EventPattern

August 17, 2026

EventPattern

Kind: Function

Source: `packages/microservices/decorators/event-pattern.decorator.ts`

Part of: Microservices

Subscribes to incoming events which fulfils chosen pattern.

`EventPattern()` marks a controller method as a handler for incoming microservice events that match a specified pattern. Unlike request-response message handlers, event handlers process published events without returning a response to the sender.

Signature

```
function EventPattern(metadata: T, transportOrExtras: Transport | symbol | Record<string, any>,
```

Parameters

Name	Type
`metadata`	`T`
`transportOrExtras`	`Transport
`maybeExtras`	`Record<string, any>`

Returns: `MethodDecorator`

Diagram

```
graph LR
  Publisher[Event Publisher] --> Transport[Microservice Transport]
  Transport --> Pattern{Event Pattern Match}
  Pattern -->|Matches| Handler["@EventPattern() Method"]
```

```
  Pattern -->|Does not match| Ignore[Ignore Event]
  Handler --> Process[Process Event Payload]
```

Usage

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

@Controller()
export class OrdersEventsController {
  @EventPattern('order.created')
  handleOrderCreated(
    @Payload() order: { id: string; customerId: string },
  ): void {
    console.log(`Processing order ${order.id} for ${order.customerId}`);
  }
}
```

AI Coding Instructions

- Apply `@EventPattern()` to microservice controller methods that consume fire-and-forget events.
- Keep event patterns consistent with the names used by publishers, such as `order.created` or `user.updated`.
- Use `@Payload()` to receive event data and validate or transform it before performing business operations.
- Do not rely on a return value from an event handler; publishers do not receive a response for event-based communication.
- Ensure the application has a configured microservice transport capable of subscribing to the selected event pattern.

Relationships

- IMPORTS → `isNil`
- IMPORTS → `isNumber`
- IMPORTS → `isObject`
- IMPORTS → `isSymbol`

Used by

7 references from 7 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (7)

- `AppController` — `integration/microservices/src/app.controller.ts`:20
- `KafkaMessagesController` — `integration/microservices/src/kafka/kafka.messages.controller.ts`:9
- `MqttController` — `integration/microservices/src/mqtt/mqtt.controller.ts`:16
- `NatsController` — `integration/microservices/src/nats/nats.controller.ts`:19
- `RedisController` — `integration/microservices/src/redis/redis.controller.ts`:12
- `RMQController` — `integration/microservices/src/rmq/rmq.controller.ts`:16
- `AppController` — `integration/microservices/src/tcp-tls/app.controller.ts`:22