Common

July 31, 2026

Common

What it is responsible for

Common manages shared framework contracts and input-oriented transformations for Nest applications. Its documented scope includes decorators such as `@Controller()` and `@Injectable()`, while named pipes turn supplied values into parsed or validated forms. `DefaultValuePipe`, `ParseFilePipe`, `ParseArrayPipe`, and `ParseBoolPipe` identify entry points for that work; `ValidationPipe` marks validation as part of the same surface. The subsystem also defines the middleware-application contract: `MiddlewareConsumer` applies user-defined middleware to routes, alongside `MiddlewareConfiguration`, `RouteInfo`, and `NestMiddleware`. The named `ConsoleLogger` and `Logger` place logging within this shared surface.

What it needs, and who needs it

Common needs no dependencies from this repository. The listed consumers—`integration/injector/src/circular-modules`, `integration/injector/src/circular-properties`, `integration/inspector/src/circular-modules`, `integration/scopes/src/inject-inquirer`, and `Integration`—need Common's contracts and pipe symbols. Without Common, those consumers cannot resolve the subsystem they depend on; references to its decorators, middleware types, logging symbols, serializers, or pipes would be unavailable. No repository dependency is named for Common.

Notable members

`DefaultValuePipe` is an identified entry point and, by its name, handles default values. `ParseArrayPipe` is another entry point and names array parsing directly. `MiddlewareConsumer` carries the explicitly documented responsibility: it defines a method for applying user-defined middleware to routes.

353 entities in `packages/common`. **5 other subsystems depend on it**, which makes it the 2nd most depended-upon part of this codebase.

What it is made of

Its 353 entities sit in 161 files under `packages/common`: 92 interfaces, 85 functions, 63 constants, 32 type aliases and 81 more.
`route-params.decorator.ts` holds 36 of them — more than any other file here.
`HttpException` declares 12 methods, the widest surface here.

Where work enters

- `DefaultValuePipe` — `packages/common/pipes/default-value.pipe.ts`:1
- `ParseFilePipe` — `packages/common/pipes/file/parse-file.pipe.ts`:1
- `ParseArrayPipe` — `packages/common/pipes/parse-array.pipe.ts`:1
- `ParseBoolPipe` — `packages/common/pipes/parse-bool.pipe.ts`:1

How it refuses and fails

6 of its components record a refusal or a failure handler.
All 6 of them refuse work outright, under a condition written into the component itself.
Their `catch` blocks handle a failure that already happened in 2 places.
Of those 2, 1 logs it and continues and 1 turns it into a return value.

Boundaries

5 other subsystems depend on this one — `Circular Modules`, `Circular Properties`, `Circular Modules`, `Inject Inquirer`, `Integration`. Changing what it exposes changes them.

Those 5 hold 11 edges between them, unevenly: `Inject Inquirer` reaches in across 3 edges, against 2 from `Integration`. What they reach is narrower than the folder: 2 of its 353 members carry every inbound edge — `forwardRef` (8) and `Logger` (3).

It depends on no other subsystem in this repository — it is a leaf.

How this code is named

These conventions cover most of the codebase. Learning them is faster than reading an index —
each one lets you find any member of its family without looking it up.

Pattern	Where	Count	Examples
`*.interface.ts`	across the repository	67	`type.interface.ts`, `file.interface.ts`, `on-init.interface.ts`, `abstract.interface.ts`
`*.decorator.ts`	across the repository	24	`sse.decorator.ts`, `bind.decorator.ts`, `catch.decorator.ts`, `inject.decorator.ts`
`*.exception.ts`	`packages/common/exceptions/`	23	`gone.exception.ts`, `http.exception.ts`, `conflict.exception.ts`, `forbidden.exception.ts`
`*.util.ts`	across the repository	16	`cli-colors.util.ts`, `forward-ref.util.ts`, `is-log-level.util.ts`, `load-package.util.ts`