DocsAccessGuard

September 4, 2026

DocsAccessGuard

Kind: Service

Source: `atloria-monorepo/apps/api/src/reader-access/docs-access.guard.ts`

DocsAccessGuard — the A3 project-level leak seam.

Class-level on PublicProjectController (after OptionalJwtAuthGuard + ReaderAuthGuard), so EVERY `:slugWithId` surface — docs, search, llms.txt, llms-full, page.md, search-index.json, MCP, chat — is gated for a non-PUBLIC project unless the caller presents a satisfying reader token or a member JWT. Routes that GRANT access (and the gate-screen branding) opt out with

`DocsAccessGuard` is a NestJS guard that protects every `:slugWithId` public-project surface from leaking content for non-`PUBLIC` projects. Running after `OptionalJwtAuthGuard` and `ReaderAuthGuard`, it permits access only when the project is public, the request has a satisfying reader token, or the caller has a valid member JWT. Access-granting routes and gate-screen branding routes can opt out through the guard's skip metadata.

Methods

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

Dependencies

- `Reflector`
- `DocsAccessService`

Where it refuses work

- `DocsAccessGuard` stops the work with an early return when `skip`.

Diagram

```
sequenceDiagram
  participant Client
  participant Auth as OptionalJwtAuthGuard / ReaderAuthGuard
  participant Guard as DocsAccessGuard
  participant Project as Project Access State
  participant Controller as PublicProjectController

  Client->>Auth: Request /:slugWithId/docs
  Auth->>Auth: Resolve optional member JWT and reader token
  Auth->>Guard: canActivate(request context)

  alt Route is marked to skip docs access gating
    Guard-->>Controller: Allow request
  else Route requires access validation
    Guard->>Project: Resolve project visibility and access state

    alt Project is PUBLIC
      Guard-->>Controller: Allow request
    else Valid member JWT or satisfying reader token
      Guard-->>Controller: Allow request
    else No valid access credential
      Guard-->>Client: Deny access / show gate flow
    end
  end
```

Usage

```
import { Controller, Get, Param, UseGuards } from '@nestjs/common';
import { OptionalJwtAuthGuard } from '../auth/optional-jwt-auth.guard';
import { ReaderAuthGuard } from '../reader-access/reader-auth.guard';
import { DocsAccessGuard } from '../reader-access/docs-access.guard';

@Controller(':slugWithId')
@UseGuards(OptionalJwtAuthGuard, ReaderAuthGuard, DocsAccessGuard)
export class PublicProjectController {
  @Get('docs')
  async getDocs(@Param('slugWithId') slugWithId: string) {
    // DocsAccessGuard has already verified that the project is public
    // or that the requester has member/reader access.
    return { slugWithId };
  }
}
```

AI Coding Instructions

- Keep `DocsAccessGuard` ordered after `OptionalJwtAuthGuard` and `ReaderAuthGuard`; it relies on authentication and reader-access context established by earlier guards.
- Apply the guard at the `PublicProjectController` level so all `:slugWithId` surfaces, including docs, search, MCP, chat, and generated files, receive consistent protection.
- Do not add a new project-content route without confirming it is covered by the controller-level guard or explicitly protected by equivalent access logic.
- Use the existing skip metadata only for routes that grant access or render gate-screen branding; do not bypass the guard for content-bearing endpoints.
- Preserve the three valid access paths: `PUBLIC` visibility, a satisfying reader token, or an authenticated project member JWT.

Relationships

- DEPENDS_ON → `reflector`
- DEPENDS_ON → `DocsAccessService`

Referenced By

- `ReaderAccessModule` (MODULE_PROVIDES)
- `ReaderAccessModule` (MODULE_EXPORTS)