file uploads, either to memory (RAM) or to disk (your server's filesystem).
* **File Filtering**: You can filter which files are allowed to be uploaded based on file types, sizes, etc.
* **Handling Multiple Files**: You can upload multiple files at once.
* **Error Handling**: Provides built-in support for handling errors during file uploads, such as file size limit exceeded.

Installation:

npm i multer

**Let’s see Real world example and explaination**

import multer from "multer";

const storage = multer.diskStorage({

    destination: function (req, file, callBackFunction) {

        callBackFunction(null, "./public/temp");

    },

    filename: function (req, file, callBackFunction) {

        callBackFunction(null, file.originalname);

        console.log("FROM MULTER FILENAME FUNCTION : ::", file);

    },

});

export const upload = multer({ storage });

This code defines a file upload handler using the **Multer** middleware in Node.js. It configures Multer to store uploaded files on the local disk with specific settings for their **destination** and **filename**.

### ****Code Breakdown****

1. **Import Multer**

import multer from "multer";

* + The multer library is imported to handle file uploads.

1. **Storage Configuration**

const storage = multer.diskStorage({

// where to store the file

    destination: function (req, file, callBackFunction) {

        callBackFunction(null, "./public/temp");

    },

// what will be the file name

    filename: function (req, file, callBackFunction) {

        callBackFunction(null, file.originalname);

        console.log("FROM MULTER FILENAME FUNCTION : ::", file);

    },

});

export const upload = multer({ storage });

* + **multer.diskStorage()**: Defines how and where to store uploaded files.

#### destination **Function**

* + **Purpose**: Specifies the folder where uploaded files will be stored.
  + **Parameters**:
    - req: The incoming request object.
    - file: Metadata about the uploaded file (e.g., original name, MIME type).
    - callBackFunction: A callback to handle errors or set the destination.
  + **Code**:
    - Saves files in the ./public/temp directory.
    - callBackFunction(null, "./public/temp"): Indicates no error (null) and sets the destination.

#### filename **Function**

* + **Purpose**: Defines how the uploaded files will be named.
  + **Parameters**:
    - Same as the destination function.
  + **Code**:
    - Uses the file's original name (file.originalname) as the new name.
    - Example file object:

{

fieldname: 'file',

originalname: 'example.jpg',

encoding: '7bit',

mimetype: 'image/jpeg'

}

1. **Multer Middleware Instance**

export const upload = multer({ storage });

* + **multer({ storage })**: Creates a middleware instance configured with the defined storage object.
  + **Exported upload**: Can be used in routes to handle file uploads.

### ****How It Works****

When a file upload request is processed:

1. The file is saved to the ./public/temp directory.
2. The file's name remains the same as the original name (file.originalname).