UseInterceptors

August 17, 2026

UseInterceptors

Kind: Function

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

Part of: Common

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

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

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

`@UseInterceptors()` binds one or more interceptors to a controller class or an individual route handler. Controller-scoped interceptors run for every handler in that controller, while method-scoped interceptors affect only the decorated endpoint. Interceptors can transform requests or responses, add cross-cutting behavior, and wrap handler execution.

Signature

```
function UseInterceptors(interceptors: (NestInterceptor | Function)[]): MethodDecorator & Class
```

Parameters

Name	Type
`interceptors`	`(NestInterceptor

Returns: `MethodDecorator & ClassDecorator`

Diagram

```
graph LR
  A[Incoming Request] --> B{Interceptor Scope}
  B -->|Controller| C[All Controller Handlers]
  B -->|Method| D[Decorated Handler Only]
  C --> E[Interceptor Chain]
  D --> E
  E --> F[Route Handler]
  F --> G[Response]
```

Usage

```
import {
  Controller,
  Get,
  UseInterceptors,
  CallHandler,
  ExecutionContext,
  NestInterceptor,
} from '@nestjs/common';
import { Observable } from 'rxjs';
import { map } from 'rxjs/operators';

class ResponseWrapperInterceptor implements NestInterceptor {
  intercept(context: ExecutionContext, next: CallHandler): Observable<unknown> {
    return next.handle().pipe(
      map((data) => ({
        data,
      })),
    );
  }
}

@Controller('users')
@UseInterceptors(ResponseWrapperInterceptor)
export class UsersController {
  @Get()
  findAll() {
    return [{ id: 1, name: 'Ada' }];
  }

  @Get('health')
  health() {
    return { status: 'ok' };
  }
}
```

AI Coding Instructions

- Apply `@UseInterceptors()` to a controller for behavior shared by all of its handlers; apply it to a method for endpoint-specific behavior.
- Pass interceptor classes, interceptor instances, or multiple interceptors as supported by the framework's decorator metadata.
- Implement interceptors with the `NestInterceptor` contract and call `next.handle()` unless intentionally short-circuiting request execution.
- Preserve interceptor ordering when composing multiple interceptors, since each interceptor wraps the next handler in the chain.
- Use interceptors for cross-cutting concerns such as response mapping, caching, logging, serialization, and timeouts rather than placing that logic directly in controllers.

Used by

17 references from 17 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (17)

- `RecipesResolver` — `integration/graphql-code-first/src/recipes/recipes.resolver.ts`:13
- `HelloController` — `integration/inspector/src/circular-hello/hello.controller.ts`:14
- `RequestChainController` — `integration/inspector/src/request-chain/request-chain.controller.ts`:5
- `PassthroughInterceptor` — `integration/nest-application/sse/src/app.controller.ts`:28
- `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

…and 9 more.