MulterOptions

August 17, 2026

MulterOptions

Kind: Interface

Source: `packages/platform-express/multer/interfaces/multer-options.interface.ts`

Part of: Platform Express

`MulterOptions` configures how NestJS's Express platform integrates with Multer for handling `multipart/form-data` uploads. It controls upload destinations, storage engines, upload limits, path preservation, and the default character set used when parsing form data.

Properties

Property	Type
`dest`	`string
`storage`	`any`
`limits`	`{ fieldNameSize?: number; fieldSize?: number; fields?: number; fileSize?: number; files?: number; parts?: number; headerPairs?: number; }`
`preservePath`	`boolean`
`defParamCharset`	`string`

Diagram

```
graph LR
  Client[Multipart Request] --> Interceptor[File Upload Interceptor]
  Interceptor --> Options[MulterOptions]
  Options --> Storage[storage or dest]
  Options --> Limits[limits]
  Options --> Parsing[preservePath and defParamCharset]
```

```
  Storage --> Files[Stored Uploaded Files]
  Limits --> Validation[Upload Constraints]
```

Usage

```
import { Module } from '@nestjs/common';
import { MulterModule } from '@nestjs/platform-express';
import { diskStorage } from 'multer';
import type { MulterOptions } from '@nestjs/platform-express/multer/interfaces/multer-options.i

const multerOptions: MulterOptions = {
  storage: diskStorage({
    destination: './uploads',
    filename: (_request, file, callback) => {
      callback(null, `${Date.now()}-${file.originalname}`);
    },
  }),
  limits: {
    fileSize: 5 * 1024 * 1024, // 5 MB
    files: 3,
  },
  preservePath: false,
  defParamCharset: 'utf8',
};

@Module({
  imports: [MulterModule.register(multerOptions)],
})
export class AppModule {}
```

AI Coding Instructions

- Prefer `storage` with a Multer storage engine when filename generation or destination handling requires custom behavior.
- Use `dest` only for simple disk-based uploads; do not configure both `dest` and a custom `storage` engine unless the underlying Multer behavior is explicitly intended.
- Always define `limits`, especially `fileSize` and `files`, to prevent excessive resource consumption from upload requests.
- Keep `preservePath` disabled unless preserving client-provided directory paths is required and has been reviewed for security implications.
- Configure these options through `MulterModule.register()` globally or `MulterModule.registerAsync()` when settings depend on injected configuration.