Reflector

August 18, 2026

Reflector

Kind: Class

Source: `packages/core/services/reflector.service.ts`

Part of: Core

Helper class providing Nest reflection capabilities.

`Reflector` is Nest's metadata access service for creating typed decorators and reading metadata attached to classes and methods. It is commonly injected into guards, interceptors, and other framework components to make runtime decisions based on decorator configuration.

Methods

Method	Signature	Returns
`createDecorator`	`createDecorator(options: CreateDecoratorOptions<TParam>)`	`ReflectableDecorator<TParam>`
`createDecorator`	`createDecorator(options: CreateDecoratorWithTransformOptions<TParam, TTransformed>)`	`ReflectableDecorator<TParam, TTransformed>`
`createDecorator`	`createDecorator(options: CreateDecoratorOptions<TParam, TTransformed>)`	`ReflectableDecorator<TParam, TTransformed>`
`get`	`get(decorator: T, target: Type	Function)`
`get`	`get(metadataKey: TKey, target: Type	Function)`
`get`	`get(metadataKeyOrDecorator: TKey, target: Type	Function)`
`getAll`	`getAll(decorator: ReflectableDecorator<TParam, TTransformed>, targets: (Type	Function)[])`
`getAll`	`getAll(metadataKey: TKey, targets: (Type	Function)[])`

Method	Signature	Returns
`getAll`	`getAll(metadataKeyOrDecorator: TKey, targets: (Type	Function)[])`
`getAllAndMerge`	`getAllAndMerge(decorator: ReflectableDecorator<TParam, TTransformed>, targets: (Type	Function)[])`
`getAllAndMerge`	`getAllAndMerge(metadataKey: TKey, targets: (Type	Function)[])`
`getAllAndMerge`	`getAllAndMerge(metadataKeyOrDecorator: TKey, targets: (Type	Function)[])`
`getAllAndOverride`	`getAllAndOverride(decorator: ReflectableDecorator<TParam, TTransformed>, targets: (Type	Function)[])`
`getAllAndOverride`	`getAllAndOverride(metadataKey: TKey, targets: (Type	Function)[])`
`getAllAndOverride`	`getAllAndOverride(metadataKeyOrDecorator: TKey, targets: (Type	Function)[])`

Where it refuses work

- `Reflector` stops the work with an early return when `isEmpty(metadataCollection)`.
- `Reflector` stops the work with an early return when `isObject(value)`.
- `Reflector` stops the work with an early return when `Array.isArray(a)`.
- `Reflector` stops the work with an early return when `isObject(a) && isObject(b)`.
- `Reflector` stops the work with an early return when `result !== undefined`.

Diagram

```
graph LR
  D[Reflector.createDecorator] --> M[Metadata decorator]
  M --> C[Controller class]
  M --> H[Route handler]
  C --> R[Reflector service]
  H --> R
  R --> G[Guard / Interceptor]
  G --> A[Authorization or runtime behavior]
```

Usage

```ts
import {
  CanActivate,
  ExecutionContext,
  Injectable,
} from '@nestjs/common';
import { Reflector } from '@nestjs/core';

// Create a typed decorator for attaching role metadata.
export const Roles = Reflector.createDecorator<string[]>();

@Injectable()
export class RolesGuard implements CanActivate {
  constructor(private readonly reflector: Reflector) {}

  canActivate(context: ExecutionContext): boolean {
    const roles = this.reflector.getAllAndMerge(Roles, [
      context.getClass(),
      context.getHandler(),
    ]);

    const user = context.switchToHttp().getRequest().user;

    return roles.length === 0 || roles.some((role) => user.roles?.includes(role));
  }
}

// Usage in a controller:
// @Roles(['admin'])
// @Get('reports')
// getReports() {}
```

AI Coding Instructions

- Prefer `Reflector.createDecorator<T>()` for new metadata keys so decorators and metadata reads remain type-safe.
- Inject `Reflector` through Nest dependency injection; do not manually instantiate it in guards or interceptors.
- Pass both `context.getClass()` and `context.getHandler()` to `getAll()` or `getAllAndMerge()` when metadata may be defined at either scope.
- Use `getAllAndMerge()` for cumulative metadata such as role or permission arrays; ensure the consuming code handles empty results.
- Keep decorator metadata serializable and lightweight, since it is evaluated during request handling.

Used by

12 references from 7 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Injected or called by (5)

- `RolesGuard` — `integration/inspector/src/common/guards/roles.guard.ts`:4
- `RolesGuard` — `sample/01-cats-app/src/common/guards/roles.guard.ts`:5
- `RolesGuard` — `sample/10-fastify/src/common/guards/roles.guard.ts`:4
- `AuthGuard` — `sample/19-auth-jwt/src/auth/auth.guard.ts`:13
- `RolesGuard` — `sample/36-hmr-esm/src/common/guards/roles.guard.ts`:5

Imported by (7)

- `RolesGuard` — `integration/inspector/src/common/guards/roles.guard.ts`:4
- `RolesGuard` — `sample/01-cats-app/src/common/guards/roles.guard.ts`:5
- `RolesGuard` — `sample/10-fastify/src/common/guards/roles.guard.ts`:4
- `AuthGuard` — `sample/19-auth-jwt/src/auth/auth.guard.ts`:13
- `RolesGuard` — `sample/36-hmr-esm/src/common/guards/roles.guard.ts`:5
- `Roles` — `sample/01-cats-app/src/common/decorators/roles.decorator.ts`:3
- `Roles` — `sample/36-hmr-esm/src/common/decorators/roles.decorator.ts`:3