GuardsConsumer

August 17, 2026

GuardsConsumer

Kind: Class

Source: `packages/core/guards/guards-consumer.ts`

Part of: Core

`GuardsConsumer` evaluates a sequence of NestJS `CanActivate` guards for an incoming execution context. It creates an `ExecutionContextHost`, runs guards in order, supports synchronous, Promise-based, and Observable results, and stops when a guard denies access.

Methods

Method	Signature	Returns
`tryActivate`	`tryActivate(guards: CanActivate[], args: unknown[], instance: Controller, callback: (...args: unknown[]) => unknown, type: TContext)`	`Promise<boolean>`
`createContext`	`createContext(args: unknown[], instance: Controller, callback: (...args: unknown[]) => unknown)`	`ExecutionContextHost`
`pickResult`	`pickResult(result: boolean	Promise

Where it refuses work

- `GuardsConsumer` stops the work with an early return when `!guards || isEmpty(guards)`.
- `GuardsConsumer` stops the work with an early return when `!result`.
- `GuardsConsumer` stops the work with an early return when `result instanceof Observable`.

Diagram

```
graph LR
  A[Route Handler Arguments] --> B[GuardsConsumer.tryActivate]
  B --> C[createContext]
  C --> D[ExecutionContextHost]
  B --> E[Guard.canActivate]
  E --> F[pickResult]
  F -->|true| G[Evaluate Next Guard]
  F -->|false| H[Reject Request]
  G -->|All guards pass| I[Allow Handler Execution]
```

Usage

```
import { GuardsConsumer } from '@nestjs/core/guards/guards-consumer';
import { CanActivate, ExecutionContext } from '@nestjs/common';

class ApiKeyGuard implements CanActivate {
  canActivate(context: ExecutionContext): boolean {
    const request = context.switchToHttp().getRequest();
    return request.headers['x-api-key'] === process.env.API_KEY;
  }
}

async function checkGuards(request: any, response: any) {
  const consumer = new GuardsConsumer();
  const guards = [new ApiKeyGuard()];

  const isAllowed = await consumer.tryActivate(
    guards,
    [request, response, () => undefined],
    ExampleController.prototype,
    ExampleController.prototype.getProfile,
    'http',
  );

  if (!isAllowed) {
    throw new Error('Access denied');
  }
}

class ExampleController {
  getProfile() {
    return { id: 1 };
  }
}
```

AI Coding Instructions

- Preserve short-circuit behavior: stop evaluating guards immediately when any guard returns `false`.
- Ensure `tryActivate()` continues to support `boolean`, `Promise<boolean>`, and `Observable<boolean>` guard results through `pickResult()`.
- Use `createContext()` to construct the execution context so guards receive the controller instance, handler callback, and request arguments consistently.
- Set the context type (`http`, `rpc`, or `ws`) before invoking guards so transport-specific guard logic can use the correct context switcher.
- Avoid swallowing errors thrown by guards; failures should propagate through NestJS's normal exception handling flow.

Relationships

- IMPORTS → `CanActivate`
- IMPORTS → `ContextType`
- IMPORTS → `Controller`
- IMPORTS → `isEmpty`

Used by

4 references from 4 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (4)

- `RpcHandlerMetadata` — `packages/microservices/context/rpc-context-creator.ts`:35
- `MicroservicesModule` — `packages/microservices/microservices-module.ts`:23
- `WsHandlerMetadata` — `packages/websockets/context/ws-context-creator.ts`:34
- `SocketModule` — `packages/websockets/socket-module.ts`:27