RequestContextHost

August 18, 2026

RequestContextHost

Kind: Class

Source: `packages/microservices/context/request-context-host.ts`

Part of: Microservices

`RequestContextHost` is a lightweight implementation of the microservices `RequestContext` interface. It stores a message pattern, payload data, and transport-specific context, exposing them through typed getter methods for use in message handlers and interceptors.

Implements: `RequestContext`

Methods

Method	Signature	Returns
`create`	`create(pattern: string	Record<string, any>, data: TData, context: TContext)`
`getData`	`getData()`	`TData`
`getPattern`	`getPattern()`	`string`
`getContext`	`getContext()`	`TContext`

Diagram

```
graph LR
  A[Incoming microservice message] --> B[RequestContextHost]
  B --> C[getPattern()]
  B --> D[getData()]
  B --> E[getContext()]
  C --> F[Message routing metadata]
  D --> G[Payload]
  E --> H[Transport-specific context]
```

Usage

```ts
import { RequestContextHost } from '@nestjs/microservices';

interface CreateUserPayload {
  email: string;
  name: string;
}

const context = RequestContextHost.create<CreateUserPayload, { requestId: string }>(
  { cmd: 'create_user' },
  {
    email: 'ada@example.com',
    name: 'Ada Lovelace',
  },
  {
    requestId: 'req_123',
  },
);

console.log(context.getPattern()); // { cmd: 'create_user' }
console.log(context.getData().email); // ada@example.com
console.log(context.getContext().requestId); // req_123
```

AI Coding Instructions

- Preserve the generic `TData` and `TContext` types so payload and transport context remain type-safe.
- Use `RequestContextHost.create()` when constructing a context outside the normal microservice request pipeline.
- Treat the context as a read-only request snapshot; access values through `getData()`, `getPattern()`, and `getContext()`.
- Support both string and object message patterns when consuming the value returned by `getPattern()`.
- Keep transport-specific metadata in `TContext`; do not mix it into the message payload data.

How it works

`RequestContextHost<TData, TContext>` is an exported public class that implements the `RequestContext<TData>` shape and represents a request as a pattern, data payload, and RPC context. Its context type is constrained to `BaseRpcContext`; both generic types default to `any`. [packages/microservices/context/request-context-host.ts:4-10]

- Its constructor requires `pattern`, `data`, and `context`, then exposes each as a `public readonly` property. `pattern` may be a string or record; `data` has type `TData`; `context` has type `TContext`. [packages/microservices/context/request-context-host.ts:11-15]
- `getData()`, `getPattern()`, and `getContext()` return those stored properties directly. [packages/microservices/context/request-context-host.ts:26-36]
- `create()` constructs a new host from the same three inputs and returns it typed as `RequestContext<TData, TContext>`. [packages/microservices/context/request-context-host.ts:17-24]
- The class contains no runtime input validation, transformation, explicit error handling, or mutation after construction. [packages/microservices/context/request-context-host.ts:11-36]

The stored context can expose its handler arguments through `BaseRpcContext.getArgs()` and a single indexed argument through `getArgByIndex()`. [packages/microservices/ctx-host/base-rpc.context.ts:4-20]

In microservice listener dispatch, request-scoped handlers recognize a `RequestContextHost` passed as their first argument, derive a context ID from it, and remove it before invoking the handler proxy. [packages/microservices/listeners-controller.ts:238-245] When no host is first, the listener creates one from the matched pattern, incoming data, and RPC context before deriving the context ID. [packages/microservices/listeners-controller.ts:246-255] During that context-ID derivation, the listener may attach a non-enumerable request-context ID property to the host and register a request-provider value with the container. [packages/microservices/listeners-controller.ts:302-320]