TenantContextInterceptor

September 4, 2026

TenantContextInterceptor

Kind: Service

Source: `atloria-monorepo/apps/api/src/common/tenant-context.interceptor.ts`

Binds the authenticated caller's organization into an AsyncLocalStorage context for the duration
of the request handler, so the PrismaService tenant backstop (defense-in-depth) can observe/scope
data-layer queries.

WHY an interceptor and not middleware: middleware runs BEFORE guards, so `req.user` isn't
populated yet. Interceptors run AFTER guards — the JWT is verified and `req.user` is set. We wrap
the handler's subscription inside `runWithTenantContext(...)` so the context stays active across
every awaited query the handler makes.

Requests with no authenticated user (public routes, webhooks) run with NO context — the backstop
treats that as system and does nothing, exactly like background work.

`TenantContextInterceptor` binds the authenticated caller’s organization ID to an `AsyncLocalStorage` tenant context for the lifetime of a NestJS request handler. It runs after authentication guards have populated `req.user`, allowing Prisma’s tenant backstop to scope data-layer queries made during the handler and any awaited downstream work. Unauthenticated requests intentionally run without tenant context, which is treated as system/background execution.

Methods

Method	Signature	Returns
`intercept`	`intercept(context: ExecutionContext, next: CallHandler)`	`Observable<unknown>`

Where it refuses work

- `TenantContextInterceptor` stops the work with an early return when `context.getType() !== 'http'`.
- `TenantContextInterceptor` stops the work with an early return when `!user?.organizationId`.

Diagram

```
sequenceDiagram
  participant Client
  participant Guard as JWT/Auth Guard
  participant Interceptor as TenantContextInterceptor
  participant Handler as Controller Handler
  participant Prisma as PrismaService

  Client->>Guard: HTTP request with JWT
  Guard->>Guard: Verify token and set req.user
  Guard->>Interceptor: Continue request pipeline

  alt Authenticated request
    Interceptor->>Interceptor: Read organization from req.user
    Interceptor->>Interceptor: runWithTenantContext(organizationId)
    Interceptor->>Handler: Subscribe to handler Observable
    Handler->>Prisma: Execute query
    Prisma->>Prisma: Read tenant context and apply backstop
    Prisma-->>Handler: Tenant-scoped result
  else Public or webhook request
    Interceptor->>Handler: Continue without tenant context
    Handler->>Prisma: Execute query
    Prisma->>Prisma: Treat as system/background work
  end

  Handler-->>Client: HTTP response
```

Usage

```
import { Module } from '@nestjs/common';
import { APP_INTERCEPTOR } from '@nestjs/core';
import { TenantContextInterceptor } from './common/tenant-context.interceptor';
```

```
@Module({
  providers: [
    {
      provide: APP_INTERCEPTOR,
      useClass: TenantContextInterceptor,
    },
  ],
})
export class AppModule {}
```

```
import { Controller, Get, UseGuards } from '@nestjs/common';
import { JwtAuthGuard } from '../auth/jwt-auth.guard';
import { PrismaService } from '../prisma/prisma.service';

@Controller('projects')
@UseGuards(JwtAuthGuard)
export class ProjectsController {
  constructor(private readonly prisma: PrismaService) {}

  @Get()
  async listProjects() {
    // TenantContextInterceptor has already bound req.user's organization.
    // PrismaService can apply its tenant-scoping backstop automatically.
    return this.prisma.project.findMany();
  }
}
```

AI Coding Instructions

- Register this interceptor globally so authenticated controller handlers consistently receive tenant context before Prisma queries execute.
- Keep tenant context binding inside `runWithTenantContext(...)` around the handler Observable subscription; do not set mutable global tenant state.
- Do not move this behavior into middleware: middleware executes before JWT/auth guards populate `req.user`.
- Preserve the no-context behavior for public routes, webhooks, jobs, and other system work; absence of a user must not cause tenant filtering failures.
- When changing the authenticated user shape, ensure the interceptor still reads the correct organization/tenant identifier from `req.user`.