RouterExplorer

August 17, 2026

RouterExplorer

Kind: Class

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

Part of: Core

`RouterExplorer` discovers controller route metadata and binds route handlers to the configured HTTP adapter. It creates proxied handlers with exception handling and request-scoped dependency resolution, making it a core part of the framework's controller-to-router integration.

Methods

Method	Signature	Returns
`explore`	`explore(instanceWrapper: InstanceWrapper, moduleKey: string, httpAdapterRef: T, host: string	RegExp
`extractRouterPath`	`extractRouterPath(metatype: Type<Controller>)`	`string[]`
`applyPathsToRouterProxy`	`applyPathsToRouterProxy(router: T, routeDefinitions: RouteDefinition[], instanceWrapper: InstanceWrapper, moduleKey: string, routePathMetadata: RoutePathMetadata, host: string	RegExp
`createRequestScopedHandler`	`createRequestScopedHandler(instanceWrapper: InstanceWrapper, requestMethod: RequestMethod, moduleRef: Module, moduleKey: string, methodName: string)`	`void`

Where it refuses work

- `RouterExplorer` stops the work with `UnknownRequestMappingException` when `isUndefined(path)`.
- `RouterExplorer` stops the work with `InternalServerErrorException` when `!next`.
- `RouterExplorer` stops the work with an early return when `Array.isArray(path)`.
- `RouterExplorer` stops the work with an early return when `!host`.

When something fails

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

Diagram

```
graph LR
  Controller[Controller instance] --> Metadata[Route metadata]
  Metadata --> RouterExplorer
  RouterExplorer --> Extract[extractRouterPath()]
  Extract --> Apply[applyPathsToRouterProxy()]
  Apply --> Proxy[Router proxy]
  Proxy --> Adapter[HTTP adapter router]
  Adapter --> Handler[Controller handler]

  Handler --> Scoped[createRequestScopedHandler()]
  Scoped --> Injector[Request-scoped injector]
```

Usage

```
import { RouterExplorer } from '@nestjs/core/router/router-explorer';

// RouterExplorer is typically created by the framework during application setup.
// This demonstrates how the framework registers a discovered controller wrapper.
routerExplorer.explore(
  controllerWrapper, // InstanceWrapper containing a controller instance
  moduleKey, // Internal module identifier
  httpAdapter, // HttpServer / platform adapter router
  undefined, // Optional host restriction
  {
    modulePath: '/api',
    controllerPath: '/users',
    methodPath: '',
  },
);
```

```
// The explorer reads @Controller(), @Get(), @Post(), and related metadata,
// then registers proxied handlers on the HTTP adapter.
```

AI Coding Instructions

- Preserve the separation between route discovery (`extractRouterPath`) and route registration (`applyPathsToRouterProxy`).
- Always register handlers through `RouterProxy` so framework exception filters and async error handling remain active.
- Use `createRequestScopedHandler` for request-scoped controllers or dependencies; do not resolve request-scoped providers from the static controller instance.
- Keep route metadata compatible with `RoutePathFactory` , including module paths, controller paths, method paths, versioning, and host constraints.
- Treat `RouterExplorer` as framework infrastructure; application code should define routes with controller decorators rather than instantiate this class directly.