RoutesResolver

August 17, 2026

RoutesResolver

Kind: Class

Source: `packages/core/router/routes-resolver.ts`

Part of: Core

`RoutesResolver` connects discovered controllers and route metadata to the underlying HTTP adapter during application initialization. It registers controller routes, configures not-found and exception handlers, and maps framework exceptions to adapter-compatible responses.

Implements: `Resolver`

Methods

Method	Signature	Returns
`resolve`	`resolve(applicationRef: T, globalPrefix: string)`	`void`
`registerRouters`	`registerRouters(routes: Map<string`	symbol
`registerNotFoundHandler`	`registerNotFoundHandler()`	`void`
`registerExceptionHandler`	`registerExceptionHandler()`	`void`
`mapExternalException`	`mapExternalException(err: any)`	`void`

Where it refuses work

- `RoutesResolver` stops the work with an early return when `versioningConfig`.

Diagram

```
graph LR
  A[Application bootstrap] --> B[RoutesResolver.resolve]
  B --> C[registerRouters]
```

```
  C --> D[Controller routes]
  D --> E[HTTP adapter router]

  B --> F[registerNotFoundHandler]
  F --> E

  B --> G[registerExceptionHandler]
  G --> H[mapExternalException]
  H --> E
```

Usage

```ts
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';

async function bootstrap() {
  const app = await NestFactory.create(AppModule);

  // RoutesResolver uses this prefix while registering controller routes.
  app.setGlobalPrefix('api');

  // During initialization, Nest internally uses RoutesResolver to:
  // - discover controller routes
  // - register them with the HTTP adapter
  // - configure 404 and exception handlers
  await app.listen(3000);
}

bootstrap();
```

AI Coding Instructions

- Treat `RoutesResolver` as framework bootstrap infrastructure; application code should usually define controllers and let the framework invoke `resolve()`.
- Preserve the registration order: controller routes should be registered before fallback not-found and exception handlers are attached.
- When adding route-related behavior, ensure global prefixes, module paths, and controller metadata are consistently applied.
- Keep adapter-specific error handling inside `mapExternalException()` so external HTTP adapters receive compatible error objects.
- Avoid registering catch-all routes or error middleware too early, as they can prevent resolved controller routes from being reached.

How it works

`RoutesResolver`

`RoutesResolver` is the HTTP-routing resolver created by `NestApplication`. During application initialization, the application invokes `resolve()` after middleware registration, then invokes its not-found and error-hook registration methods after initialization hooks. `packages/core/nest-application.ts:190-194` `packages/core/nest-application.ts:207-218`

It implements the `Resolver` contract: route resolution plus registration of not-found and exception handlers. `packages/core/router/interfaces/resolver.interface.ts:1-5`

Relationships

- IMPORTS → `BadRequestException`
- IMPORTS → `HttpException`
- IMPORTS → `NotFoundException`
- IMPORTS → `HOST_METADATA`
- IMPORTS → `MODULE_PATH`
- IMPORTS → `VERSION_METADATA`
- IMPORTS → `Controller`
- IMPORTS → `HttpServer`
- IMPORTS → `Type`
- IMPORTS → `VersionValue`
- IMPORTS → `Logger`