ContextIdFactory

August 18, 2026

ContextIdFactory

Kind: Class

Source: `packages/core/helpers/context-id-factory.ts`

Part of: Core

`ContextIdFactory` creates and resolves context identifiers used to scope dependency injection instances to a specific request or custom execution context. It supports generating standalone IDs, reusing IDs attached to requests, and applying a custom context ID strategy for advanced transports or multi-tenant workflows.

Methods

Method	Signature	Returns
`create`	`create()`	`ContextId`
`getByRequest`	`getByRequest(request: T, propsToInspect: string[])`	`ContextId`
`apply`	`apply(strategy: ContextIdStrategy)`	`void`

Where it refuses work

- `ContextIdFactory` stops the work with an early return when `!request`.
- `ContextIdFactory` stops the work with an early return when `request[REQUEST_CONTEXT_ID as any]`.
- `ContextIdFactory` stops the work with an early return when `request[key]?.[REQUEST_CONTEXT_ID]`.
- `ContextIdFactory` stops the work with an early return when `!this.strategy`.

Diagram

```mermaid
graph LR
  A[Incoming request or custom context] --> B[ContextIdFactory.getByRequest]
  B --> C{Context ID already attached?}
  C -->|Yes| D[Reuse existing ContextId]
  C -->|No| E[ContextIdFactory.create]
  E --> F[New ContextId]
  F --> G[Request-scoped provider resolution]
  D --> G
  H[ContextIdFactory.apply] --> I[Custom ContextIdStrategy]
  I --> B
```

Usage

```ts
import { ContextIdFactory, ModuleRef } from '@nestjs/core';
import { UsersService } from './users.service';

async function resolveRequestScopedService(
  request: Request,
  moduleRef: ModuleRef,
) {
  // Reuses the request context ID when one exists, or creates a new one.
  const contextId = ContextIdFactory.getByRequest(request);

  const usersService = await moduleRef.resolve(UsersService, contextId);

  return usersService;
}

// Create an isolated context for background jobs or tests.
const jobContextId = ContextIdFactory.create();
```

AI Coding Instructions

- Use `getByRequest(request)` when resolving request-scoped providers so providers share the same lifecycle within one request.
- Use `create()` for non-HTTP execution paths such as queue jobs, scheduled tasks, tests, or manually managed scopes.
- Resolve scoped dependencies through `ModuleRef.resolve(provider, contextId)` with the same context ID throughout the operation.
- Apply a custom strategy with `ContextIdFactory.apply()` only when integrating custom transports or parent-child context behavior; ensure the strategy is initialized during application bootstrap.

- Avoid creating a new context ID repeatedly during a single request, as this produces separate instances of request-scoped providers.

Used by

1 reference from 1 file. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (1)

- `ListenersController` — `packages/microservices/listeners-controller.ts`:45