MicroservicesModule

August 18, 2026

MicroservicesModule

Kind: Class

Source: `packages/microservices/microservices-module.ts`

Part of: Microservices

`MicroservicesModule` coordinates microservice transport setup within the application. It registers configured listeners and clients, binds them during startup, and closes active connections during shutdown.

Methods

Method	Signature	Returns
`register`	`register(container: NestContainer, graphInspector: GraphInspector, config: ApplicationConfig, options: TAppOptions)`	`void`
`setupListeners`	`setupListeners(container: NestContainer, serverInstance: Server)`	`void`
`setupClients`	`setupClients(container: NestContainer)`	`void`
`bindListeners`	`bindListeners(controllers: Map<string`	symbol
`bindClients`	`bindClients(items: Map<string`	symbol
`close`	`close()`	`void`

Where it refuses work

- `MicroservicesModule` stops the work with `RuntimeException` when `!this.listenersController`, in 2 places.
- `MicroservicesModule` stops the work with an early return when `this.appOptions?.preview`.

Diagram

```
graph LR
  A[Application Bootstrap] --> B[MicroservicesModule]
  B --> C[register()]
  C --> D[setupListeners()]
  C --> E[setupClients()]
  D --> F[bindListeners()]
  E --> G[bindClients()]
  F --> H[Incoming Microservice Messages]
  G --> I[Outbound Client Requests]
  B --> J[close()]
  J --> K[Close Listeners and Clients]
```

Usage

```
import { MicroservicesModule } from '@your-scope/microservices';

// Create the module with the application's configured microservice
// listeners and client connections.
const microservices = new MicroservicesModule(/* application dependencies */);

// During application startup, register and bind transports.
await microservices.register();

// The application can now receive messages through listeners
// and send requests through configured clients.

// During graceful shutdown, release transport resources.
await microservices.close();
```

AI Coding Instructions

- Keep listener and client initialization separated: use `setupListeners()` and `setupClients()` for construction, then `bindListeners()` and `bindClients()` for activation.
- Ensure `register()` is called during application bootstrap before handling microservice traffic.
- Add new transport integrations through the module’s setup and binding flow rather than creating untracked connections elsewhere.
- Always invoke `close()` during graceful shutdown to release listeners, client connections, and related resources.

- Preserve startup ordering so listeners and clients are fully configured before they are bound.

Relationships

- IMPORTS → `Controller`
- IMPORTS → `NestApplicationContextOptions`
- IMPORTS → `ApplicationConfig`
- IMPORTS → `RuntimeException`
- IMPORTS → `GuardsConsumer`
- IMPORTS → `GuardsContextCreator`
- IMPORTS → `NestContainer`
- IMPORTS → `Injector`
- IMPORTS → `InstanceWrapper`
- IMPORTS → `GraphInspector`
- IMPORTS → `InterceptorsConsumer`
- IMPORTS → `InterceptorsContextCreator`
- IMPORTS → `PipesConsumer`
- IMPORTS → `PipesContextCreator`

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)

- `NestApplication` — `packages/core/nest-application.ts`:54