WebCrawlerService

September 4, 2026

WebCrawlerService

Kind: Service

Source: `atloria-monorepo/apps/api/src/documentation/services/web-crawler.service.ts`

Simple web crawler for documentation pages
Fetches and converts HTML to markdown

`WebCrawlerService` is a NestJS service that retrieves documentation pages from the web and converts their HTML content into Markdown. It provides a reusable backend integration point for importing external documentation into the application's documentation pipeline.

Methods

Method	Signature	Returns	Description
`crawlPage`	`crawlPage(url: string)`	`Promise<CrawledPage`	null>`
`crawlPages`	`crawlPages(urls: string[], concurrency: unknown)`	`Promise<CrawledPage[]>`	Crawl multiple pages concurrently
`extractText`	`extractText(page: CrawledPage)`	`string`	Extract just the text content (no markdown formatting)
`summarize`	`summarize(page: CrawledPage, maxLength: unknown)`	`string`	Summarize page content (first N characters)
`validateUrl`	`validateUrl(url: string)`	`Promise<boolean>`	Validate if URL is accessible
`healthCheck`	`healthCheck()`	`Promise<boolean>`	Health check

Where it refuses work

- `WebCrawlerService` stops the work with an early return when `text.length <= maxLength`.

When something fails

- `WebCrawlerService` handles failure in 3 places: it turns it into a return value in all 3.

Diagram

```
sequenceDiagram
    participant Client as Documentation Consumer
    participant Service as WebCrawlerService
    participant Remote as External Documentation Site
    participant Converter as HTML-to-Markdown Converter

    Client->>Service: crawl(url)
    Service->>Remote: Fetch HTML page
    Remote-->>Service: HTML response
    Service->>Converter: Convert HTML content
    Converter-->>Service: Markdown content
    Service-->>Client: Return crawled Markdown
```

Usage

```
import { Injectable } from '@nestjs/common';
import { WebCrawlerService } from './web-crawler.service';

@Injectable()
export class DocumentationImportService {
  constructor(private readonly webCrawlerService: WebCrawlerService) {}

  async importExternalDocs(url: string) {
    const markdown = await this.webCrawlerService.crawl(url);

    return {
      sourceUrl: url,
      content: markdown,
    };
  }
}
```

AI Coding Instructions

- Keep network access inside `WebCrawlerService`; callers should provide URLs and consume normalized Markdown results.
- Validate and sanitize input URLs before crawling, especially when URLs originate from user-controlled sources.
- Handle fetch failures, redirects, timeouts, and non-HTML responses with clear NestJS-compatible errors.
- Preserve meaningful document structure during HTML-to-Markdown conversion, including headings, links, code blocks, and lists.
- Integrate the service through NestJS dependency injection rather than constructing it manually.

Referenced By

- `CompetitorResearchService` (DEPENDS_ON)