INestApplicationContext

August 18, 2026

INestApplicationContext

Kind: Interface

Source: `packages/common/interfaces/nest-application-context.interface.ts`

Part of: Common

Interface defining NestApplicationContext.

`INestApplicationContext` defines the contract for a Nest standalone application context without an HTTP server. It provides dependency-injection access, lifecycle management, logging configuration, request-scoped provider resolution, and shutdown hook support. It is commonly returned by `NestFactory.createApplicationContext()` for CLI tools, workers, scripts, and background services.

Diagram

```
graph LR
  A[AppModule] --> B[NestFactory.createApplicationContext]
  B --> C[INestApplicationContext]

  C --> D[get / resolve providers]
  C --> E[select module context]
  C --> F[init / close lifecycle]
  C --> G[useLogger / flushLogs]
  C --> H[enableShutdownHooks]
```

Usage

```
import { INestApplicationContext } from '@nestjs/common';
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';
import { ReportService } from './report.service';
```

```ts
async function bootstrap() {
  const app: INestApplicationContext =
    await NestFactory.createApplicationContext(AppModule);

  app.enableShutdownHooks();

  const reportService = app.get(ReportService);
  await reportService.generateDailyReport();

  await app.close();
}

void bootstrap();
```

AI Coding Instructions

- Use `NestFactory.createApplicationContext()` when building non-HTTP processes such as CLI commands, queues, cron workers, or migration scripts.
- Retrieve singleton providers with `app.get()`; use `app.resolve()` when resolving request-scoped or transient providers.
- Always call `await app.close()` when the process completes to release lifecycle resources and trigger shutdown hooks.
- Register `enableShutdownHooks()` for long-running processes that should gracefully handle termination signals.
- Do not use HTTP application methods such as `listen()` with `INestApplicationContext`; use `INestApplication` when an HTTP server is required.

Used by

5 references from 5 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (5)

- `NestApplicationContext` — `packages/core/nest-application-context.ts`:40
- `IEntryNestModule` — `packages/core/nest-factory.ts`:39
- `ModuleDebugEntry` — `packages/core/repl/repl-context.ts`:22
- `WsAdapter` — `packages/platform-ws/adapters/ws-adapter.ts`:39
- `BaseWsInstance` — `packages/websockets/adapters/ws-adapter.ts`:8