EnhancerMetadataCacheEntry

August 18, 2026

EnhancerMetadataCacheEntry

Kind: Interface

Source: `packages/core/inspector/interfaces/enhancer-metadata-cache-entry.interface.ts`

Part of: Core

`EnhancerMetadataCacheEntry` describes a cached metadata record for an enhancer discovered by the NestJS inspector. It links an enhancer—such as a guard, interceptor, pipe, or filter—to its target node, owning module, class or method, and resolved instance wrapper. The inspector uses these entries to efficiently associate enhancer metadata with the serialized dependency graph.

Properties

Property	Type
`targetNodeId`	`string`
`moduleToken`	`string`
`classRef`	`Type`
`methodKey`	`string
`enhancerRef`	`unknown`
`enhancerInstanceWrapper`	`InstanceWrapper`
`subtype`	`EnhancerSubtype`

Diagram

```
graph LR
  Target[Target Node<br/>targetNodeId] --> Entry[EnhancerMetadataCacheEntry]
  Module[Module<br/>moduleToken] --> Entry
  Class[Controller or Provider<br/>classRef] --> Entry
  Method[Optional Method<br/>methodKey] --> Entry
```

```
  Enhancer[Enhancer Metadata<br/>enhancerRef] --> Entry
  Wrapper[InstanceWrapper<br/>enhancerInstanceWrapper] --> Entry
  Subtype[EnhancerSubtype<br/>Guard / Pipe / Filter / Interceptor] --> Entry
```

Usage

```
import { Type } from '@nestjs/common';
import { InstanceWrapper } from '../../injector/instance-wrapper';
import { EnhancerSubtype } from '../enums/enhancer-subtype.enum';
import { EnhancerMetadataCacheEntry } from '../interfaces/enhancer-metadata-cache-entry.interfa

function cacheEnhancer(
  entry: EnhancerMetadataCacheEntry,
  cache: Map<string, EnhancerMetadataCacheEntry[]>,
) {
  const entries = cache.get(entry.targetNodeId) ?? [];

  entries.push(entry);
  cache.set(entry.targetNodeId, entries);
}

const entry: EnhancerMetadataCacheEntry = {
  targetNodeId: 'controller:users',
  moduleToken: 'UsersModule',
  classRef: UsersController as Type,
  methodKey: 'findAll',
  enhancerRef: AuthGuard,
  enhancerInstanceWrapper: authGuardWrapper as InstanceWrapper,
  subtype: EnhancerSubtype.GUARD,
};

cacheEnhancer(entry, enhancerMetadataCache);
```

AI Coding Instructions

- Preserve the relationship between `targetNodeId`, `moduleToken`, and `enhancerInstanceWrapper`; these values identify where an enhancer belongs in the inspected graph.
- Treat `methodKey` as optional: `undefined` indicates a class-level enhancer rather than a method-level enhancer.
- Use `subtype` to distinguish enhancer categories when serializing or grouping metadata; do not infer the category solely from `enhancerRef`.
- Keep `enhancerRef` typed as `unknown` unless the consuming code has safely narrowed its runtime shape.

- When adding cache entries, avoid replacing existing entries for the same target node because multiple enhancers may apply to one controller or method.

Relationships

- IMPORTS → `Type`
- IMPORTS → `EnhancerSubtype`