Payload

August 18, 2026

Payload

Kind: Function

Source: `packages/microservices/decorators/payload.decorator.ts`

Part of: Microservices

`Payload()` decorates a microservice message handler parameter with the incoming message payload. It can receive pipes, such as `ValidationPipe`, to transform or validate the payload before the handler receives it.

Signature

```
function Payload(propertyOrPipe: string | (Type<PipeTransform> | PipeTransform), pipes: (Type<P
```

Parameters

Name	Type
`propertyOrPipe`	`string
`pipes`	`(Type

Returns: `ParameterDecorator`

Diagram

```
graph LR
  Message[Incoming microservice message] --> PayloadDecorator["@Payload()"]
  PayloadDecorator --> Pipes[Parameter pipes]
  Pipes --> Handler[Message handler parameter]
```

Usage

```
import { Controller, ValidationPipe } from '@nestjs/common';
import { MessagePattern, Payload } from '@nestjs/microservices';

class CreateCatDto {
  name: string;
  age: number;
}

@Controller()
export class CatsController {
  @MessagePattern('cats.create')
  create(
    @Payload(new ValidationPipe()) createDto: CreateCatDto,
  ) {
    return createDto;
  }
}
```

AI Coding Instructions

- Apply `@Payload()` only to parameters in handlers decorated with microservice pattern decorators such as `@MessagePattern()`.
- Pass parameter pipes to `@Payload()` when the incoming payload needs validation or transformation.
- Keep the decorated parameter type aligned with the expected message payload shape.
- Do not use `@Payload()` to access message metadata; use the relevant microservice context parameter for transport-specific data.

Relationships

- IMPORTS → `PipeTransform`
- IMPORTS → `Type`