UseGuards

August 17, 2026

UseGuards

Kind: Function

Source: `packages/common/decorators/core/use-guards.decorator.ts`

Part of: Common

Decorator that binds guards to the scope of the controller or method, depending on its context.

When `@UseGuards` is used at the controller level, the guard will be applied to every handler (method) in the controller.

When `@UseGuards` is used at the individual handler level, the guard will apply only to that specific method.

`@UseGuards()` binds one or more guards to a controller class or individual route handler. Guards run before the request reaches the handler and determine whether execution should continue, making them suitable for authentication, authorization, and request-level access control.

Signature

```
function UseGuards(guards: (CanActivate | Function)[]): MethodDecorator & ClassDecorator
```

Parameters

Name	Type
`guards`	`(CanActivate

Returns: `MethodDecorator & ClassDecorator`

Diagram

```
graph LR
  Request[Incoming Request] --> Guard[Guard canActivate()]
  Guard -->|true| Handler[Controller Handler]
  Guard -->|false / throws| Denied[Request Denied]

  Controller["@UseGuards() on Controller"] --> AllHandlers[All Controller Handlers]
  Method["@UseGuards() on Method"] --> Handler
```

Usage

```
import {
  Controller,
  Get,
  UseGuards,
  CanActivate,
  ExecutionContext,
} from '@nestjs/common';

class AuthGuard implements CanActivate {
  canActivate(context: ExecutionContext): boolean {
    const request = context.switchToHttp().getRequest();

    return Boolean(request.headers.authorization);
  }
}

@Controller('users')
@UseGuards(AuthGuard) // Applies to every handler in this controller
export class UsersController {
  @Get()
  findAll() {
    return ['user-1', 'user-2'];
  }

  @Get('public')
  // This route still inherits the controller-level guard.
  findPublicProfile() {
    return { name: 'Public User' };
  }
}

@Controller('reports')
export class ReportsController {
  @Get()
  @UseGuards(AuthGuard) // Applies only to this handler
  findReports() {
    return ['report-1'];
```

```
  }
}
```

AI Coding Instructions

- Apply `@UseGuards()` at the controller level for access rules shared by all routes; apply it at the handler level for route-specific rules.
- Pass guard classes, guard instances, or multiple guards as arguments, for example `@UseGuards(AuthGuard, RolesGuard)`.
- Ensure custom guards implement the `CanActivate` interface and return a boolean, Promise, or Observable result.
- Remember that controller-level guards are inherited by handler methods; do not assume a method-level decorator replaces controller guards.
- Use guards for authorization decisions, and keep validation or request transformation concerns in pipes and interceptors.

Used by

15 references from 15 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (15)

- `RecipesResolver` — `integration/graphql-code-first/src/recipes/recipes.resolver.ts`:13
- `CatsController` — `integration/inspector/src/cats/cats.controller.ts`:8
- `HelloController` — `integration/inspector/src/circular-hello/hello.controller.ts`:14
- `HelloController` — `integration/scopes/src/circular-hello/hello.controller.ts`:14
- `HelloController` — `integration/scopes/src/circular-transient/hello.controller.ts`:14
- `TestController` — `integration/scopes/src/circular-transient/test.controller.ts`:12
- `HelloController` — `integration/scopes/src/hello/hello.controller.ts`:14
- `HelloController` — `integration/scopes/src/msvc/hello.controller.ts`:8

…and 7 more.