InterceptorsContextCreator

August 18, 2026

InterceptorsContextCreator

Kind: Class

Source: `packages/core/interceptors/interceptors-context-creator.ts`

Part of: Core

`InterceptorsContextCreator` resolves the interceptors that apply to a Nest handler, including global, controller-level, and method-level metadata. It converts interceptor classes or instances into concrete `NestInterceptor` instances using the current module and request context.

Extends: `ContextCreator`

Methods

Method	Signature	Returns
`create`	`create(instance: Controller, callback: (...args: unknown[]) => unknown, module: string, contextId: undefined, inquirerId: string)`	`NestInterceptor[]`
`createConcreteContext`	`createConcreteContext(metadata: T, contextId: undefined, inquirerId: string)`	`R`
`getInterceptorInstance`	`getInterceptorInstance(metatype: Function	NestInterceptor, contextId: undefined, inquirerId: string)`
`getInstanceByMetatype`	`getInstanceByMetatype(metatype: Type<unknown>)`	`InstanceWrapper
`getGlobalMetadata`	`getGlobalMetadata(contextId: undefined, inquirerId: string)`	`T`

Where it refuses work

- `InterceptorsContextCreator` stops the work with an early return when `isEmpty(metadata)`.
- `InterceptorsContextCreator` stops the work with an early return when `isObject`.
- `InterceptorsContextCreator` stops the work with an early return when `!instanceWrapper`.
- `InterceptorsContextCreator` stops the work with an early return when `!this.moduleContext`.
- `InterceptorsContextCreator` stops the work with an early return when `!moduleRef`.
- `InterceptorsContextCreator` stops the work with an early return when `!this.config`.

Diagram

```
graph LR
  A[Global Interceptors] --> D[InterceptorsContextCreator.create]
  B[Controller @UseInterceptors] --> D
  C[Handler @UseInterceptors] --> D
  D --> E[createConcreteContext]
  E --> F[getInterceptorInstance]
  F --> G[Module Injectable Wrapper]
  G --> H[NestInterceptor[]]
```

Usage

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

@Injectable()
export class LoggingInterceptor implements NestInterceptor {
  intercept(context: ExecutionContext, next: CallHandler): Observable<unknown> {
    const handlerName = context.getHandler().name;

    console.log(`Before ${handlerName}`);

    return next.handle().pipe(
      tap(() => console.log(`After ${handlerName}`)),
```

```
    );
  }
}

@UseInterceptors(LoggingInterceptor)
export class UsersController {
  findAll() {
    return [];
  }
}
```

Nest internally uses `InterceptorsContextCreator` to collect `LoggingInterceptor` metadata, resolve its injectable instance, and execute it around the controller handler.

AI Coding Instructions

- Preserve interceptor resolution order: global interceptors should be applied before controller- and handler-scoped interceptors.
- Support both interceptor instances (`{ intercept() {} }`) and interceptor classes resolved through the module injector.
- Return only valid `NestInterceptor` instances; unresolved providers should be filtered out rather than causing downstream failures.
- Respect request-scoped providers by resolving instances with the active `ContextId` and optional inquirer identifier.
- When adding interceptor metadata, use Nest's `@UseInterceptors()` integration rather than directly constructing this internal context creator.

Relationships

- IMPORTS → `INTERCEPTORS_METADATA`
- IMPORTS → `Controller`
- IMPORTS → `NestInterceptor`
- IMPORTS → `Type`
- IMPORTS → `isEmpty`
- IMPORTS → `isFunction`

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)

- `RpcHandlerMetadata` — `packages/microservices/context/rpc-context-creator.ts`:35
- `MicroservicesModule` — `packages/microservices/microservices-module.ts`:23
- `WsHandlerMetadata` — `packages/websockets/context/ws-context-creator.ts`:34
- `SocketModule` — `packages/websockets/socket-module.ts`:27