TechDocsExportService

September 5, 2026

TechDocsExportService

Kind: Service

Source: `atloria-monorepo/apps/api/src/technical-docs/techdocs-export.service.ts`

`TechDocsExportService` is a NestJS backend service responsible for exporting technical documentation to Confluence. It encapsulates the export workflow so controllers or scheduled jobs can trigger documentation publishing without managing Confluence-specific integration details.

Methods

Method	Signature	Returns
`exportToConfluence`	`exportToConfluence(projectId: string, userId: string, dto: ConfluenceExportDto)`	`unknown`

Dependencies

- `PrismaService`
- `TechnicalDocsMaterializerService`

Where it refuses work

- `TechDocsExportService` stops the work with `NotFoundException` when `!member` — “Project not found”.
- `TechDocsExportService` stops the work with `BadRequestException` when `!baseUrl || !email || !apiToken || !spaceKey` — “baseUrl, email, apiToken and spaceKey are required.”.
- `TechDocsExportService` stops the work with `BadRequestException` when `!pages.length`.

When something fails

- `TechDocsExportService` handles failure in 1 place: it logs it and continues in all 1.

Diagram

```
sequenceDiagram
  participant Caller as Controller / Job
  participant ExportService as TechDocsExportService
  participant Confluence as Confluence API

  Caller->>ExportService: exportToConfluence()
  ExportService->>ExportService: Prepare technical documentation export
  ExportService->>Confluence: Create or update Confluence content
  Confluence-->>ExportService: Export result
  ExportService-->>Caller: Return export result
```

Usage

```
import { Injectable } from '@nestjs/common';
import { TechDocsExportService } from './techdocs-export.service';

@Injectable()
export class TechDocsExportJob {
  constructor(
    private readonly techDocsExportService: TechDocsExportService,
  ) {}

  async run() {
    const result = await this.techDocsExportService.exportToConfluence();

    return result;
  }
}
```

AI Coding Instructions

- Inject `TechDocsExportService` through NestJS dependency injection rather than constructing it manually.
- Keep Confluence export orchestration inside `exportToConfluence()`; callers should only trigger the export and handle its result.
- Preserve async behavior when consuming the export method, since external Confluence API calls may be asynchronous.
- Add clear error handling and logging around export failures, especially for authentication, network, and Confluence API errors.

- Ensure any new export data is transformed into Confluence-compatible content before sending it through the integration layer.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `TechnicalDocsMaterializerService`

Referenced By

- `TechnicalDocsController` (DEPENDS_ON)
- `TechnicalDocsModule` (MODULE_PROVIDES)