PageSettleService

September 4, 2026

PageSettleService

Kind: Service

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

Service for waiting until a page has fully settled before capture.

Extracted verbatim from BatchScreenshotService so batch capture, flow-replay (WS-A) and clip capture (WS-B) share one framework-agnostic settle strategy.

`PageSettleService` centralizes the logic for determining when a browser page is ready for screenshot capture. It combines application readiness checks, long-polling detection, and DOM stability waits so batch screenshots, flow replay, and clip capture use the same framework-agnostic settle strategy.

Methods

Method	Signature	Returns	Description
`detectLongPollingApp`	`detectLongPollingApp(page: Page)`	`Promise<boolean>`	Detect if the app uses persistent connections (longpolling, WebSockets, SSE) that would prevent networkidle from ever resolving.
`waitForAngularAppReady`	`waitForAngularAppReady(page: Page, timeoutMs: number)`	`Promise<void>`	GENERIC: Wait for page content to be fully

Method	Signature	Returns	Description
			ready Framework-agnostic approach that works for ANY web application: 1.
`waitForPageReady`	`waitForPageReady(page: Page, timeoutMs: number)`	`Promise<void>`	Smart page readiness check — waits for loading indicators to disappear and meaningful content to appear.
`waitForDomStability`	`waitForDomStability(page: Page, checkIntervalMs: number, maxChecks: number)`	`Promise<void>`	Wait for DOM to stabilize (stop changing) Takes snapshots of the DOM and waits until consecutive snapshots match

Where it refuses work

- `PageSettleService` stops the work with an early return when `!body`, in 2 places.
- `PageSettleService` stops the work with an early return when `(window as any).odoo || document.querySelector('.o_web_client')`.

When something fails

- `PageSettleService` handles failure in 5 places: it logs it and continues in 2, discards it silently in 2, and turns it into a return value in 1. A failure discarded silently leaves no trace for whoever debugs this later.

Diagram

```
sequenceDiagram
    participant Caller as Screenshot/Replay Service
    participant Settle as PageSettleService
    participant Page as Browser Page
    participant App as Web Application

    Caller->>Settle: waitForPageReady()
    Settle->>Page: detectLongPollingApp()

    alt Long-polling application detected
        Settle->>App: waitForAngularAppReady()
        App-->>Settle: Application ready
    else Standard application
        Settle->>Page: Wait for page readiness/network completion
        Page-->>Settle: Page ready
    end

    Settle->>Page: waitForDomStability()
    Page-->>Settle: DOM stable
    Settle-->>Caller: Ready for capture
```

Usage

```
import { PageSettleService } from './services/page-settle.service';

@Injectable()
export class ScreenshotCaptureService {
  constructor(
    private readonly pageSettleService: PageSettleService,
  ) {}

  async capture(page: Page): Promise<Buffer> {
    // Configure or provide the target page using the service's expected API.
    await this.pageSettleService.waitForPageReady();

    return page.screenshot({
      fullPage: true,
      type: 'png',
    });
  }
}
```

AI Coding Instructions

- Call `waitForPageReady()` immediately before capture operations rather than duplicating readiness logic in screenshot, replay, or clip services.
- Preserve the framework-agnostic behavior: Angular-specific checks should remain isolated in `waitForAngularAppReady()`.
- Account for long-polling applications; do not rely exclusively on network-idle signals when adding new settle conditions.
- Keep DOM mutation and stability timing logic inside `waitForDomStability()` so all capture workflows receive consistent behavior.
- When extending the service, prefer bounded timeouts and graceful fallbacks to avoid blocking screenshot workers indefinitely.

Referenced By

- `ScreenshotModule` (MODULE_PROVIDES)
- `BatchScreenshotService` (DEPENDS_ON)
- `FlowReplayService` (DEPENDS_ON)