ExcludeRouteMetadata

August 18, 2026

ExcludeRouteMetadata

Kind: Interface

Source: `packages/core/router/interfaces/exclude-route-metadata.interface.ts`

Part of: Core

`ExcludeRouteMetadata` describes a route that should be excluded from router-level behavior, such as middleware or guard application. It combines the original route `path`, a compiled `pathRegex` used for matching requests, and the associated NestJS `requestMethod`.

Properties

Property	Type
`path`	`string`
`pathRegex`	`RegExp`
`requestMethod`	`RequestMethod`

Diagram

```
graph LR
  A[Incoming Request] --> B{Request Method Matches?}
  B -->|Yes| C{Path Matches pathRegex?}
  B -->|No| D[Do Not Exclude]
  C -->|Yes| E[Exclude Route Behavior]
  C -->|No| D

  F[ExcludeRouteMetadata] --> G[path: string]
  F --> H[pathRegex: RegExp]
  F --> I[requestMethod: RequestMethod]
```

Usage

```ts
import { RequestMethod } from '@nestjs/common';
import { ExcludeRouteMetadata } from './interfaces/exclude-route-metadata.interface';

const excludedHealthCheck: ExcludeRouteMetadata = {
  path: '/health',
  pathRegex: /^\/health\/?$/,
  requestMethod: RequestMethod.GET,
};

function shouldExclude(
  route: ExcludeRouteMetadata,
  path: string,
  method: RequestMethod,
): boolean {
  return (
    route.requestMethod === method &&
    route.pathRegex.test(path)
  );
}

shouldExclude(excludedHealthCheck, '/health', RequestMethod.GET); // true
```

AI Coding Instructions

- Keep `path` as the human-readable route definition and use `pathRegex` for runtime path matching.
- Ensure `pathRegex` matches both expected route parameters and optional trailing slashes when required.
- Compare `requestMethod` with NestJS `RequestMethod` enum values rather than raw HTTP method strings.
- Reset or avoid stateful regular expressions using the `g` or `y` flags before calling `.test()` repeatedly.
- Use this metadata when integrating route exclusion logic into middleware, guards, interceptors, or router configuration.

How it works

`ExcludeRouteMetadata` is a TypeScript interface for a route that is excluded from matching behavior such as global-prefix application or middleware execution. It has no executable members itself. [packages/core/router/interfaces/exclude-route-metadata.interface.ts:3-18]

Its required fields are:

- `path: string` — the route path. [packages/core/router/interfaces/exclude-route-metadata.interface.ts:4-7]
- `pathRegex: RegExp` — a regular expression for that path. [packages/core/router/interfaces/exclude-route-metadata.interface.ts:9-12]
- `requestMethod: RequestMethod` — the HTTP method associated with the exclusion. [packages/core/router/interfaces/exclude-route-metadata.interface.ts:14-17]

Internal mapping converts each configured exclusion from either a path string or `RouteInfo` into this shape. It converts legacy path syntax, builds `pathRegex` from the slash-prefixed converted path, and assigns `RequestMethod.ALL` for string entries or the source route's method for `RouteInfo` entries. [packages/core/middleware/utils.ts:15-34] If regular-expression construction throws a `TypeError`, that mapper calls `LegacyRouteConverter.printError(originalPath)` and rethrows the same error. [packages/core/middleware/utils.ts:35-40]

Exclusion matching first accepts a route when its metadata method is `RequestMethod.ALL` (or numeric `-1`) or equals the requested method; it then runs `pathRegex` against the slash-prefixed candidate path. [packages/core/router/utils/exclude-route.util.ts:5-22]

For global prefixes, `NestApplication.setGlobalPrefix()` maps configured `options.exclude` entries into `ExcludeRouteMetadata` records and stores them in application configuration. [packages/core/nest-application.ts:379-390] During route-path creation, a matching exclusion leaves the route path without the global prefix; nonmatching paths receive the prefix. [packages/core/router/route-path-factory.ts:59-77] [packages/core/router/route-path-factory.ts:117-143]

For middleware exclusions, the request URL is read from the HTTP adapter, stripped of its query string, and matched with the adapter-reported request method. When matched, generated middleware wrappers call `next()` rather than invoking the middleware. [packages/core/middleware/utils.ts:67-78] [packages/core/middleware/utils.ts:84-95][packages/core/middleware/utils.ts:118-135]

Relationships

- IMPORTS → `RequestMethod`