Neo4jService

September 4, 2026

Neo4jService

Kind: Service

Source: `atloria-monorepo/apps/api/src/graph/services/neo4j.service.ts`

`Neo4jService` is a NestJS service that manages the Neo4j driver lifecycle and provides a shared database access layer for the API. It initializes the driver when the module starts, closes it during shutdown, and exposes helpers for opening sessions and executing read or write Cypher queries.

Methods

Method	Signature	Returns
`onModuleInit`	`onModuleInit()`	`unknown`
`onModuleDestroy`	`onModuleDestroy()`	`unknown`
`getSession`	`getSession()`	`Session`
`query`	`query(cypher: string, params: any)`	`Promise<any[]>`
`write`	`write(cypher: string, params: any)`	`Promise<any>`

Dependencies

- `ConfigService`

When something fails

- `Neo4jService` handles failure in 1 place: it logs it and continues in all 1.

Diagram

```
sequenceDiagram
  participant App as NestJS Application
  participant Service as Neo4jService
  participant Driver as Neo4j Driver
  participant DB as Neo4j Database

  App->>Service: onModuleInit()
  Service->>Driver: Create/connect driver

  App->>Service: query(cypher, params)
  Service->>Driver: session()
  Driver->>DB: Run read query
  DB-->>Service: Records
  Service-->>App: Promise<any[]>

  App->>Service: write(cypher, params)
  Service->>Driver: session()
  Driver->>DB: Run write query
  DB-->>Service: Write result
  Service-->>App: Promise<any>

  App->>Service: onModuleDestroy()
  Service->>Driver: close()
```

Usage

```
import { Injectable } from '@nestjs/common';
import { Neo4jService } from '../graph/services/neo4j.service';

@Injectable()
export class UserGraphService {
  constructor(private readonly neo4jService: Neo4jService) {}

  async findUserById(userId: string) {
    const records = await this.neo4jService.query(
      `
        MATCH (user:User { id: $userId })
        RETURN user
      `,
      { userId },
    );

    return records[0] ?? null;
  }

  async createFollowRelationship(userId: string, targetUserId: string) {
    return this.neo4jService.write(
      `
        MATCH (user:User { id: $userId })
```

```
        MATCH (target:User { id: $targetUserId })
        MERGE (user)-[:FOLLOWS]->(target)
      `,
      { userId, targetUserId },
    );
  }
}
```

AI Coding Instructions

- Inject `Neo4jService` into NestJS providers instead of creating Neo4j drivers or sessions directly.
- Use `query()` for read-oriented Cypher operations and `write()` for mutations such as `CREATE`, `MERGE`, `SET`, or `DELETE`.
- Always pass dynamic Cypher values as parameters (for example, `$userId`) rather than interpolating user input into query strings.
- Do not manually call `onModuleInit()` or `onModuleDestroy()`; NestJS invokes these lifecycle hooks automatically.
- Use `getSession()` only when a custom transaction or lower-level Neo4j session handling is required, and ensure the session is closed after use.

Relationships

- DEPENDS_ON → `configservice`

Referenced By

- `Neo4jDocsOrchestratorService` (DEPENDS_ON)
- `GraphModule` (MODULE_PROVIDES)
- `GraphModule` (MODULE_EXPORTS)
- `KgSyncService` (DEPENDS_ON)
- `KnowledgeGraphService` (DEPENDS_ON)
- `TechDocsRagService` (DEPENDS_ON)
- `WorkflowStorageService` (DEPENDS_ON)