RolesGuard

September 5, 2026

RolesGuard

Kind: Service

Source: `atloria-monorepo/apps/api/src/auth/guards/roles.guard.ts`

`RolesGuard` is a NestJS authorization guard that determines whether an authenticated request can access a route based on role requirements. It runs after authentication, reads role metadata defined on controllers or handlers, and returns a boolean indicating whether access should be granted.

Methods

Method	Signature	Returns
`canActivate`	`canActivate(context: ExecutionContext)`	`boolean`

Dependencies

- `Reflector`

Where it refuses work

- `RolesGuard` stops the work with `ForbiddenException` when `!user` — “User not authenticated”.
- `RolesGuard` stops the work with `ForbiddenException` when `!hasRole`.
- `RolesGuard` stops the work with an early return when `!requiredRoles || requiredRoles.length === 0`.

Diagram

```
sequenceDiagram
  participant Client
  participant NestJS
```

```
  participant RolesGuard
  participant Reflector
  participant RequestHandler

  Client->>NestJS: Request protected endpoint
  NestJS->>RolesGuard: canActivate(context)
  RolesGuard->>Reflector: Read required role metadata
  Reflector-->>RolesGuard: Required roles
  RolesGuard->>RolesGuard: Read authenticated user roles
  RolesGuard-->>NestJS: true or false

  alt Access granted
    NestJS->>RequestHandler: Execute controller handler
    RequestHandler-->>Client: Response
  else Access denied
    NestJS-->>Client: 403 Forbidden
  end
```

Usage

```
import { Controller, Get, UseGuards } from '@nestjs/common';
import { RolesGuard } from '@/auth/guards/roles.guard';
import { Roles } from '@/auth/decorators/roles.decorator';
import { Role } from '@/auth/enums/role.enum';

@Controller('admin')
@UseGuards(RolesGuard)
export class AdminController {
  @Get('dashboard')
  @Roles(Role.ADMIN)
  getDashboard() {
    return { message: 'Admin dashboard data' };
  }
}
```

AI Coding Instructions

- Apply `RolesGuard` to routes only after an authentication guard has populated `request.user`.
- Define required roles through the project's role metadata decorator (for example, `@Roles(...)`) rather than hardcoding authorization checks in controllers.
- Preserve handler-level and controller-level metadata lookup behavior when modifying `canActivate()`.
- Return `false` for users without the required role; let NestJS handle the resulting authorization response.

- Keep role names aligned with the application’s shared role enum or constants to avoid mismatched string values.

Relationships

- DEPENDS_ON → `reflector`

Referenced By

- `AuthModule` (MODULE_PROVIDES)
- `AuthModule` (MODULE_EXPORTS)