AIService

September 4, 2026

AIService

Kind: Service

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

`AIService` is a NestJS backend service that centralizes access to configured AI providers and document-generation capabilities. It resolves the active provider, exposes provider discovery, and supports document enhancement, outlining, writing improvement, summarization, text generation, and streaming completions.

Methods

Method	Signature	Returns	Description
`getProvider`	`getProvider(name: string)`	`AIProvider`	Get AI provider by name or default from config
`listProviders`	`listProviders()`	`unknown`	List all available providers
`enhanceDocument`	`enhanceDocument(documentId: string, providerName: string)`	`Promise<any>`	Enhance a document with AI suggestions
`generateOutline`	`generateOutline(topic: string, audience: string, providerName: string)`	`Promise<string>`	Generate an outline for a topic
`improveWriting`	`improveWriting(text: string, providerName: string)`	`Promise<string>`	Improve writing quality of text

Method	Signature	Returns	Description
`summarizeDocument`	`summarizeDocument(documentId: string, providerName: string)`	`Promise<string>`	Summarize a document
`streamCompletion`	`streamCompletion(prompt: string, providerName: string)`	`AsyncIterableIterator<string>`	Stream AI completion
`generateText`	`generateText(prompt: string, providerName: string)`	`Promise<string>`	Generate text using specified provider

Dependencies

- `PrismaService`
- `ConfigService`
- `AzureClaudeProvider`
- `AzureOpenAIProvider`
- `AzureResponsesProvider`
- `AzureResponsesProvider`

Where it refuses work

- `AIService` stops the work with `NotFoundException` when `!document` — “Document not found”, in 2 places.
- `AIService` stops the work with `NotFoundException` when `!provider`.

Diagram

```
sequenceDiagram
    participant Client as API Controller
    participant Service as AIService
    participant Provider as AIProvider
    participant Model as AI Model API

    Client->>Service: generateOutline(document)
    Service->>Service: getProvider()
    Service->>Provider: generateText(prompt/options)
    Provider->>Model: completion request
    Model-->>Provider: generated text
    Provider-->>Service: outline result
    Service-->>Client: Promise<string>

    Client->>Service: streamCompletion(prompt)
    Service->>Provider: streamCompletion(prompt)
```

```
    Provider-->>Service: AsyncIterableIterator<string>
    Service-->>Client: stream tokens
```

Usage

```
import { Injectable } from '@nestjs/common';
import { AIService } from './ai.service';

@Injectable()
export class DocumentService {
  constructor(private readonly aiService: AIService) {}

  async createOutline(content: string): Promise<string> {
    return this.aiService.generateOutline(content);
  }

  async improveContent(content: string): Promise<string> {
    return this.aiService.improveWriting(content);
  }

  async streamResponse(prompt: string): Promise<void> {
    for await (const chunk of this.aiService.streamCompletion(prompt)) {
      process.stdout.write(chunk);
    }
  }

  getAvailableProviders() {
    return this.aiService.listProviders();
  }
}
```

AI Coding Instructions

- Use `AIService` as the application-level integration point rather than calling an `AIProvider` directly from controllers or feature services.
- Call `getProvider()` only when provider-specific behavior is necessary; prefer high-level methods such as `summarizeDocument()` or `improveWriting()` for document workflows.
- Preserve streaming behavior by consuming `streamCompletion()` with `for await...of`; do not eagerly collect chunks unless a complete response is required.
- Handle provider failures, missing configuration, and malformed AI output at the controller or calling-service boundary.
- Add new AI capabilities as focused service methods that delegate to the active provider and return consistently typed results.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `configservice`
- DEPENDS_ON → `AzureClaudeProvider`
- DEPENDS_ON → `AzureOpenAIProvider`
- DEPENDS_ON → `azureresponsesprovider`
- DEPENDS_ON → `azureresponsesprovider`

Referenced By

- `AIController` (DEPENDS_ON)
- `AIModule` (MODULE_PROVIDES)
- `AIModule` (MODULE_EXPORTS)
- `CodeAnalysisService` (DEPENDS_ON)
- `PlaywrightGeneratorService` (DEPENDS_ON)
- `UIAnalyzerService` (DEPENDS_ON)
- `EntityExtractorService` (DEPENDS_ON)
- `PublicProjectController` (DEPENDS_ON)