



COMMON

OnModuleDestroy

Kind: Interface

VERSION

47

UPDATED

7 August 2026

DOCUMENT

reference-subsystem-packages-common/interface-onmoduledestroy

CONTENTS

1	Diagram	2	3	AI Coding Instructions	3
2	Usage	2	4	Used by	3

Kind: Interface

Source: `packages/common/interfaces/hooks/on-destroy.interface.ts`

Part of: [Common](#)

Interface defining method called just before Nest destroys the host module (`app.close()` method has been evaluated). Use to perform cleanup on resources (e.g., Database connections).

`OnModuleDestroy` is a Nest lifecycle interface for providers, controllers, and modules that need to release resources before their host module is destroyed. Nest invokes `onModuleDestroy()` after `app.close()` begins shutdown, making it suitable for closing database connections, stopping background workers, or clearing external clients.

1 Diagram

```
graph LR
  A[Application calls app.close()] --> B[Nest starts module shutdown]
  B --> C[Provider implements OnModuleDestroy]
  C --> D[onModuleDestroy() is invoked]
  D --> E[Release resources<br/>close connections / stop workers]
  E --> F[Host module is destroyed]
```

2 Usage

```
import { Injectable, OnModuleDestroy } from '@nestjs/common';

@Injectable()
export class DatabaseService implements OnModuleDestroy {
  private readonly connection = createDatabaseConnection();

  async onModuleDestroy(): Promise<void> {
    await this.connection.close();
  }
}

function createDatabaseConnection() {
  return {
    async close() {
      console.log('Database connection closed');
    },
  };
}
```

3 AI Coding Instructions

- Implement `OnModuleDestroy` on injectable providers that own resources requiring explicit cleanup.
- Define an `onModuleDestroy()` method; it may be synchronous or return a `Promise`.
- Close only resources owned by the current provider, such as database clients, message consumers, or timers.
- Make cleanup safe to run once and avoid throwing unnecessary errors during application shutdown.
- Ensure shutdown is initiated through `app.close()` so Nest can invoke lifecycle hooks.

4 Used by

4 references from 4 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (4)

- `KafkaController` — `integration/microservices/src/kafka/kafka.controller.ts`:15
- `KafkaConcurrentController` — `integration/microservices/src/kafka-concurrent/kafka-concurrent.controller.ts`:24
- `DatabaseConnection` — `integration/repl/src/database/database.connection.ts`:3
- `callModuleDestroyHook` — `packages/core/hooks/on-module-destroy.hook.ts`:41