PlaywrightService

September 4, 2026

PlaywrightService

Kind: Service

Source: `atloria-monorepo/apps/screenshot-worker/src/app/playwright/playwright.service.ts`

Playwright service for browser automation and screenshots

Features:

- Multiple browser support (Chromium, Firefox, WebKit)
- Custom viewport sizes
- Full-page screenshots
- Wait for selectors/timeouts
- Basic authentication
- Browser pooling for performance

`PlaywrightService` provides browser automation and screenshot generation for the screenshot worker. It manages pooled Playwright browser instances across Chromium, Firefox, and WebKit, applying navigation, authentication, viewport, wait, and full-page capture options before returning screenshot data.

Methods

Method	Signature	Returns	Description
`getBrowser`	`getBrowser(type: BrowserType)`	`Promise<Browser>`	Get or create a browser instance
`takeScreenshot`	`takeScreenshot(options: ScreenshotOptions)`	`Promise<Buffer>`	Take a screenshot of a URL
`takeResponsiveScreenshots`	`takeResponsiveScreenshots(url: string, viewports: Array<{ name: string; width: number; height: number }>)`	`Promise<Array<{ name: string; screenshot: Buffer }>>`	Take screenshots at multiple viewport sizes
`takeAuthenticatedScreenshot`	`takeAuthenticatedScreenshot(options: AuthenticatedScreenshotOptions)`	`Promise<Buffer>`	Take screenshot

Method	Signature	Returns	Description
			after form-based login Generic implementation that works with any website
`getPageHtml`	`getPageHtml(url: string, browserType: BrowserType)`	`Promise<string>`	Get HTML content of a page (for AI analysis)
`onModuleDestroy`	`onModuleDestroy()`	`unknown`	Cleanup on module destroy

Where it refuses work

- `PlaywrightService` stops the work with an early return when `browser && browser.isConnected()`.

When something fails

- `PlaywrightService` handles failure in 4 places: it lets it reach the caller in 3, and logs it and continues in 1.

Diagram

```
sequenceDiagram
  participant Caller as Screenshot Worker
  participant Service as PlaywrightService
  participant Pool as Browser Pool
  participant Browser as Playwright Browser
  participant Page as Browser Page
  participant Target as Target URL

  Caller->>Service: captureScreenshot(options)
  Service->>Pool: acquire(browserType)
  Pool-->>Service: browser instance
  Service->>Browser: newPage(viewport)
  Browser-->>Service: page
  Service->>Page: set authentication (optional)
  Service->>Page: goto(url)
  Page->>Target: HTTP request
  Target-->>Page: rendered document
  Service->>Page: waitForSelector / waitForTimeout
  Service->>Page: screenshot(fullPage)
  Page-->>Service: image buffer
  Service->>Pool: release(browser)
  Service-->>Caller: screenshot buffer
```

Usage

```
import { Injectable } from '@nestjs/common';
import { PlaywrightService } from './playwright/playwright.service';

@Injectable()
export class ScreenshotJobService {
  constructor(private readonly playwrightService: PlaywrightService) {}

  async generatePreview(): Promise<Buffer> {
    return this.playwrightService.captureScreenshot({
      url: 'https://example.com/dashboard',
      browser: 'chromium',
      viewport: {
        width: 1440,
        height: 900,
      },
      fullPage: true,
      waitForSelector: '[data-page-ready="true"]',
      waitForTimeout: 500,
      authentication: {
        username: process.env.PREVIEW_USERNAME!,
        password: process.env.PREVIEW_PASSWORD!,
      },
    });
  }
}
```

AI Coding Instructions

- Reuse the service's browser-pooling flow; do not launch a new Playwright browser for every screenshot request.
- Always release browser resources in `finally` blocks when adding or modifying page/browser lifecycle logic.
- Validate URLs, browser types, viewport dimensions, and timeout values before passing user-controlled input to Playwright.
- Prefer `waitForSelector` for deterministic page readiness; use fixed timeouts only when no reliable selector is available.
- Keep screenshot options compatible across Chromium, Firefox, and WebKit, and test browser-specific behavior when adding new options.

Referenced By

- `DiscoveryService` (DEPENDS_ON)
- `PdfService` (DEPENDS_ON)
- `PlaywrightModule` (MODULE_PROVIDES)
- `PlaywrightModule` (MODULE_EXPORTS)

- `BatchScreenshotController` (DEPENDS_ON)
- `ScreenshotService` (DEPENDS_ON)
- `BatchScreenshotService` (DEPENDS_ON)
- `FlowReplayService` (DEPENDS_ON)