# RoutesMapper

August 18, 2026

## RoutesMapper

**Kind:** Class

**Source:** `packages/core/middleware/routes-mapper.ts`

**Part of:** Core

`RoutesMapper` resolves route declarations into normalized `RouteInfo` objects that middleware can consume consistently. It bridges controller metadata, literal path definitions, and application routing configuration so middleware registration can target the correct HTTP paths and methods.

### Methods

| Method | Signature | Returns |
| --- | --- | --- |
| `mapRouteToRouteInfo` | `mapRouteToRouteInfo(controllerOrRoute: Type | RouteInfo |

### Where it refuses work

- `RoutesMapper` stops the work with an early return when `!metatype`, in 2 places.
- `RoutesMapper` stops the work with an early return when `isString(controllerOrRoute)`.
- `RoutesMapper` stops the work with an early return when `this.isRouteInfo(routePathOrPaths, controllerOrRoute)`.
- `RoutesMapper` stops the work with an early return when `typeof version !== 'string' && Array.isArray(version)`.
- `RoutesMapper` stops the work with an early return when `!moduleRefsSet`.
- `RoutesMapper` stops the work with an early return when `versioningConfig`.

### Diagram

```mermaid
graph LR
  A[Route declaration] --> B[RoutesMapper]
  B --> C[Controller metadata]
  B --> D[Application route configuration]
  C --> E[Normalized RouteInfo[]]
  D --> E
  E --> F[Middleware registration]
```

## Usage

```ts
import { RoutesMapper } from '@nestjs/core/middleware/routes-mapper';
import { UsersController } from './users.controller';

// RoutesMapper is typically created and used internally by Nest's
// middleware subsystem with container and application configuration.
const mapper = new RoutesMapper(container, appConfig, routePathFactory);

// Resolve a controller's route metadata into middleware-ready route info.
const routes = mapper.mapRouteToRouteInfo(UsersController);

for (const route of routes) {
  console.log(route.path, route.method);
}
```

## AI Coding Instructions

- Keep route normalization centralized in `RoutesMapper`; middleware code should consume `RouteInfo` rather than reimplementing controller metadata lookup.
- Preserve support for controller classes, path strings, and explicit route objects when changing route-resolution behavior.
- Return an empty `RouteInfo[]` for route declarations that cannot be resolved instead of throwing during middleware configuration.
- Ensure changes remain compatible with application global prefixes, versioning, and controller path metadata.
- Treat this class as middleware infrastructure; instantiate it through framework wiring where possible rather than constructing it ad hoc in application code.