UploadedFile

August 18, 2026

UploadedFile

Kind: Function

Source: `packages/common/decorators/http/route-params.decorator.ts`

Part of: Common

Route handler parameter decorator. Extracts the `file` object and populates the decorated parameter with the value of `file`. Used in conjunction with multer middleware for Express-based applications.

For example:

```
uploadFile(@UploadedFile() file) {
  console.log(file);
}
```

`@UploadedFile()` is a route handler parameter decorator that reads the uploaded file object from the request and assigns it to the decorated parameter. In Express applications, it works with Multer middleware or a Nest file interceptor that places a file on the request.

Signature

```
function UploadedFile(fileKey: string | (Type<PipeTransform> | PipeTransform), pipes: (Type<Pip
```

Parameters

Name	Type
`fileKey`	`string
`pipes`	`(Type

Returns: `ParameterDecorator`

Diagram

```
graph LR
  Client[Client multipart request] --> Middleware[Multer or FileInterceptor]
  Middleware --> Request[Request with file object]
  Request --> Decorator[@UploadedFile()]
  Decorator --> Handler[Route handler parameter]
```

Usage

```
import {
  Controller,
  Post,
  UploadedFile,
  UseInterceptors,
} from '@nestjs/common';
import { FileInterceptor } from '@nestjs/platform-express';

@Controller('uploads')
export class UploadController {
  @Post()
  @UseInterceptors(FileInterceptor('file'))
  uploadFile(@UploadedFile() file: Express.Multer.File) {
    return {
      filename: file.filename,
      mimetype: file.mimetype,
      size: file.size,
    };
  }
}
```

AI Coding Instructions

- Pair `@UploadedFile()` with Multer middleware or `FileInterceptor()` so the request contains a file object.
- Match the field name passed to `FileInterceptor('file')` with the multipart form field sent by the client.
- Handle missing files when the upload field is optional or middleware may reject the request.
- Add validation pipes or file checks when routes need to restrict file type or size.