SetMetadata

August 18, 2026

SetMetadata

Kind: Function

Source: `packages/common/decorators/core/set-metadata.decorator.ts`

Part of: Common

Decorator that assigns metadata to the class/function using the specified `key`.

Requires two parameters:

- `key` - a value defining the key under which the metadata is stored
- `value` - metadata to be associated with `key`

This metadata can be reflected using the `Reflector` class.

Example: `@SetMetadata('roles', ['admin'])`

`SetMetadata` is a decorator factory that attaches custom metadata to a class or method under a specified key. NestJS components, such as guards and interceptors, can later read this metadata through `Reflector` to drive runtime behavior like authorization or feature configuration.

Signature

```
function SetMetadata(metadataKey: K, metadataValue: V): CustomDecorator<K>
```

Parameters

Name	Type
`metadataKey`	`K`
`metadataValue`	`V`

Returns: `CustomDecorator<K>`

Diagram

```
graph LR
  A["@SetMetadata(key, value)"] --> B["Class or method"]
  B --> C["Reflect metadata storage"]
  C --> D["Reflector"]
  D --> E["Guard / interceptor / application logic"]
```

Usage

```
import { Controller, Get, SetMetadata } from '@nestjs/common';
import { Reflector } from '@nestjs/core';

export const ROLES_KEY = 'roles';

@Controller('users')
export class UsersController {
  @Get()
  @SetMetadata(ROLES_KEY, ['admin'])
  findAll() {
    return [];
  }
}

// Example usage inside a guard:
const roles = reflector.get<string[]>(ROLES_KEY, context.getHandler());
```

AI Coding Instructions

- Use stable, descriptive metadata keys; export key constants when multiple files need to read the same metadata.
- Apply `SetMetadata` to classes for controller-wide configuration or methods for route-specific configuration.
- Read metadata through Nest's `Reflector`, typically using `getAllAndOverride` when both class- and method-level values should be supported.
- Keep metadata values serializable and predictable, such as strings, arrays, or configuration objects.
- Avoid using duplicate generic string keys across unrelated features, as metadata keys share the decorated target's metadata store.

Used by

7 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.

Imported by (7)

- `CreateDecoratorOptions` — `packages/core/services/reflector.service.ts`:8
- `FilterByInclude` — `packages/core/discovery/discovery-service.ts`:16
- `RouteConfig` — `packages/platform-fastify/decorators/route-config.decorator.ts`:9
- `RouteConstraints` — `packages/platform-fastify/decorators/route-constraints.decorator.ts`:10
- `RouteSchema` — `packages/platform-fastify/decorators/route-schema.decorator.ts`:14
- `Roles` — `sample/10-fastify/src/common/decorators/roles.decorator.ts`:3
- `IS_PUBLIC_KEY` — `sample/19-auth-jwt/src/auth/decorators/public.decorator.ts`:3