MiddlewareContainer

August 19, 2026

MiddlewareContainer

Kind: Class

Source: `packages/core/middleware/container.ts`

Part of: Core

`MiddlewareContainer` stores middleware providers and their route-level configurations for the application. It maintains collections keyed by injection token and module key, allowing the middleware runtime to resolve middleware instances and apply configurations during request pipeline setup.

Methods

Method	Signature	Returns
`getMiddlewareCollection`	`getMiddlewareCollection(moduleKey: string)`	`Map<InjectionToken, InstanceWrapper>`
`getConfigurations`	`getConfigurations()`	`Map<string, Set<MiddlewareConfiguration>>`
`insertConfig`	`insertConfig(configList: MiddlewareConfiguration[], moduleKey: string)`	`void`

Diagram

```
graph LR
  Module[Module registration] --> Config[MiddlewareConfiguration]
  Config --> Insert[MiddlewareContainer.insertConfig]
  Insert --> ConfigMap["configurations: Map<moduleKey, Set<MiddlewareConfiguration>>"]
  Providers[Middleware providers] --> MiddlewareMap["middleware: Map<InjectionToken, InstanceWr
  MiddlewareMap --> Resolver[Middleware resolver]
  ConfigMap --> Resolver
  Resolver --> Pipeline[HTTP middleware pipeline]
```

Usage

```
import { MiddlewareContainer } from '@nestjs/core/middleware/container';
import { RequestMethod } from '@nestjs/common';
import type { MiddlewareConfiguration } from '@nestjs/common/interfaces/middleware';

const middlewareContainer = new MiddlewareContainer(applicationConfig);

const configuration: MiddlewareConfiguration = {
  middleware: [LoggerMiddleware],
  forRoutes: [
    {
      path: 'users',
      method: RequestMethod.ALL,
    },
  ],
};

const moduleKey = 'UsersModule';

middlewareContainer.insertConfig([configuration], moduleKey);

const configurations = middlewareContainer.getConfigurations();
const userMiddlewareConfigs = configurations.get(moduleKey);

console.log(userMiddlewareConfigs);
```

AI Coding Instructions

- Use `insertConfig()` to add middleware configurations so multiple configuration entries for the same module are preserved in a `Set`.
- Treat module keys as stable, unique identifiers; inconsistent keys will create separate configuration groups.
- Do not replace the maps returned by `getMiddlewareCollection()` or `getConfigurations()`; consumers depend on the container's shared collections.
- Keep middleware provider registration and route configuration synchronized so each configured middleware can be resolved from its injection token.
- When adding configuration behavior, preserve insertion order because middleware execution order can affect request handling.