ResourceOrgGuard

September 4, 2026

ResourceOrgGuard

Kind: Service

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

`ResourceOrgGuard` is a NestJS authorization guard that determines whether the current request is allowed to access a resource within an organization. It runs before protected route handlers, validating the caller's organization context and resource-level organization ownership.

Methods

Method	Signature	Returns
`canActivate`	`canActivate(context: ExecutionContext)`	`Promise<boolean>`

Dependencies

- `PrismaService`
- `Reflector`

Where it refuses work

- `ResourceOrgGuard` stops the work with `ForbiddenException` when `!specs || specs.length === 0` — “Resource ownership not declared for this route”.
- `ResourceOrgGuard` stops the work with `ForbiddenException` when `!user?.organizationId` — “User not authenticated”.
- `ResourceOrgGuard` stops the work with `NotFoundException` when `org === null`.
- `ResourceOrgGuard` stops the work with `ForbiddenException` when `org !== user.organizationId` — “Access denied: this resource belongs to another organization”.

Diagram

```
sequenceDiagram
  participant Client
  participant Controller
  participant ResourceOrgGuard
  participant Request
  participant AuthService
  participant ResourceService

  Client->>Controller: Request protected resource
  Controller->>ResourceOrgGuard: canActivate(context)
  ResourceOrgGuard->>Request: Read authenticated user and route params
  ResourceOrgGuard->>AuthService: Resolve user's organization access
  AuthService-->>ResourceOrgGuard: Authorized organization context
  ResourceOrgGuard->>ResourceService: Verify resource belongs to organization
  ResourceService-->>ResourceOrgGuard: Ownership result

  alt User can access resource organization
    ResourceOrgGuard-->>Controller: true
    Controller-->>Client: Continue request
  else Unauthorized organization access
    ResourceOrgGuard-->>Controller: false / throw exception
    Controller-->>Client: 403 Forbidden
  end
```

Usage

```
import { Controller, Get, Param, UseGuards } from '@nestjs/common';
import { ResourceOrgGuard } from '@/auth/guards/resource-org.guard';

@Controller('resources')
export class ResourceController {
  @Get(':resourceId')
  @UseGuards(ResourceOrgGuard)
  async findOne(@Param('resourceId') resourceId: string) {
    // ResourceOrgGuard has already verified that the authenticated user
    // can access this resource through its organization.
    return { resourceId };
  }
}
```

AI Coding Instructions

- Apply `ResourceOrgGuard` with `@UseGuards()` on routes that expose organization-owned resources.

- Ensure authentication runs before this guard so the request contains the authenticated user or organization context it expects.
- Keep route parameter names consistent with the guard's resource lookup logic; update the guard if controller parameter names change.
- Return `true` only after confirming both user organization membership and resource-to-organization ownership.
- Use NestJS authorization exceptions for denied access rather than exposing whether a resource exists in another organization.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `reflector`

Referenced By

- `ReaderAccountModule` (MODULE_PROVIDES)