MetadataScanner

August 19, 2026

MetadataScanner

Kind: Class

Source: `packages/core/metadata-scanner.ts`

Part of: Core

`MetadataScanner` inspects a class prototype and its inheritance chain to discover callable method names. It is used by the framework to scan providers, controllers, and other instances for methods that may carry metadata or require registration.

Methods

Method	Signature	Returns
`scanFromPrototype`	`scanFromPrototype(instance: T, prototype: object	null, callback: (name: string) => R)`
`getAllFilteredMethodNames`	`getAllFilteredMethodNames(prototype: object)`	`IterableIterator<string>`
`getAllMethodNames`	`getAllMethodNames(prototype: object	null)`

Where it refuses work

- `MetadataScanner` stops the work with an early return when `!prototype`, in 2 places.
- `MetadataScanner` stops the work with an early return when `this.cachedScannedPrototypes.has(prototype)`.

Diagram

```
graph LR
  A[Class Instance] --> B[Prototype]
  B --> C[MetadataScanner]
  C --> D[getAllFilteredMethodNames]
```

```
  D --> E[Inherited Method Names]
  E --> F[scanFromPrototype Callback]
  F --> G[Collected Results]
```

Usage

```
import { MetadataScanner } from '@nestjs/core/metadata-scanner';

const ROUTE_METADATA = 'custom:route';

class UserController {
  @Reflect.metadata(ROUTE_METADATA, '/users')
  findAll() {
    return [];
  }

  findOne() {
    return {};
  }
}

const controller = new UserController();
const scanner = new MetadataScanner();

const routes = scanner.scanFromPrototype(
  controller,
  Object.getPrototypeOf(controller),
  (methodName) => {
    const handler = controller[methodName];
    const path = Reflect.getMetadata(ROUTE_METADATA, handler);

    return path ? { methodName, path } : undefined;
  },
).filter(Boolean);

console.log(routes);
// [{ methodName: 'findAll', path: '/users' }]
```

AI Coding Instructions

- Pass the instance and its prototype to `scanFromPrototype()` so inherited methods can be discovered correctly.
- Use the scan callback to read method-level metadata and transform discovered methods into framework-specific definitions.
- Do not rely on getters or setters being returned; the scanner is intended for callable prototype methods.

- Prefer `scanFromPrototype()` when processing methods, and use `getAllMethodNames()` only when raw method names are needed.
- Avoid scanning the same prototype repeatedly in custom integrations; the scanner caches discovered prototype method names.

Relationships

- IMPORTS → `Injectable`
- IMPORTS → `isConstructor`
- IMPORTS → `isFunction`
- IMPORTS → `isNil`

Used by

7 references from 7 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (7)

- `WebhooksExplorer` — `integration/discovery/src/webhooks.explorer.ts`:5
- `ClientProperties` — `packages/microservices/listener-metadata-explorer.ts`:16
- `ListenersController` — `packages/microservices/listeners-controller.ts`:45
- `Test` — `packages/testing/test.ts`:8
- `TestingModuleOptions` — `packages/testing/testing-module.builder.ts`:29
- `MessageMappingProperties` — `packages/websockets/gateway-metadata-explorer.ts`:15
- `WebSocketsController` — `packages/websockets/web-sockets-controller.ts`:29