Lock

August 18, 2026

Lock

Kind: Constant

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

Part of: Common

Route handler (method) Decorator. Routes Webdav LOCK requests to the specified path.

`Lock` is a route-handler decorator for WebDAV `LOCK` requests. Apply it to a controller method to map a WebDAV lock operation to a specified route path, allowing the handler to create, refresh, or manage resource locks within the request-routing layer.

Definition

```
createMappingDecorator(RequestMethod.LOCK)
```

Value

```
createMappingDecorator(RequestMethod.LOCK)
```

Diagram

```
graph LR
  Client[WebDAV Client] -->|LOCK request| Router[HTTP Router]
  Router -->|matches @Lock path| Handler[Controller Method]
  Handler --> LockService[Lock Handling Logic]
  LockService --> Response[WebDAV LOCK Response]
```

Usage

```ts
import { Controller, Req } from '@nestjs/common';
import { Lock } from '@your-package/common/decorators/http/request-mapping.decorator';

@Controller('files')
export class FilesController {
  @Lock(':path*')
  async lockFile(@Req() request: Request) {
    const path = request.params.path;

    // Create or refresh the WebDAV lock for the requested resource.
    return {
      path,
      status: 'locked',
    };
  }
}
```

AI Coding Instructions

- Use `@Lock()` only on controller methods intended to handle WebDAV `LOCK` requests.
- Define a route path that matches the resource hierarchy your WebDAV client may lock, such as `:path*` for nested resources.
- Ensure the handler returns a response compatible with your WebDAV lock implementation, including lock token and timeout details when required.
- Keep lock persistence, validation, and conflict detection in dedicated services rather than inside the decorator or routing layer.
- Pair `Lock` handling with related WebDAV methods such as `UNLOCK`, `PROPFIND`, and `PUT` to enforce lock state consistently.