# callModuleDestroyHook

August 18, 2026

## callModuleDestroyHook

**Kind:** Function

**Source:** `packages/core/hooks/on-module-destroy.hook.ts`

**Part of:** Core

Calls the `onModuleDestroy` function on the module and its children (providers / controllers).

`callModuleDestroyHook` runs the `onModuleDestroy()` lifecycle hook for a module's providers, controllers, injectables, middleware, and the module class itself. It is used during application shutdown to ensure managed resources—such as database connections, queues, and timers—are released in the correct lifecycle phase.

### Signature

```
async function callModuleDestroyHook(module: Module): Promise<any>
```

### Parameters

| Name | Type |
| --- | --- |
| `module` | `Module` |

**Returns:** `Promise<any>`

### Diagram

```
graph LR
  A[Application shutdown] --> B[callModuleDestroyHook]
  B --> C[Module providers]
  B --> D[Controllers]
  B --> E[Injectables and middleware]
```

```
  C --> F[onModuleDestroy hooks]
  D --> F
  E --> F
  B --> G[Module class]
  G --> H[Module onModuleDestroy hook]
```

## Usage

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

class DatabaseService implements OnModuleDestroy {
  async onModuleDestroy() {
    console.log('Closing database connection');
    // await this.database.close();
  }
}

@Module({
  providers: [DatabaseService],
})
export class DatabaseModule implements OnModuleDestroy {
  onModuleDestroy() {
    console.log('Destroying DatabaseModule');
  }
}

// Nest calls callModuleDestroyHook internally when the application closes:
// await app.close();
```

## AI Coding Instructions

- Treat this as an internal lifecycle utility; application code should normally trigger it through `app.close()` rather than calling it directly.
- Ensure resource-owning providers implement `onModuleDestroy()` and make cleanup operations safe to run once.
- Preserve the distinction between non-transient and transient provider instances when changing hook invocation logic.
- Keep module-class destruction after child provider, controller, injectable, and middleware hooks have completed.
- Await asynchronous cleanup hooks so shutdown does not complete before resources are released.

## Relationships

- IMPORTS → `OnModuleDestroy`
- IMPORTS → `isFunction`
- IMPORTS → `isNil`