



CORE

NestApplication

Kind: Class

VERSION
47

UPDATED
7 August 2026

DOCUMENT
reference-subsystem-packages-core/class-nestapplication

CONTENTS

1	Methods	2	5	Usage	5
2	Properties	4	6	AI Coding Instructions	6
3	Where it refuses work	4	7	Relationships	6
4	Diagram	5	8	Used by	7

Kind: Class

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

Part of: Core

`NestApplication` is the core runtime class behind a Nest HTTP application. It coordinates HTTP server creation, module registration, WebSocket integration, parser middleware, and application lifecycle initialization. Applications are typically created through `NestFactory`, which configures and returns a `NestApplication` instance.

Extends: `NestApplicationContext`

Implements: `INestApplication`

1 Methods

Method	Signature	Returns
<code>dispose</code>	<code>dispose()</code>	<code>Promise<void></code>
<code>getHttpAdapter</code>	<code>getHttpAdapter()</code>	<code>AbstractHttpAdapter</code>
<code>registerHttpServer</code>	<code>registerHttpServer()</code>	<code>void</code>
<code>getUnderlyingHttpServer</code>	<code>getUnderlyingHttpServer()</code>	<code>T</code>
<code>applyOptions</code>	<code>applyOptions()</code>	<code>void</code>
<code>createServer</code>	<code>createServer()</code>	<code>T</code>
<code>registerModules</code>	<code>registerModules()</code>	<code>void</code>
<code>registerWsModule</code>	<code>registerWsModule()</code>	<code>void</code>
<code>init</code>	<code>init()</code>	<code>Promise<this></code>
<code>registerParserMiddleware</code>	<code>registerParserMiddleware()</code>	<code>void</code>
<code>registerRouter</code>	<code>registerRouter()</code>	<code>void</code>

<code>registerRouterHooks</code>	<code>registerRouterHooks()</code>	<code>void</code>
<code>connectMicroservice</code>	<code>connectMicroservice(microserviceOptions: T, hybridAppOptions: NestHybridApplicationOptions)</code>	<code>INestMicroservice</code>
<code>getMicroservices</code>	<code>getMicroservices()</code>	<code>INestMicroservice[]</code>
<code>getHttpServer</code>	<code>getHttpServer()</code>	<code>void</code>
<code>startAllMicroservices</code>	<code>startAllMicroservices()</code>	<code>Promise<this></code>
<code>use</code>	<code>use(args: [any, any?])</code>	<code>this</code>
<code>useBodyParser</code>	<code>useBodyParser(args: [any, any?])</code>	<code>this</code>
<code>enableCors</code>	<code>enableCors(options: any)</code>	<code>void</code>
<code>enableVersioning</code>	<code>enableVersioning(options: VersioningOptions)</code>	<code>this</code>
<code>listen</code>	<code>`listen(port: number</code>	<code>string)`</code>
<code>listen</code>	<code>`listen(port: number</code>	<code>string, hostname: string)`</code>
<code>listen</code>	<code>`listen(port: number</code>	<code>string, args: any[])`</code>
<code>getUrl</code>	<code>getUrl()</code>	<code>Promise<string></code>
<code>setGlobalPrefix</code>	<code>setGlobalPrefix(prefix: string, options: GlobalPrefixOptions)</code>	<code>this</code>
<code>useWebSocketAdapter</code>	<code>useWebSocketAdapter(adapter: WebSocketAdapter)</code>	<code>this</code>
<code>useGlobalFilters</code>	<code>useGlobalFilters(filters: ExceptionFilter[])</code>	<code>this</code>
<code>useGlobalPipes</code>	<code>useGlobalPipes(pipes: PipeTransform<any>[])</code>	<code>this</code>

<code>useGlobalInterceptors</code>	<code>useGlobalInterceptors(interceptors: NestInterceptor[])</code>	<code>this</code>
<code>useGlobalGuards</code>	<code>useGlobalGuards(guards: CanActivate[])</code>	<code>this</code>
<code>useStaticAssets</code>	<code>useStaticAssets(options: any)</code>	<code>this</code>
<code>useStaticAssets</code>	<code>useStaticAssets(path: string, options: any)</code>	<code>this</code>
<code>useStaticAssets</code>	<code>useStaticAssets(pathOrOptions: any, options: any)</code>	<code>this</code>
<code>setBaseViewsDir</code>	<code>setBaseViewsDir(path: string)</code>	<code>string[]</code>
<code>setViewEngine</code>	<code>setViewEngine(engineOrOptions: any)</code>	<code>this</code>

2 Properties

Property	Type
<code>logger</code>	<code>any</code>

3 Where it refuses work

- `NestApplication` stops the work with an early return when `!this.appOptions || !this.appOptions.cors`.
- `NestApplication` stops the work with an early return when `!passCustomOptions`.
- `NestApplication` stops the work with an early return when `!this.socketModule`.
- `NestApplication` stops the work with an early return when `this.isInitialized`.
- `NestApplication` stops the work with an early return when `originalCallbackArgs[0] instanceof Error`.
- `NestApplication` stops the work with an early return when `platform() === 'win32'`.

4 Diagram

```
graph LR
  A[NestFactory.create()] --> B[NestApplication]
  B --> C[createServer()]
  B --> D[registerModules()]
  B --> E[registerWsModule()]
  B --> F[registerParserMiddleware()]
  B --> G[init()]
  G --> H[HTTP Adapter]
  H --> I[Underlying HTTP Server]
  B --> J[dispose()]
```

5 Usage

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';

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

  // Initialize registered modules, middleware, and adapters.
  await app.init();

  const httpAdapter = app.getHttpAdapter();
  const server = app.getUnderlyingHttpServer();

  console.log(`Using adapter: ${httpAdapter.getType()}`);
  console.log(`Server available: ${Boolean(server)}`);

  await app.listen(3000);

  process.on('SIGTERM', async () => {
    await app.close();
    await app.dispose();
  });
}

bootstrap();
```

6 AI Coding Instructions

- Create application instances through `NestFactory.create()` rather than constructing `NestApplication` directly.
- Preserve the initialization order: register modules and infrastructure before calling `init()`.
- Use `getHttpAdapter()` for adapter-specific integrations; avoid assuming Express or Fastify APIs are always available.
- Use `getUnderlyingHttpServer()` only when direct access to the native HTTP server is required.
- Ensure shutdown paths clean up resources through the application lifecycle, including disposal when appropriate.

7 Relationships

- IMPORTS → `CanActivate`
- IMPORTS → `ExceptionHandler`
- IMPORTS → `HttpServer`
- IMPORTS → `INestApplication`
- IMPORTS → `INestMicroservice`
- IMPORTS → `NestHybridApplicationOptions`
- IMPORTS → `NestInterceptor`
- IMPORTS → `PipeTransform`
- IMPORTS → `VersioningOptions`
- IMPORTS → `VersioningType`
- IMPORTS → `WebSocketAdapter`
- IMPORTS → `GlobalPrefixOptions`
- IMPORTS → `NestApplicationOptions`
- IMPORTS → `Logger`
- IMPORTS → `loadPackage`
- IMPORTS → `addLeadingSlash`
- IMPORTS → `isFunction`
- IMPORTS → `isObject`
- IMPORTS → `isString`
- IMPORTS → `SocketModule`
- IMPORTS → `MicroservicesModule`
- IMPORTS → `-nestjs-microservices`

8 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)

- `TestingModule` — `packages/testing/testing-module.ts`:26
- `BaseWsInstance` — `packages/websockets/adapters/ws-adapter.ts`:8