ExternalHandlerMetadata

August 17, 2026

ExternalHandlerMetadata

Kind: Interface

Source: `packages/core/helpers/interfaces/external-handler-metadata.interface.ts`

Part of: Core

`ExternalHandlerMetadata` describes the runtime metadata required to invoke an external handler with dependency-injected parameters. It records the handler's expected argument count, reflected parameter types, and a resolver for retrieving parameter metadata within a specific module and dependency-injection context.

Properties

Property	Type
`argsLength`	`number`
`paramtypes`	`any[]`
`getParamsMetadata`	`( moduleKey: string, contextId?: ContextId, inquirerId?: string, ) => ParamPropertiesWithMetatype[]`

Diagram

```
graph LR
  Handler[External Handler] --> Metadata[ExternalHandlerMetadata]
  Metadata --> Args[argsLength: number]
  Metadata --> Types[paramtypes: any[]]
  Metadata --> Resolver[getParamsMetadata()]
  Resolver --> Module[moduleKey]
  Resolver --> Context[contextId / inquirerId]
  Resolver --> Params[ParamPropertiesWithMetatype[]]
```

Usage

```ts
import type { ContextId } from '@nestjs/core';
import type { ExternalHandlerMetadata } from './external-handler-metadata.interface';

const handlerMetadata: ExternalHandlerMetadata = {
  argsLength: 2,
  paramtypes: [UserService, RequestContext],
  getParamsMetadata(
    moduleKey: string,
    contextId?: ContextId,
    inquirerId?: string,
  ) {
    // Resolve parameter metadata for the active module/context.
    return resolveHandlerParams(moduleKey, contextId, inquirerId);
  },
};

const params = handlerMetadata.getParamsMetadata(
  'users-module',
  requestContextId,
);

if (params.length !== handlerMetadata.argsLength) {
  throw new Error('Resolved handler parameters do not match the handler signature.');
}
```

AI Coding Instructions

- Keep `argsLength` aligned with the number of parameters expected by the external handler.
- Preserve reflected constructor/parameter types in `paramtypes`; do not assume all entries are concrete classes.
- Call `getParamsMetadata` with the correct `moduleKey` so dependencies are resolved from the intended module scope.
- Forward `contextId` and `inquirerId` when resolving request-scoped or transient providers.
- Validate returned parameter metadata before invoking a handler, especially when metadata may be incomplete or dynamically resolved.