EntityExtractorService

September 4, 2026

EntityExtractorService

Kind: Service

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

`EntityExtractorService` is a NestJS backend service responsible for extracting `KGEntity` records from a document. It supports the graph subsystem by transforming document content into entities that can be persisted, linked, or used to build knowledge-graph relationships.

Methods

Method	Signature	Returns
`extractFromDocument`	`extractFromDocument(documentId: string)`	`Promise<KGEntity[]>`

Dependencies

- `PrismaService`
- `AIService`

Where it refuses work

- `EntityExtractorService` stops the work with an early return when `!document`.

When something fails

- `EntityExtractorService` handles failure in 2 places: it logs it and continues in 1, and turns it into a return value in 1.

Diagram

```
sequenceDiagram
    participant Caller as Graph Workflow
    participant Extractor as EntityExtractorService
    participant Document as Source Document
    participant KG as Knowledge Graph

    Caller->>Extractor: extractFromDocument()
    Extractor->>Document: Read document content
    Extractor->>Extractor: Identify and normalize entities
    Extractor-->>Caller: Promise<KGEntity[]>
    Caller->>KG: Store or link extracted entities
```

Usage

```
import { Injectable } from '@nestjs/common';
import { EntityExtractorService } from './entity-extractor.service';
import type { KGEntity } from '../types/kg-entity.type';

@Injectable()
export class GraphIngestionService {
  constructor(
    private readonly entityExtractorService: EntityExtractorService,
  ) {}

  async extractEntities(): Promise<KGEntity[]> {
    const entities =
      await this.entityExtractorService.extractFromDocument();

    // Persist entities or create graph relationships here.
    return entities;
  }
}
```

AI Coding Instructions

- Keep entity extraction logic inside `EntityExtractorService`; callers should consume the returned `KGEntity[]` rather than duplicate parsing logic.
- Treat `extractFromDocument()` as asynchronous and always `await` its result before creating graph nodes or relationships.
- Preserve the expected `KGEntity` shape when adding normalization or enrichment logic so downstream graph operations remain compatible.
- Handle empty extraction results gracefully; an empty array can be valid when no entities are found.

- Register and inject the service through its NestJS module rather than constructing it manually.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `AIService`

Referenced By

- `GraphController` (DEPENDS_ON)
- `GraphModule` (MODULE_PROVIDES)
- `GraphModule` (MODULE_EXPORTS)