Head

August 18, 2026

Head

Kind: Constant

Source: `packages/common/decorators/http/request-mapping.decorator.ts`

Part of: Common

Route handler (method) Decorator. Routes HTTP HEAD requests to the specified path.

`Head` is a route handler decorator that maps an HTTP `HEAD` request to a controller method. It behaves like a `GET` route for path matching, but the response should contain headers without a response body. Use it for endpoints that expose resource metadata, availability, caching information, or content length.

Definition

```
createMappingDecorator(RequestMethod.HEAD)
```

Value

```
createMappingDecorator(RequestMethod.HEAD)
```

Diagram

```
graph LR
  Client[HTTP Client] -->|HEAD /resource| Router[HTTP Router]
  Router -->|Matches path and HEAD method| HeadDecorator[@Head decorator]
  HeadDecorator --> Controller[Controller Method]
  Controller --> Headers[Response Headers]
  Headers --> Client
```

Usage

```
import { Controller, Head, HttpCode } from '@nestjs/common';

@Controller('files')
export class FilesController {
  @Head(':id')
  @HttpCode(200)
  checkFileAvailability(): void {
    // Set response metadata such as ETag, Content-Length,
    // or Last-Modified through the response object when needed.
    // HEAD responses should not include a response body.
  }
}
```

AI Coding Instructions

- Use `@Head()` on controller methods that must handle HTTP `HEAD` requests; provide a path argument when mapping a specific route.
- Keep `HEAD` handlers bodyless, since clients expect headers and status information rather than response content.
- Pair `HEAD` routes with equivalent `GET` routes when clients need both metadata-only and full-resource access.
- Configure cache-related headers, `Content-Length`, `ETag`, or `Last-Modified` through the framework response APIs when implementing resource checks.
- Ensure the route path does not conflict with other method decorators unintentionally; `HEAD` and `GET` may share a path but serve different HTTP methods.