PlaywrightGeneratorService

September 4, 2026

PlaywrightGeneratorService

Kind: Service

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

Playwright Test Generator Service

Generates comprehensive Playwright tests from UI documentation:

1. Page Object Models
2. Test suites with fixtures
3. Selectors and utilities
4. Test data generators

`PlaywrightGeneratorService` is a NestJS service that converts UI documentation into a complete Playwright testing package. It generates page object models, fixture-based test suites, selectors, utilities, and test-data helpers, then can write the generated suite to the configured output location.

Methods

Method	Signature	Returns	Description
`generateTestSuite`	`generateTestSuite(pageDoc: {		

```
path: string;
component: string;
title: string;
description: string;
elements: any[];
validations: any[];
interactions: any[];
```

}) | Promise | `Generate complete Playwright test suite for a page` |

| writeTestSuite | writeTestSuite(suite: PlaywrightTestSuite, outputDir:

string) | Promise` | Write test suite to files |

Dependencies

- `AIService`

Diagram

```
sequenceDiagram
  participant Caller as API/Orchestrator
  participant Generator as PlaywrightGeneratorService
  participant UI as UI Documentation
  participant Suite as PlaywrightTestSuite
  participant FS as File System

  Caller->>Generator: generateTestSuite(uiDocumentation)
  Generator->>UI: Parse pages, components, flows, and selectors
  Generator->>Generator: Generate POMs, fixtures, tests, utilities, and data
  Generator-->>Caller: PlaywrightTestSuite

  Caller->>Generator: writeTestSuite(testSuite)
  Generator->>Suite: Prepare generated source files
  Generator->>FS: Write Playwright project files
  Generator-->>Caller: Promise<void>
```

Usage

```
import { Injectable } from '@nestjs/common';
import { PlaywrightGeneratorService } from './ai/services/playwright-generator.service';

@Injectable()
export class TestGenerationJob {
  constructor(
    private readonly playwrightGenerator: PlaywrightGeneratorService,
  ) {}

  async generateAndWriteTests(): Promise<void> {
    const testSuite = await this.playwrightGenerator.generateTestSuite();

    await this.playwrightGenerator.writeTestSuite();

    console.log(`Generated Playwright suite: ${testSuite.name}`);
  }
}
```

AI Coding Instructions

- Keep generated artifacts organized by responsibility: page objects, fixtures, specs, selectors, and test-data utilities should remain separate outputs.
- Preserve Playwright best practices in generated tests, including role- or test-id-based selectors, fixture usage, and isolated test state.
- Update both `generateTestSuite()` and `writeTestSuite()` when introducing a new generated artifact type so it is included in the in-memory suite and persisted to disk.
- Avoid generating brittle selectors based on CSS structure, text that changes frequently, or implementation-specific DOM details.
- Treat UI documentation as the source of truth; validate that documented pages, user flows, and selectors are represented in the generated suite.

Relationships

- DEPENDS_ON → `AIService`

Referenced By

- `AIModule` (MODULE_PROVIDES)
- `AIModule` (MODULE_EXPORTS)