RouteDefinition

August 19, 2026

RouteDefinition

Kind: Interface

Source: `packages/core/router/paths-explorer.ts`

Part of: Core

`RouteDefinition` describes a discovered HTTP route within the router exploration process. It combines the route path, HTTP method, controller callback, method name, and optional API version so the framework can register and dispatch requests consistently.

Properties

Property	Type
`path`	`string[]`
`requestMethod`	`RequestMethod`
`targetCallback`	`RouterProxyCallback`
`methodName`	`string`
`version`	`VersionValue`

Diagram

```
graph LR
  Controller[Controller Method] --> Definition[RouteDefinition]
  Definition --> Path[path: string[]]
  Definition --> Method[requestMethod: RequestMethod]
  Definition --> Callback[targetCallback: RouterProxyCallback]
  Definition --> Name[methodName: string]
  Definition --> Version[version: VersionValue]
  Definition --> Router[Router Registration]
```

Usage

```ts
import { RequestMethod } from '@nestjs/common';
import type { RouteDefinition } from './paths-explorer';

const usersController = {
  findAll() {
    return [{ id: 1, name: 'Ada' }];
  },
};

const route: RouteDefinition = {
  path: ['users'],
  requestMethod: RequestMethod.GET,
  targetCallback: usersController.findAll.bind(usersController),
  methodName: 'findAll',
  version: '1',
};

// Used by the router explorer to register GET /v1/users.
```

AI Coding Instructions

- Preserve `path` as a string array; route exploration may produce multiple normalized paths for one handler.
- Use `RequestMethod` enum values instead of raw HTTP method strings.
- Bind `targetCallback` to its controller instance when constructing definitions manually, so `this` context is retained.
- Keep `methodName` aligned with the original controller method for metadata lookup, logging, and diagnostics.
- Pass the correct `version` value when integrating with versioned routing; avoid treating versioned and unversioned routes as interchangeable.

How it works

`RouteDefinition` is an exported TypeScript interface describing a discovered controller-method route. It contains:

- `path`: one or more route-path strings. `packages/core/router/paths-explorer.ts:17-23`
- `requestMethod`: the `RequestMethod` metadata value for the controller method. `packages/core/router/paths-explorer.ts:19`
- `targetCallback`: the instance method callback, typed to accept `(req, res, next)` and return `void` or `Promise<void>`. `packages/core/router/paths-explorer.ts:20`

`packages/core/router/router-proxy.ts:4-8`

- `methodName`: the discovered method's name. `packages/core/router/paths-explorer.ts:21`
- Optional `version`: method-level version metadata. `packages/core/router/paths-explorer.ts:22`

`PathsExplorer` creates a `RouteDefinition` while scanning method names on a controller prototype. A definition is created only when `PATH_METADATA` exists on the prototype method; methods without that metadata return `null` and are excluded from `scanForPaths` results. `packages/core/router/paths-explorer.ts:36-50` `packages/core/router/paths-explorer.ts:58-63`

During construction, a string path is converted to a one-item array and given a leading slash. For a non-string path value, the code maps its entries and adds a leading slash to each. The request method and optional version come from reflection metadata on the prototype callback, while `targetCallback` comes from the controller instance. `packages/core/router/paths-explorer.ts:58-81`

In HTTP route registration, the route's `version` is assigned to `routePathMetadata.methodVersion`; its request method selects the adapter router method; and every route path is expanded through `RoutePathFactory` before the handler is registered with the router. `packages/core/router/router-explorer.ts:140-151` `packages/core/router/router-explorer.ts:163-173` `packages/core/router/router-explorer.ts:198-247`

Relationships

- IMPORTS → `METHOD_METADATA`
- IMPORTS → `PATH_METADATA`
- IMPORTS → `VERSION_METADATA`
- IMPORTS → `RequestMethod`
- IMPORTS → `Controller`
- IMPORTS → `VersionValue`
- IMPORTS → `addLeadingSlash`
- IMPORTS → `isString`
- IMPORTS → `isUndefined`