MiddlewareModule

August 18, 2026

MiddlewareModule

Kind: Class

Source: `packages/core/middleware/middleware-module.ts`

Part of: Core

`MiddlewareModule` coordinates middleware discovery, configuration loading, and route-level registration for the application. It resolves middleware implementations, normalizes their configuration, and attaches them to the appropriate routes during application initialization.

Methods

Method	Signature	Returns
`register`	`register(middlewareContainer: MiddlewareContainer, container: NestContainer, config: ApplicationConfig, injector: Injector, httpAdapter: HttpServer, graphInspector: GraphInspector, options: TAppOptions)`	`void`
`resolveMiddleware`	`resolveMiddleware(middlewareContainer: MiddlewareContainer, modules: Map<string, Module>)`	`void`
`loadConfiguration`	`loadConfiguration(middlewareContainer: MiddlewareContainer, moduleRef: Module, moduleKey: string)`	`void`
`registerMiddleware`	`registerMiddleware(middlewareContainer: MiddlewareContainer, applicationRef: any)`	`void`
`registerMiddlewareConfig`	`registerMiddlewareConfig(middlewareContainer: MiddlewareContainer, config: MiddlewareConfiguration, moduleKey: string, applicationRef: any)`	`void`

Method	Signature	Returns
`registerRouteMiddleware`	`registerRouteMiddleware(middlewareContainer: MiddlewareContainer, routeInfo: RouteInfo, config: MiddlewareConfiguration, moduleKey: string, applicationRef: any)`	`void`

Where it refuses work

- `MiddlewareModule` stops the work with `RuntimeException` when `isUndefined(instanceWrapper)`.
- `MiddlewareModule` stops the work with `InvalidMiddlewareException` when `isUndefined(instance?.use)`.
- `MiddlewareModule` stops the work with an early return when `!instance.configure`.
- `MiddlewareModule` stops the work with an early return when `!this.appOptions.preview`.
- `MiddlewareModule` stops the work with an early return when `!(middlewareBuilder instanceof MiddlewareBuilder)`.
- `MiddlewareModule` stops the work with an early return when `isModuleAGlobal && isModuleBGlobal`.

When something fails

- `MiddlewareModule` handles failure in 2 places: it logs it and continues in 1, and lets it reach the caller in 1.

Diagram

```
graph LR
  A[Application Bootstrap] --> B[MiddlewareModule.register]
  B --> C[loadConfiguration]
  C --> D[registerMiddlewareConfig]
  D --> E[resolveMiddleware]
  E --> F[registerMiddleware]
  F --> G[registerRouteMiddleware]
  G --> H[Application Routes]
```

Usage

```
import { MiddlewareModule } from '@your-package/core';

// Resolve the module from the application's dependency container.
```

```
const middlewareModule = app.get(MiddlewareModule);

// During bootstrap, load middleware configuration and bind middleware to routes.
await middlewareModule.register();
```

AI Coding Instructions

- Call `register()` during application bootstrap; it orchestrates configuration loading and middleware-to-route binding.
- Add new middleware through the supported configuration flow rather than registering route handlers directly.
- Ensure configured middleware can be resolved by the application container before it is referenced in route configuration.
- Keep route middleware configuration explicit, including the target route, HTTP method, and middleware ordering where applicable.
- When changing registration behavior, preserve the sequence of configuration loading, middleware resolution, and route registration.