ParserService

September 4, 2026

ParserService

Kind: Service

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

`ParserService` is a NestJS backend service responsible for parsing source content and returning structured `ParseResult` data. It exposes document-level parsing through `parseDocument()` and provides a list of supported parser languages through `getSupportedLanguages()` for API consumers and integrations.

Methods

Method	Signature	Returns
`parse`	`parse(code: string, language: string, filename: string)`	`Promise<ParseResult>`
`parseDocument`	`parseDocument(documentId: string, organizationId: string)`	`Promise<any>`
`getSupportedLanguages`	`getSupportedLanguages()`	`string[]`

Dependencies

- `PrismaService`
- `TypeScriptParser`
- `JavaScriptParser`
- `PythonParser`

Where it refuses work

- `ParserService` stops the work with `BadRequestException` when `!parser`.
- `ParserService` stops the work with `NotFoundException` when `!document` — “Document not found”.

When something fails

- `ParserService` handles failure in 1 place: it turns it into a return value in all 1.

Diagram

```
sequenceDiagram
    participant Client
    participant ParserService
    participant Parser as Language Parser

    Client->>ParserService: getSupportedLanguages()
    ParserService-->>Client: string[]

    Client->>ParserService: parse()
    ParserService->>Parser: Parse configured source input
    Parser-->>ParserService: ParseResult
    ParserService-->>Client: Promise<ParseResult>

    Client->>ParserService: parseDocument()
    ParserService->>Parser: Parse document content
    Parser-->>ParserService: Parsed document data
    ParserService-->>Client: Promise<any>
```

Usage

```
import { Injectable } from '@nestjs/common';
import { ParserService } from './parser/parser.service';

@Injectable()
export class DocumentAnalysisService {
  constructor(private readonly parserService: ParserService) {}

  async analyzeDocument() {
    const supportedLanguages = this.parserService.getSupportedLanguages();

    const result = await this.parserService.parse();

    return {
      supportedLanguages,
      result,
    };
  }

  async parseUploadedDocument() {
    const documentResult = await this.parserService.parseDocument();

    return documentResult;
```

```
  }
}
```

AI Coding Instructions

- Inject `ParserService` through NestJS dependency injection rather than constructing it directly.
- Use `parse()` when consumers require the service's standard `ParseResult` contract.
- Use `parseDocument()` for document-specific parsing flows; validate or narrow its `any` return value before exposing it to typed application code.
- Check `getSupportedLanguages()` before accepting or routing language-specific parsing requests.
- Preserve async error handling around parsing calls so parser failures can be translated into appropriate API exceptions.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `TypeScriptParser`
- DEPENDS_ON → `JavaScriptParser`
- DEPENDS_ON → `pythonparser`