NestInterceptor

August 18, 2026

NestInterceptor

Kind: Interface

Source: `packages/common/interfaces/features/nest-interceptor.interface.ts`

Part of: Common

Interface describing implementation of an interceptor.

`NestInterceptor` defines a hook for wrapping and transforming request handling in NestJS. Implementations receive the current `ExecutionContext` and a `CallHandler`, allowing them to run logic before or after the route handler, modify responses, handle errors, or measure execution time.

Diagram

```
graph LR
  A[Incoming Request] --> B[Interceptor.intercept]
  B --> C[ExecutionContext]
  B --> D[CallHandler]
  D --> E[Route Handler]
  E --> F[Observable Response]
  F --> G[Interceptor Response Transformation]
  G --> H[Client Response]
```

Usage

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

```ts
@Injectable()
export class ResponseEnvelopeInterceptor implements NestInterceptor {
  intercept(
    context: ExecutionContext,
    next: CallHandler,
  ): Observable<{ data: unknown }> {
    return next.handle().pipe(
      map((data) => ({ data })),
    );
  }
}

// Register globally or attach to a controller/route:
// @UseInterceptors(ResponseEnvelopeInterceptor)
```

AI Coding Instructions

- Implement `intercept(context, next)` and always return an `Observable` from `next.handle()` unless intentionally short-circuiting the request.
- Use RxJS operators such as `map`, `catchError`, `tap`, and `finalize` to transform responses, handle errors, or perform side effects.
- Use `ExecutionContext` to access transport-specific details, such as `context.switchToHttp().getRequest()` for HTTP requests.
- Register interceptors with `@UseInterceptors()`, as global interceptors, or through dependency injection when they require application services.
- Avoid subscribing to `next.handle()` inside the interceptor; return the observable so Nest can manage the response lifecycle.

Used by

36 references from 36 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (36)

- `DataInterceptor` — `integration/graphql-code-first/src/common/interceptors/data.interceptor.ts`:10
- `Interceptor` — `integration/inspector/src/circular-hello/interceptors/logging.interceptor.ts`:10
- `TimeoutInterceptor` — `integration/inspector/src/common/interceptors/timeout.interceptor.ts`:10

- `LoggingInterceptor` — `integration/inspector/src/core/interceptors/logging.interceptor.ts`:10
- `TransformInterceptor` — `integration/inspector/src/core/interceptors/transform.interceptor.ts`:14
- `LoggingInterceptor` — `integration/inspector/src/request-chain/interceptors/logging.interceptor.ts`:10
- `PassthroughInterceptor` — `integration/nest-application/sse/src/app.controller.ts`:28
- `Interceptor` — `integration/scopes/src/circular-hello/interceptors/logging.interceptor.ts`:10

…and 28 more.