DocumentIndexingService

September 4, 2026

DocumentIndexingService

Kind: Service

Source: `atloria-monorepo/apps/api/src/documentation/services/document-indexing.service.ts`

Document Indexing Service

Coordinates documentation indexing to Azure AI Search.

Synchronization Strategy:

1. Event-driven: Index immediately after document create/update
2. Batch fallback: Hourly cron job syncs any missed documents

Multi-Tenancy: All indexed documents include projectId for filtering

Database as Source of Truth: PostgreSQL is authoritative, Azure AI Search is read-optimized replica

`DocumentIndexingService` coordinates synchronizing documentation records from PostgreSQL (source of truth) into Azure AI Search (read-optimized index). It supports immediate, event-driven indexing on document create/update and an hourly batch fallback cron job to catch missed changes. All indexed records include `projectId` to enforce multi-tenant filtering in search queries.

Methods

Method	Signature	Returns	Description
`onModuleInit`	`onModuleInit()`	`unknown`	Initialize search index on module startup
`indexWorkflow`	`indexWorkflow(documentId: string, workflow: any, projectId: string, organizationId: string, versionInfo: { docVersionId?: string; versionNumber?: number })`	`Promise<void>`	Index a workflow document

Method	Signature	Returns	Description
`indexFeature`	`indexFeature(documentId: string, feature: any, projectId: string, organizationId: string, versionInfo: { docVersionId?: string; versionNumber?: number })`	`Promise<void>`	Index a feature document
`indexTutorial`	`indexTutorial(documentId: string, tutorial: any, projectId: string, organizationId: string, versionInfo: { docVersionId?: string; versionNumber?: number })`	`Promise<void>`	Index a tutorial document
`indexMdxDocument`	`indexMdxDocument(documentId: string, document: { title: string; content: string; type?: string; audienceIds?: string[] }, projectId: string, organizationId: string, docVersionId: string, versionNumber: number)`	`Promise<void>`	Index an MDX/plain-text document (user manuals, AI-generated docs, wiki pages)
`updateIndexedDocument`	`updateIndexedDocument(documentId: string, documentType: 'workflow' | 'feature' | 'tutorial', content: any, projectId: string, organizationId: string, versionInfo: { docVersionId?: string; versionNumber?: number })`		
`deleteIndexedDocument`	`deleteIndexedDocument(documentId: string)`	`Promise<void>`	Delete a document from search index
`deleteProjectIndex`	`deleteProjectIndex(projectId: string)`	`Promise<void>`	Delete all documents for a project
`removeVersionsFromIndex`	`removeVersionsFromIndex(projectId: string, docVersionIds: string[])`	`Promise<void>`	Remove every indexed document belonging to the given doc versions from the search index.

Method	Signature	Returns	Description
`syncUnindexedDocuments`	`syncUnindexedDocuments()`	`Promise<void>`	Batch sync: index documents that were never indexed OR were edited after their last indexing (updatedAt > searchIndexedAt — the indexing writes pin the two e…
`reindexProject`	`reindexProject(projectId: string)`	`Promise<void>`	Re-index all documents for a project (useful for schema changes)

Dependencies

- `PrismaService`
- `AzureSearchService`
- `KeywordExtractorService`
- `SnippetsService` *(optional)*

Where it refuses work

- `DocumentIndexingService` stops the work with an early return when `!doc`.
- `DocumentIndexingService` stops the work with an early return when `docVersionIds.length === 0`.

When something fails

- `DocumentIndexingService` handles failure in 14 places: it logs it and continues in 10, turns it into a return value in 2, lets it reach the caller in 1, and discards it silently in 1. A failure discarded silently leaves no trace for whoever debugs this later.

Diagram

```
sequenceDiagram
  autonumber
```

```
  participant App as API/NestJS
  participant DB as PostgreSQL
  participant DIS as DocumentIndexingService
  participant Search as Azure AI Search
  participant Cron as Hourly Cron

  App->>DB: Create/Update Document (projectId, content, metadata)
  DB-->>App: Commit OK
  App->>DIS: Emit/Handle document.changed event
  DIS->>DB: Load authoritative document by id
  DB-->>DIS: Document payload
  DIS->>Search: Upsert document into index (includes projectId)
  Search-->>DIS: Ack

  Cron->>DIS: Run hourly sync job
  DIS->>DB: Query for missed/out-of-sync documents
  DB-->>DIS: List of documents
  loop For each document
    DIS->>Search: Upsert document into index (includes projectId)
    Search-->>DIS: Ack
  end
```

Usage

```typescript
import { Injectable } from '@nestjs/common';
import { DocumentIndexingService } from './documentation/services/document-indexing.service';

@Injectable()
export class DocumentsEventsHandler {
  constructor(private readonly indexing: DocumentIndexingService) {}

  // Example: called after a document is created/updated (event-driven path)
  async onDocumentChanged(documentId: string) {
    // Implementation detail depends on the service API:
    // call the service method that performs a DB fetch + Azure Search upsert.
    await this.indexing.indexDocumentById(documentId);
  }

  // Example: cron/batch fallback entrypoint (hourly)
  async hourlyReindex() {
    await this.indexing.syncMissedDocuments();
  }
}
```

AI Coding Instructions

- Always treat PostgreSQL as the source of truth: fetch the latest document from DB before indexing; never rely on stale event payloads for final index content.

- Ensure every Azure AI Search document includes `projectId` and keep it filterable; missing `projectId` breaks multi-tenant isolation.
- Preserve idempotency: indexing operations should be safe to retry (upsert, not insert-only) because events/cron can overlap.
- Keep event-driven and cron paths consistent by reusing the same internal indexing routine (normalization, field mapping, serialization).
- When changing document schema/fields, update both DB-to-index mapping and any Azure Search index configuration (fields, analyzers) in lockstep to avoid runtime indexing errors.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `AzureSearchService`
- DEPENDS_ON → `KeywordExtractorService`
- DEPENDS_ON → `SnippetsService`

Referenced By

- `DocumentService` (DEPENDS_ON)
- `DocumentationModule` (MODULE_PROVIDES)
- `DocumentationModule` (MODULE_EXPORTS)
- `DocAutomationService` (DEPENDS_ON)
- `SyncJobService` (DEPENDS_ON)
- `SyncController` (DEPENDS_ON)
- `SyncService` (DEPENDS_ON)
- `TechnicalDocsParserService` (DEPENDS_ON)
- `TechnicalDocsService` (DEPENDS_ON)