SyncService

September 5, 2026

SyncService

Kind: Service

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

`SyncService` coordinates batch synchronization work for the API. Its `syncBatch()` method executes the configured sync flow and returns a `SyncBatchResponseDto` describing the outcome, making it the primary backend integration point for consumers that need to trigger a batch sync.

Methods

Method	Signature	Returns	Description
`syncBatch`	`syncBatch(projectId: string, dto: SyncBatchDto, user: JwtPayload)`	`Promise<SyncBatchResponseDto>`	Sync batch of documents to database

Dependencies

- `PrismaService`
- `DocumentService`
- `DocumentVersionService`
- `DocumentIndexingService`

Where it refuses work

- `SyncService` stops the work with `BadRequestException` when `dto.documents.length > this.MAX_BATCH_SIZE`.
- `SyncService` stops the work with `BadRequestException` when `dto.documents.length === 0` — “Batch cannot be empty”.

- `SyncService` stops the work with `BadRequestException` when `size > this.MAX_DOCUMENT_SIZE`.
- `SyncService` stops the work with `BadRequestException` when `!doc.title || !doc.slug || !doc.content`.
- `SyncService` stops the work with `BadRequestException` when `!slugPattern.test(doc.slug)`.
- `SyncService` stops the work with `BadRequestException` when `!project` — “Project not found or you do not have access”.

When something fails

- `SyncService` handles failure in 5 places: it logs it and continues in 3, turns it into a return value in 1, and lets it reach the caller in 1.

Diagram

```
sequenceDiagram
  participant Client
  participant Controller
  participant SyncService
  participant SyncBatchResponseDto

  Client->>Controller: Request batch synchronization
  Controller->>SyncService: syncBatch()
  SyncService->>SyncService: Execute batch sync workflow
  SyncService-->>Controller: Promise<SyncBatchResponseDto>
  Controller-->>Client: Return sync result
```

Usage

```
import { Injectable } from '@nestjs/common';
import { SyncService } from './sync.service';
import { SyncBatchResponseDto } from './dto/sync-batch-response.dto';

@Injectable()
export class SyncController {
  constructor(private readonly syncService: SyncService) {}

  async syncBatch(): Promise<SyncBatchResponseDto> {
    return this.syncService.syncBatch();
  }
}
```

AI Coding Instructions

- Keep batch synchronization orchestration inside `SyncService`; controllers should only validate requests and delegate to `syncBatch()`.
- Preserve the `Promise<SyncBatchResponseDto>` return contract so API consumers receive a consistent synchronization result.
- Handle failures within the service using NestJS-compatible exceptions or the project's established error-handling conventions.
- Avoid introducing long-running synchronous work; await external I/O and batch operations so the NestJS event loop remains responsive.
- Update dependent controllers, DTOs, and tests whenever the batch sync response shape or workflow changes.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `DocumentService`
- DEPENDS_ON → `DocumentVersionService`
- DEPENDS_ON → `DocumentIndexingService`

Referenced By

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