DocumentPersistenceService

September 4, 2026

DocumentPersistenceService

Kind: Service

Source: `atloria-monorepo/apps/api/src/collaboration/document-persistence.service.ts`

Service for managing periodic document saves
Handles auto-save functionality for collaborative documents

`DocumentPersistenceService` manages periodic persistence for collaborative documents in the NestJS API. It coordinates auto-save scheduling and write operations so active document changes are stored reliably without requiring clients to explicitly save every update. It sits between the collaboration layer that receives edits and the underlying document storage mechanism.

Methods

Method	Signature	Returns	Description
`onModuleDestroy`	`onModuleDestroy()`	`void`	Cleanup on module destroy
`startPeriodicSave`	`startPeriodicSave(documentId: string, ydoc: Y.Doc, intervalMs: number)`	`() => void`	Start periodic save for a document
`stopPeriodicSave`	`stopPeriodicSave(documentId: string)`	`void`	Stop periodic save for a document
`stopAllPeriodicSaves`	`stopAllPeriodicSaves()`	`void`	Stop all periodic saves
`saveDocument`	`saveDocument(documentId: string)`	`Promise<boolean>`	Immediately save a document

Method	Signature	Returns	Description
`saveAllDocuments`	`saveAllDocuments()`	`Promise<Map<string, boolean>>`	Save all active documents Useful for graceful shutdown
`hasPeriodicSave`	`hasPeriodicSave(documentId: string)`	`boolean`	Check if a document has active periodic saves
`getActiveDocumentCount`	`getActiveDocumentCount()`	`number`	Get the number of documents with active periodic saves
`getActiveDocumentIds`	`getActiveDocumentIds()`	`string[]`	Get list of document IDs with active periodic saves
`updateDocument`	`updateDocument(documentId: string, ydoc: Y.Doc)`	`void`	Update the Y.Doc reference for a document Useful when Y.Doc is replaced
`flushAll`	`flushAll()`	`Promise<void>`	Trigger an immediate save for all documents Does not wait for the scheduled interval
`getDocumentContent`	`getDocumentContent(documentId: string)`	`string	null`	
`hasPendingChanges`	`hasPendingChanges(documentId: string)`	`Promise<boolean>`	Check if there are any pending changes that haven't been saved This is a

Method	Signature	Returns	Description
			basic implementation - for production, you might want to track changes more precisel…

Dependencies

- `CollaborationService`

Where it refuses work

- `DocumentPersistenceService` stops the work with an early return when `!ydoc`.
- `DocumentPersistenceService` stops the work with an early return when `currentContent === null`.

When something fails

- `DocumentPersistenceService` handles failure in 2 places: it turns it into a return value in all 2.

Diagram

```
sequenceDiagram
    participant Client as Collaboration Client
    participant Collaboration as Collaboration Service
    participant Persistence as DocumentPersistenceService
    participant Storage as Document Repository/Database

    Client->>Collaboration: Send document update
    Collaboration->>Persistence: Queue document for auto-save
    Persistence->>Persistence: Reset or schedule save timer

    Note over Persistence: Auto-save interval expires

    Persistence->>Storage: Persist latest document state
    Storage-->>Persistence: Save result
    Persistence-->>Collaboration: Document persistence completed
```

Usage

```
import { Injectable } from '@nestjs/common';
import { DocumentPersistenceService } from './document-persistence.service';

@Injectable()
export class CollaborationService {
  constructor(
    private readonly documentPersistenceService: DocumentPersistenceService,
  ) {}

  async handleDocumentUpdate(documentId: string, content: unknown) {
    // Apply the update to the in-memory collaboration document first.
    // Then queue persistence rather than writing on every client update.
    this.documentPersistenceService.scheduleSave(documentId, content);
  }

  async closeDocument(documentId: string, content: unknown) {
    // Flush pending changes when a collaboration session ends.
    await this.documentPersistenceService.saveNow(documentId, content);
  }
}
```

AI Coding Instructions

- Queue or debounce saves through `DocumentPersistenceService`; avoid writing to the database for every collaboration update.
- Always flush pending document changes during document/session shutdown to prevent losing updates still waiting for the auto-save timer.
- Ensure persistence operations use the latest authoritative document state, not a stale update payload captured when a timer was created.
- Clean up timers and in-memory save tracking when documents are closed or deleted to prevent memory leaks.
- Propagate or log persistence failures with enough document context for retry and operational debugging.

Relationships

- DEPENDS_ON → `CollaborationService`

Referenced By

- `CollaborationModule` (MODULE_PROVIDES)

- `CollaborationModule` (MODULE_EXPORTS)