NestApplicationContext

August 18, 2026

NestApplicationContext

Kind: Class

Source: `packages/core/nest-application-context.ts`

Part of: Core

`NestApplicationContext` provides a non-HTTP NestJS application runtime for resolving providers, selecting modules, and managing dependency injection. It underpins application contexts created through `NestFactory.createApplicationContext()`, allowing scripts, CLI tools, workers, and tests to access Nest-managed services without starting a web server.

Extends: `AbstractInstanceResolver`

Implements: `INestApplicationContext`

Methods

Method	Signature	Returns
`selectContextModule`	`selectContextModule()`	`void`
`select`	`select(moduleType: Type`	DynamicModule, selectOptions: SelectOptions)`
`get`	`get(typeOrToken: Type`	Function
`get`	`get(typeOrToken: Type`	Function
`get`	`get(typeOrToken: Type`	Function
`get`	`get(typeOrToken: Type`	Abstract
`resolve`	`resolve(typeOrToken: Type`	Function
`resolve`	`resolve(typeOrToken: Type`	Function

Method	Signature	Returns
`resolve`	`resolve(typeOrToken: Type	Function
`resolve`	`resolve(typeOrToken: Type	Function
`resolve`	`resolve(typeOrToken: Type	Abstract
`registerRequestByContextId`	`registerRequestByContextId(request: T, contextId: ContextId)`	`void`
`init`	`init()`	`Promise<this>`
`close`	`close(signal: string)`	`Promise<void>`
`useLogger`	`useLogger(logger: LoggerService	LogLevel[]
`flushLogs`	`flushLogs()`	`void`
`flushLogsOnOverride`	`flushLogsOnOverride()`	`void`
`enableShutdownHooks`	`enableShutdownHooks(signals: (ShutdownSignal	string)[], options: ShutdownHooksOptions)`
`dispose`	`dispose()`	`Promise<void>`
`listenToShutdownSignals`	`listenToShutdownSignals(signals: string[], options: ShutdownHooksOptions)`	`void`
`unsubscribeFromProcessSignals`	`unsubscribeFromProcessSignals()`	`void`
`callInitHook`	`callInitHook()`	`Promise<void>`
`callDestroyHook`	`callDestroyHook()`	`Promise<void>`
`callBootstrapHook`	`callBootstrapHook()`	`Promise<void>`
`callShutdownHook`	`callShutdownHook(signal: string)`	`Promise<void>`
`callBeforeShutdownHook`	`callBeforeShutdownHook(signal: string)`	`Promise<void>`
`assertNotInPreviewMode`	`assertNotInPreviewMode(methodName: string)`	`void`

Properties

Property	Type
`isInitialized`	`any`
`injector`	`Injector`

Property	Type
`logger`	`any`

Where it refuses work

- `NestApplicationContext` stops the work with `UnknownModuleException` when `!selectedModule`.
- `NestApplicationContext` stops the work with an early return when `this.isInitialized`.
- `NestApplicationContext` stops the work with an early return when `receivedSignal`.
- `NestApplicationContext` stops the work with an early return when `!this.shutdownCleanupRef`.
- `NestApplicationContext` stops the work with an early return when `this._moduleRefsForHooksByDistance`.

When something fails

- `NestApplicationContext` handles failure in 1 place: it logs it and continues in all 1.

Diagram

```
graph LR
  A[NestFactory.createApplicationContext] --> B[NestApplicationContext]
  B --> C[Module Container]
  C --> D[Root Module]
  C --> E[Feature Modules]
  B --> F[get Provider]
  B --> G[resolve Scoped Provider]
  B --> H[select Module Context]
  H --> I[get Provider from Module]
```

Usage

```
import { Injectable, Module } from '@nestjs/common';
import { NestFactory } from '@nestjs/core';

@Injectable()
class ReportService {
  generate() {
    return 'Report generated';
  }
}
```

```
@Module({
  providers: [ReportService],
  exports: [ReportService],
})
class ReportsModule {}

@Module({
  imports: [ReportsModule],
})
class AppModule {}

async function bootstrap() {
  const app = await NestFactory.createApplicationContext(AppModule);

  // Resolve a singleton provider from the application context.
  const reports = app.get(ReportService);
  console.log(reports.generate());

  // Select a module before resolving providers from its module scope.
  const reportsContext = app.select(ReportsModule);
  const scopedReports = reportsContext.get(ReportService);

  await app.close();
}

bootstrap();
```

AI Coding Instructions

- Use `get()` for singleton or static-scope providers; use `resolve()` when working with request-scoped or transient providers.
- Call `select(ModuleClass)` before retrieving a provider when module-specific lookup or strict module boundaries are required.
- Prefer injection tokens or provider classes consistently; ensure queried providers are registered and exported when accessed across module boundaries.
- Always close application contexts created for scripts, tests, or workers with `await app.close()` to release lifecycle resources.
- Do not instantiate `NestApplicationContext` directly in application code; create it through `NestFactory.createApplicationContext()`.

How it works

`NestApplicationContext` is a public class that extends `AbstractInstanceResolver` and implements `INestApplicationContext`. It represents an application context backed by a

`NestContainer`, with an optionally selected module and a module-navigation scope. [packages/core/nest-application-context.ts:40-46] [packages/core/nest-application-context.ts:68-81]

The constructor stores the container and options, creates an `Injector`, obtains the container's `ModuleCompiler`, and logs a preview-mode warning when `appOptions.preview` is true. [packages/core/nest-application-context.ts:68-81] Its instance-link registry is created lazily from the container on the first lookup. [packages/core/nest-application-context.ts:61-66]

Relationships

- IMPORTS → `INestApplicationContext`
- IMPORTS → `Logger`
- IMPORTS → `LoggerService`
- IMPORTS → `LogLevel`
- IMPORTS → `ShutdownSignal`
- IMPORTS → `Abstract`
- IMPORTS → `DynamicModule`
- IMPORTS → `GetOrResolveOptions`
- IMPORTS → `SelectOptions`
- IMPORTS → `ShutdownHooksOptions`
- IMPORTS → `Type`
- IMPORTS → `NestApplicationContextOptions`
- IMPORTS → `isEmpty`

Used by

2 references from 2 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (2)

- `NestMicroservice` — `packages/microservices/nest-microservice.ts`:35
- `TestingModule` — `packages/testing/testing-module.ts`:26