PdfService

September 4, 2026

PdfService

Kind: Service

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

Renders a fully-composed HTML document to PDF bytes using headless Chromium.

This is the ONE place in the platform that owns Chromium PDF rendering — the screenshot-worker pod is the only image that ships a browser. Other services (e.g. the API's doc-version export) compose HTML and POST it here over HTTP rather than launching Chromium in-process.

`PdfService` renders fully composed HTML documents into PDF bytes using headless Chromium in the screenshot-worker. It is the platform’s single browser-rendering boundary: upstream services prepare HTML and send it to this worker over HTTP instead of bundling or launching Chromium themselves. The service is responsible for configuring the page content, waiting for rendering readiness, and returning the generated PDF payload.

Methods

Method	Signature	Returns
`renderPdf`	`renderPdf(html: string, options: RenderPdfOptions)`	`Promise<Buffer>`

Dependencies

- `PlaywrightService`

When something fails

- `PdfService` handles failure in 1 place: it lets it reach the caller in all 1.

Diagram

```
sequenceDiagram
  participant API as Calling Service
  participant Worker as Screenshot Worker
  participant PdfService
  participant Chromium as Headless Chromium

  API->>Worker: POST HTML document and PDF options
  Worker->>PdfService: renderPdf(html, options)
  PdfService->>Chromium: Create page and set HTML content
  Chromium-->>PdfService: Document rendered
  PdfService->>Chromium: Generate PDF
  Chromium-->>PdfService: PDF bytes
  PdfService-->>Worker: PDF bytes
  Worker-->>API: PDF response
```

Usage

```
import { Injectable } from '@nestjs/common';
import { PdfService } from './pdf/pdf.service';

@Injectable()
export class ExportService {
  constructor(private readonly pdfService: PdfService) {}

  async exportInvoice(): Promise<Buffer> {
    const html = `
      <!doctype html>
      <html>
        <head>
          <style>
            body { font-family: Arial, sans-serif; padding: 32px; }
            h1 { color: #1f2937; }
          </style>
        </head>
        <body>
          <h1>Invoice #INV-1001</h1>
          <p>Generated by the document export service.</p>
        </body>
      </html>
    `;

    return this.pdfService.renderPdf(html, {
      format: 'A4',
      printBackground: true,
      margin: {
        top: '16mm',
        right: '16mm',
```

```
          bottom: '16mm',
          left: '16mm',
        },
      });
    }
  }
```

AI Coding Instructions

- Keep Chromium and PDF-generation logic inside `PdfService`; other services should send composed HTML to the screenshot-worker rather than launching a browser locally.
- Pass complete, self-contained HTML whenever possible, including required styles and assets, so rendering is deterministic in the worker environment.
- Preserve PDF options such as page format, margins, and `printBackground` when adding export flows; visual output often depends on them.
- Ensure callers handle returned PDF bytes as binary data (`Buffer`/stream) and set `Content-Type: application/pdf` when returning downloads.
- Avoid adding browser-specific rendering behavior to API services; integrate through the screenshot-worker HTTP boundary instead.

Relationships

- DEPENDS_ON → `PlaywrightService`

Referenced By

- `PdfController` (DEPENDS_ON)
- `PdfModule` (MODULE_PROVIDES)
- `PdfModule` (MODULE_EXPORTS)