ArgumentsHost

August 19, 2026

ArgumentsHost

Kind: Interface

Source: `packages/common/interfaces/features/arguments-host.interface.ts`

Part of: Common

Provides methods for retrieving the arguments being passed to a handler.
Allows choosing the appropriate execution context (e.g., Http, RPC, or WebSockets) to retrieve the arguments from.

`ArgumentsHost` provides access to the arguments supplied when a framework handler is invoked. It lets guards, interceptors, filters, and other framework components inspect the current execution context and switch to the appropriate transport-specific host, such as HTTP, RPC, or WebSockets.

Diagram

```
graph LR
  A[Handler invocation] --> B[ArgumentsHost]
  B --> C[getArgs / getArgByIndex]
  B --> D[switchToHttp]
  B --> E[switchToRpc]
  B --> F[switchToWs]
  D --> G[HTTP request, response, next]
  E --> H[RPC data and context]
  F --> I[WebSocket client and data]
```

Usage

```
import {
  ArgumentsHost,
  Catch,
  ExceptionFilter,
  HttpException,
```

```
} from '@nestjs/common';

@Catch(HttpException)
export class HttpExceptionFilter implements ExceptionFilter {
  catch(exception: HttpException, host: ArgumentsHost) {
    const http = host.switchToHttp();
    const response = http.getResponse();
    const request = http.getRequest();

    response.status(exception.getStatus()).json({
      statusCode: exception.getStatus(),
      path: request.url,
      message: exception.message,
    });
  }
}
```

AI Coding Instructions

- Use `switchToHttp()`, `switchToRpc()`, or `switchToWs()` before reading transport-specific arguments.
- Prefer the typed host adapters over accessing raw values through `getArgs()` or `getArgByIndex()`.
- Keep shared guards, interceptors, and filters transport-aware; not every `ArgumentsHost` represents an HTTP request.
- Use `getType()` when behavior must differ between HTTP, RPC, and WebSocket execution contexts.
- Avoid assuming argument positions are consistent across transports; use the relevant context adapter methods instead.

Used by

13 references from 13 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (13)

- `UnauthorizedFilter` — `integration/graphql-code-first/src/common/filters/unauthorized.filter.ts`:4
- `HttpExceptionFilter` — `integration/inspector/src/common/filters/http-exception.filter.ts`:8
- `RequestFilter` — `integration/websockets/src/request.filter.ts`:4

- `BaseExceptionFilter` — `packages/core/exceptions/base-exception-filter.ts`:17
- `ExceptionsHandler` — `packages/core/exceptions/exceptions-handler.ts`:9
- `ExternalExceptionFilter` — `packages/core/exceptions/external-exception-filter.ts`:3
- `ExternalExceptionsHandler` — `packages/core/exceptions/external-exceptions-handler.ts`:8
- `BaseRpcExceptionFilter` — `packages/microservices/exceptions/base-rpc-exception-filter.ts`:15

…and 5 more.