MiddlewareBuilder

August 18, 2026

MiddlewareBuilder

Kind: Class

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

Part of: Core

`MiddlewareBuilder` collects middleware registrations and converts them into normalized `MiddlewareConfiguration` entries for the framework's middleware pipeline. It exposes a fluent `apply()` API through `MiddlewareConfigProxy`, while `build()` returns the accumulated configurations and `getHttpAdapter()` provides access to the underlying HTTP server adapter.

Implements: `MiddlewareConsumer`

Methods

Method	Signature	Returns
`apply`	`apply(middleware: Array<Type`	Function
`build`	`build()`	`MiddlewareConfiguration[]`
`getHttpAdapter`	`getHttpAdapter()`	`HttpServer`

Where it refuses work

- `MiddlewareBuilder` stops the work with an early return when `route.method !== item.method`.

Diagram

```
graph LR
  A[Application Module] --> B[MiddlewareBuilder]
  B -->|apply(...middleware)| C[MiddlewareConfigProxy]
```

```
  C -->|forRoutes / exclude| D[MiddlewareConfiguration]
  B -->|build()| E[MiddlewareConfiguration[]]
  B -->|getHttpAdapter()| F[HttpServer]
  E --> G[Middleware Pipeline]
  F --> G
```

Usage

```
import { MiddlewareBuilder } from '@nestjs/core/middleware/builder';
import { RoutesMapper } from '@nestjs/core/middleware/routes-mapper';

// Typically created and used internally by the framework.
const builder = new MiddlewareBuilder(
  new RoutesMapper(container, applicationConfig),
  httpAdapter,
);

builder
  .apply(LoggerMiddleware, AuthenticationMiddleware)
  .exclude('health')
  .forRoutes('users');

const middlewareConfigurations = builder.build();

// The HTTP adapter used to register middleware with the server.
const adapter = builder.getHttpAdapter();
```

AI Coding Instructions

- Use `apply()` as the entry point for registering one or more middleware classes/functions; route selection is configured through the returned `MiddlewareConfigProxy`.
- Call `build()` only after all middleware registrations are complete, since it returns the builder's accumulated configuration list.
- Preserve middleware ordering: configurations are processed in registration order, so avoid rearranging stored entries without considering execution order.
- Use `getHttpAdapter()` when integration code needs server-specific middleware behavior; avoid coupling configuration-building logic directly to a specific HTTP platform.
- Keep route mapping and exclusion logic delegated to `MiddlewareConfigProxy` and route-mapping utilities rather than duplicating route normalization in the builder.

Relationships

- IMPORTS → `HttpServer`
- IMPORTS → `MiddlewareConsumer`
- IMPORTS → `Type`
- IMPORTS → `MiddlewareConfigProxy`
- IMPORTS → `MiddlewareConfiguration`
- IMPORTS → `RouteInfo`
- IMPORTS → `stripEndSlash`