PageProfilerService

September 4, 2026

PageProfilerService

Kind: Service

Source: `atloria-monorepo/apps/screenshot-worker/src/app/discovery/page-profiler.service.ts`

Profiles a page's interactive elements using a single page.evaluate() call.
ARIA-first selectors with framework-specific fallbacks.

`PageProfilerService` inspects the current browser page and returns a `RawPageProfile` describing interactive elements such as buttons, links, inputs, and controls. It performs the discovery in a single `page.evaluate()` execution to minimize browser round trips, preferring ARIA-based selectors while applying framework-specific fallbacks when needed.

Methods

Method	Signature	Returns
`profilePage`	`profilePage(page: Page)`	`Promise<RawPageProfile>`

Where it refuses work

- `PageProfilerService` stops the work with an early return when `rect.width < 10 || rect.height < 10`, in 2 places.
- `PageProfilerService` stops the work with an early return when `rect.width < 50 || rect.height < 50`, in 2 places.
- `PageProfilerService` stops the work with an early return when `!label || label.length > 50 || foundTabLabels.has(label.toLowerCase())`.
- `PageProfilerService` stops the work with an early return when `!text || text.length > 60`.
- `PageProfilerService` stops the work with an early return when `foundButtonTexts.has(text.toLowerCase())`.

- `PageProfilerService` stops the work with an early return when `modal.matches(excl) || modal.closest(excl)`.

When something fails

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

Diagram

```
sequenceDiagram
  participant Worker as Screenshot Worker
  participant Profiler as PageProfilerService
  participant Page as Playwright Page
  participant DOM as Browser DOM

  Worker->>Profiler: profilePage()
  Profiler->>Page: page.evaluate(profile interactive elements)
  Page->>DOM: Query ARIA roles, labels, and fallback attributes
  DOM-->>Page: Element metadata and selector candidates
  Page-->>Profiler: Raw page profile data
  Profiler-->>Worker: RawPageProfile
```

Usage

```
import { Injectable } from '@nestjs/common';
import { PageProfilerService } from './page-profiler.service';

@Injectable()
export class DiscoveryService {
  constructor(
    private readonly pageProfilerService: PageProfilerService,
  ) {}

  async discoverInteractiveElements() {
    const profile = await this.pageProfilerService.profilePage();

    console.log(`Found ${profile.elements.length} interactive elements`);

    return profile;
  }
}
```

AI Coding Instructions

- Keep page inspection work inside the existing single `page.evaluate()` call; avoid adding per-element Playwright calls that increase browser round trips.
- Prefer accessible metadata such as ARIA roles, accessible names, labels, and native element semantics when generating selector candidates.
- Treat framework-specific selectors as fallbacks only; do not make application implementation details the primary discovery strategy.
- Preserve the `RawPageProfile` contract when adding element metadata so downstream discovery, screenshot, and automation flows remain compatible.
- Handle missing labels, dynamically rendered controls, and non-standard interactive elements defensively without failing the entire page profile.

Referenced By

- `DiscoveryModule` (MODULE_PROVIDES)
- `DiscoveryModule` (MODULE_EXPORTS)
- `DiscoveryService` (DEPENDS_ON)