HandlerMetadata

August 19, 2026

HandlerMetadata

Kind: Interface

Source: `packages/core/helpers/handler-metadata-storage.ts`

Part of: Core

`HandlerMetadata` describes the runtime metadata required to invoke and serialize the result of a controller handler. It combines parameter resolution details, HTTP response configuration, SSE behavior, custom headers, and the response-handling function used by the request execution pipeline.

Properties

Property	Type
`argsLength`	`number`
`isSseHandler`	`boolean`
`paramtypes`	`any[]`
`httpStatusCode`	`number`
`responseHeaders`	`any[]`
`hasCustomHeaders`	`boolean`
`getParamsMetadata`	`( moduleKey: string, contextId?: ContextId, inquirerId?: string, ) => (ParamProperties & { metatype?: any })[]`
`fnHandleResponse`	`HandleResponseFn`

Diagram

```
graph LR
  Request[Incoming Request] --> Metadata[HandlerMetadata]
  Metadata --> Params[getParamsMetadata]
```

```
  Params --> Handler[Controller Handler]
  Metadata --> Status[httpStatusCode]
  Metadata --> Headers[responseHeaders]
  Metadata --> SSE[isSseHandler]
  Handler --> ResponseHandler[fnHandleResponse]
  Status --> ResponseHandler
  Headers --> ResponseHandler
  SSE --> ResponseHandler
  ResponseHandler --> Response[HTTP Response]
```

Usage

```typescript
import type { HandlerMetadata } from './handler-metadata-storage';

const metadata: HandlerMetadata = {
  argsLength: 2,
  isSseHandler: false,
  paramtypes: [String, Number],
  httpStatusCode: 201,
  responseHeaders: [{ name: 'x-api-version', value: 'v1' }],
  hasCustomHeaders: true,

  getParamsMetadata: (moduleKey, contextId, inquirerId) => [
    {
      index: 0,
      type: 'body',
      data: undefined,
      metatype: String,
    },
    {
      index: 1,
      type: 'param',
      data: 'id',
      metatype: Number,
    },
  ],

  fnHandleResponse: async (
    result,
    response,
    request,
  ) => {
    response.status(201);
    response.setHeader('x-api-version', 'v1');
    response.send(result);
  },
};
```

```
// The request pipeline uses this metadata to resolve handler arguments
// and apply the configured response behavior.
```

AI Coding Instructions

- Use `getParamsMetadata()` to resolve parameter metadata per module and DI context; do not treat parameter metadata as globally static when request-scoped providers are involved.
- Keep `argsLength` aligned with the target handler's declared argument positions, including optional or injected parameters.
- Set `isSseHandler` only for streaming Server-Sent Events handlers; SSE responses require different lifecycle and serialization behavior.
- When adding response headers, update both `responseHeaders` and `hasCustomHeaders` so the response pipeline applies them correctly.
- Ensure `fnHandleResponse` respects the configured HTTP status, headers, and adapter-specific response APIs.