SyncJobService

September 4, 2026

SyncJobService

Kind: Service

Source: `atloria-monorepo/apps/api/src/project/sync-job.service.ts`

SyncJobService

Manages background sync jobs for reindexing documents across organization or project. Provides progress tracking and status reporting.

`SyncJobService` manages background synchronization jobs used to reindex documents for an organization or individual project. It creates and tracks job progress, exposes status information for clients, and coordinates the work required to keep indexed content current.

Methods

Method	Signature	Returns	Description
`startFullSync`	`startFullSync(dto: StartFullSyncDto, organizationId: string, userId: string)`	`Promise<SyncJobDto>`	Start a full sync job in the background
`getJobStatus`	`getJobStatus(jobId: string, organizationId: string)`	`Promise<SyncJobDto>`	Get job status
`getOrganizationJobs`	`getOrganizationJobs(organizationId: string)`	`Promise<SyncJobDto[]>`	Get all jobs for an organization
`cancelJob`	`cancelJob(jobId: string, organizationId: string)`	`Promise<SyncJobDto>`	Cancel a running job

Dependencies

- `PrismaService`
- `DocumentIndexingService`

Where it refuses work

- `SyncJobService` stops the work with `NotFoundException` when `!job || (organizationId !== undefined && job.organizationId !== organizationId)`, in 2 places.
- `SyncJobService` stops the work with an early return when `!job`.

When something fails

- `SyncJobService` handles failure in 2 places: it logs it and continues in all 2.

Diagram

```
sequenceDiagram
  participant Client
  participant Controller
  participant SyncJobService
  participant JobStore
  participant Indexer

  Client->>Controller: Request document reindex
  Controller->>SyncJobService: createSyncJob(scope)
  SyncJobService->>JobStore: Create job (pending)
  JobStore-->>SyncJobService: Job ID
  SyncJobService->>Indexer: Start background reindex

  loop For each document
    Indexer->>SyncJobService: Report progress
    SyncJobService->>JobStore: Update progress/status
  end

  Indexer->>SyncJobService: Complete or fail
  SyncJobService->>JobStore: Save final status
  Client->>Controller: Get sync job status
  Controller->>SyncJobService: getJobStatus(jobId)
  SyncJobService->>JobStore: Read job state
  JobStore-->>SyncJobService: Job status
  SyncJobService-->>Controller: Progress and result
  Controller-->>Client: Sync job status
```

Usage

```
import { Injectable } from '@nestjs/common';
import { SyncJobService } from './sync-job.service';

@Injectable()
export class ProjectReindexService {
  constructor(private readonly syncJobService: SyncJobService) {}
```

```
  async reindexProject(projectId: string, organizationId: string) {
    const job = await this.syncJobService.createSyncJob({
      organizationId,
      projectId,
    });

    return {
      jobId: job.id,
      status: job.status,
    };
  }

  async getReindexStatus(jobId: string) {
    return this.syncJobService.getJobStatus(jobId);
  }
}
```

AI Coding Instructions

- Create sync jobs before starting long-running reindex work so callers can immediately receive a job identifier and poll for status.
- Update progress consistently during document processing, including terminal `completed` or `failed` states.
- Preserve organization and project scope when creating or querying jobs to prevent cross-tenant status access.
- Keep expensive indexing operations asynchronous; do not block HTTP request handlers while a sync job is running.
- Propagate indexing failures to the job record with actionable error details so clients can distinguish retryable failures from completed jobs.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `DocumentIndexingService`

Referenced By

- `ProjectModule` (MODULE_PROVIDES)
- `ProjectModule` (MODULE_EXPORTS)
- `SyncController` (DEPENDS_ON)