ModulesContainer

August 18, 2026

ModulesContainer

Kind: Class

Source: `packages/core/injector/modules-container.ts`

Part of: Core

`ModulesContainer` is the core registry for application modules, storing `Module` instances and providing lookup by module ID. It also publishes RPC-capable instance wrappers through an observable registry so transport-related infrastructure can discover newly registered targets.

Extends: `Map`

Methods

Method	Signature	Returns
`getById`	`getById(id: string)`	`Module
`getRpcTargetRegistry`	`getRpcTargetRegistry()`	`Observable<T>`
`addRpcTarget`	`addRpcTarget(target: T)`	`void`

Diagram

```
graph LR
  A[Module registration] --> B[ModulesContainer]
  B --> C[Map of module tokens to Module instances]
  C --> D[getById(moduleId)]
  B --> E[RPC target ReplaySubject]
  F[addRpcTarget(wrapper)] --> E
  E --> G[getRpcTargetRegistry()]
  G --> H[RPC/transport consumers]
```

Usage

```
import { ModulesContainer } from '@nestjs/core/injector/modules-container';
import { InstanceWrapper } from '@nestjs/core/injector/instance-wrapper';

class PaymentsRpcHandler {
  processPayment() {
    return { status: 'accepted' };
  }
}

const modules = new ModulesContainer();

// Subscribe before or after targets are registered.
// The underlying replay-based registry can expose previously added targets.
const subscription = modules.getRpcTargetRegistry().subscribe((target) => {
  console.log(`Discovered RPC target: ${String(target.token)}`);
});

const paymentsTarget = new InstanceWrapper({
  token: PaymentsRpcHandler,
  name: PaymentsRpcHandler.name,
  metatype: PaymentsRpcHandler,
});

modules.addRpcTarget(paymentsTarget);

// Module IDs are distinct from the container's map keys.
const module = modules.getById('module-id');
console.log(module?.metatype?.name);

subscription.unsubscribe();
```

AI Coding Instructions

- Treat `ModulesContainer` as framework infrastructure; prefer interacting with it through Nest's injector and application lifecycle rather than manually constructing module entries in application code.
- Use `getById()` when matching Nest module IDs; do not assume the `Map` key is the same as `Module.id`.
- Register only valid `InstanceWrapper` objects through `addRpcTarget()` so RPC consumers receive the metadata they expect.
- Subscribe to `getRpcTargetRegistry()` when integrating transport or RPC discovery logic, and dispose subscriptions when the consumer is destroyed.

- Preserve the observable-based registration flow instead of replacing it with direct polling of module collections.