# TestingModule

August 18, 2026

## TestingModule

**Kind:** Class

**Source:** `packages/testing/testing-module.ts`

**Part of:** Testing

`TestingModule` is the compiled test container returned by Nest's testing utilities. It provides access to resolved providers and exposes factory methods for creating test-specific Nest HTTP applications or microservices from the module configuration.

**Extends:** `NestApplicationContext`

### Methods

| Method | Signature | Returns |
| --- | --- | --- |
| `createNestApplication` | `createNestApplication(httpAdapter: HttpServer | AbstractHttpAdapter, options: NestApplicationOptions)` |
| `createNestApplication` | `createNestApplication(options: NestApplicationOptions)` | `T` |
| `createNestApplication` | `createNestApplication(serverOrOptions: | HttpServer |
| `createNestMicroservice` | `createNestMicroservice(options: NestMicroserviceOptions & T)` | `INestMicroservice` |

### Properties

| Property | Type |
| --- | --- |
| `graphInspector` | `GraphInspector` |

### Where it refuses work

- `TestingModule` stops the work with an early return when `!options || isUndefined(options.logger)`.
- `TestingModule` stops the work with an early return when `!(prop in receiver) && prop in adapter`.

## Diagram

```
graph LR
  A[Test.createTestingModule] --> B[TestingModuleBuilder]
  B --> C[TestingModule]
  C --> D[Resolve providers with get/resolve]
  C --> E[createNestApplication]
  C --> F[createNestMicroservice]
  E --> G[INestApplication]
  F --> H[INestMicroservice]
```

## Usage

```
import { Test } from '@nestjs/testing';
import { INestApplication } from '@nestjs/common';
import { AppModule } from '../src/app.module';

describe('AppModule', () => {
  let app: INestApplication;

  beforeAll(async () => {
    const moduleRef = await Test.createTestingModule({
      imports: [AppModule],
    }).compile();

    app = moduleRef.createNestApplication();
    await app.init();
  });

  afterAll(async () => {
    await app.close();
  });

  it('starts the application', async () => {
    expect(app).toBeDefined();
  });
});
```

## AI Coding Instructions

- Build and compile a testing module with `Test.createTestingModule(...).compile()` before calling `createNestApplication()` or `createNestMicroservice()`.
- Use `moduleRef.get()` or `moduleRef.resolve()` to retrieve and test providers directly when an HTTP server is unnecessary.
- Always call `await app.init()` before sending requests to an application created by `createNestApplication()`.
- Close created applications or microservices in teardown hooks to prevent open handles between test suites.
- Override providers, guards, or modules on the testing module builder before calling `compile()` to isolate external dependencies.

## How it works

`TestingModule` is the testing package's exported application-context class. It extends `NestApplicationContext`, retaining context operations such as dependency lookup, scoped resolution, lifecycle initialization, and shutdown; it is returned by `TestingModuleBuilder.compile()` after dependency scanning, instance creation, and application-provider setup. packages/testing/testing-module.ts:26 packages/testing/testing-module.builder.ts:97-131 packages/core/nest-application-context.ts:165-175 packages/core/nest-application-context.ts:226-245 packages/core/nest-application-context.ts:253-284

Its constructor receives a Nest container, graph inspector, context module, application configuration, and optional module scope. It initializes the inherited context with empty context options, stores the supplied graph inspector, and passes the supplied context module and scope to the parent context. packages/testing/testing-module.ts:29-40

## Relationships

- IMPORTS → `HttpServer`
- IMPORTS → `INestApplication`
- IMPORTS → `INestMicroservice`
- IMPORTS → `Logger`
- IMPORTS → `NestApplicationOptions`
- IMPORTS → `Type`
- IMPORTS → `NestMicroserviceOptions`
- IMPORTS → `NestApplicationContextOptions`

- IMPORTS → `loadPackage`
- IMPORTS → `isUndefined`
- IMPORTS → `AbstractHttpAdapter`
- IMPORTS → `NestApplication`
- IMPORTS → `NestApplicationContext`
- IMPORTS → `ApplicationConfig`
- IMPORTS → `NestContainer`
- IMPORTS → `Module`
- IMPORTS → `GraphInspector`
- IMPORTS → `-nestjs-microservices`
- IMPORTS → `-nestjs-platform-express`