ParseFilePipeBuilder

August 18, 2026

ParseFilePipeBuilder

Kind: Class

Source: `packages/common/pipes/file/parse-file-pipe.builder.ts`

Part of: Common

`ParseFilePipeBuilder` provides a fluent API for configuring and creating a `ParseFilePipe` for uploaded-file validation. It collects file validators—such as maximum size and allowed file type—and builds a pipe that can be applied to `@UploadedFile()` parameters in NestJS controllers.

Methods

Method	Signature	Returns
`addMaxSizeValidator`	`addMaxSizeValidator(options: MaxFileSizeValidatorOptions)`	`void`
`addFileTypeValidator`	`addFileTypeValidator(options: FileTypeValidatorOptions)`	`void`
`addValidator`	`addValidator(validator: FileValidator)`	`void`
`build`	`build(additionalOptions: Omit<ParseFileOptions, 'validators'>)`	`ParseFilePipe`

Diagram

```
graph LR
  A[ParseFilePipeBuilder] --> B[addMaxSizeValidator]
  A --> C[addFileTypeValidator]
  A --> D[addValidator]
  B --> E[Validator collection]
  C --> E
  D --> E
  E --> F[build]
```

```
  F --> G[ParseFilePipe]
  G --> H[@UploadedFile() validation]
```

Usage

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

@Controller('uploads')
export class UploadController {
  private readonly imageUploadPipe = new ParseFilePipeBuilder()
    .addFileTypeValidator({
      fileType: /(jpg|jpeg|png)$/,
    })
    .addMaxSizeValidator({
      maxSize: 5 * 1024 * 1024, // 5 MB
    })
    .build({
      errorHttpStatusCode: HttpStatus.UNPROCESSABLE_ENTITY,
    });

  @Post('image')
  @UseInterceptors(FileInterceptor('file'))
  uploadImage(@UploadedFile(this.imageUploadPipe) file: Express.Multer.File) {
    return {
      filename: file.filename,
      mimetype: file.mimetype,
      size: file.size,
    };
  }
}
```

AI Coding Instructions

- Use the fluent builder chain to add validators before calling `build()`; `build()` returns the `ParseFilePipe` used by controller parameters.
- Prefer `addMaxSizeValidator()` and `addFileTypeValidator()` for standard upload constraints; use `addValidator()` for custom `FileValidator` implementations.

- Configure size limits in bytes and ensure file-type rules match the formats accepted by the upload endpoint.
- Pass pipe-level options, such as `errorHttpStatusCode` or `fileIsRequired`, to `build()` rather than embedding them in individual validators.
- Reuse a built pipe when multiple endpoints share the same validation policy, but create separate builders for endpoint-specific rules.

Used by

1 reference from 1 file. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (1)

- `AppController` — `sample/29-file-upload/src/app.controller.ts`:15