ScreenshotService

September 4, 2026

ScreenshotService

Kind: Service

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

Screenshot orchestration service

Coordinates Playwright and Storage services to:

1. Take screenshots
2. Upload to Azure Blob Storage
3. Return public URLs

`ScreenshotService` orchestrates browser-based screenshot capture and asset storage for the screenshot worker. It coordinates Playwright page rendering with Azure Blob Storage uploads, returning public URLs for standard, responsive, authenticated, and smart authenticated captures.

Methods

Method	Signature	Returns	Description
`takeAndUpload`	`takeAndUpload(request: TakeScreenshotRequest)`	`Promise<ScreenshotResult>`	Take a screenshot and upload it with intelligent metadata tracking
`takeResponsiveScreenshots`	`takeResponsiveScreenshots(url: string, projectId: string, organizationId: string, capturedBy: string)`	`Promise<ScreenshotResult[]>`	Take responsive screenshots at multiple viewport sizes
`takeAuthenticatedScreenshot`	`takeAuthenticatedScreenshot(request: TakeAuthenticatedScreenshotRequest)`	`Promise<ScreenshotResult>`	Take authenticated screenshot (login first, then capture)
`takeSmartAuthenticatedScreenshot`	`takeSmartAuthenticatedScreenshot(_request: TakeSmartAuthenticatedScreenshotRequest)`	`Promise<ScreenshotResult>`	Smart authenticated screenshot - DEPRECATED Smart detection is no longer available.

Dependencies

- `PlaywrightService`
- `StorageService`
- `ScreenshotMetadataService`

When something fails

- `ScreenshotService` handles failure in 3 places: it lets it reach the caller in 2, and logs it and continues in 1.

Diagram

```
sequenceDiagram
    participant Caller
    participant ScreenshotService
    participant PlaywrightService
    participant StorageService
    participant AzureBlobStorage

    Caller->>ScreenshotService: takeAndUpload(options)
    ScreenshotService->>PlaywrightService: Capture page screenshot
    PlaywrightService-->>ScreenshotService: Screenshot buffer
    ScreenshotService->>StorageService: Upload buffer and metadata
    StorageService->>AzureBlobStorage: Store screenshot blob
    AzureBlobStorage-->>StorageService: Public blob URL
    StorageService-->>ScreenshotService: Uploaded URL
    ScreenshotService-->>Caller: ScreenshotResult

    Note over Caller,ScreenshotService: Responsive and authenticated methods follow the same flow<br/>with viewport, session,
```

Usage

```
import { Injectable } from '@nestjs/common';
import { ScreenshotService } from './screenshot/screenshot.service';

@Injectable()
export class ScreenshotJobService {
  constructor(private readonly screenshotService: ScreenshotService) {}

  async captureLandingPage() {
    const result = await this.screenshotService.takeAndUpload({
      url: 'https://example.com',
      filename: 'example-homepage.png',
    });

    return {
      screenshotUrl: result.url,
    };
  }

  async captureResponsiveLandingPage() {
    const screenshots =
      await this.screenshotService.takeResponsiveScreenshots({
        url: 'https://example.com',
        filenamePrefix: 'example-homepage',
      });

    return screenshots.map((screenshot) => screenshot.url);
  }
}
```

AI Coding Instructions

- Keep screenshot capture logic in the Playwright integration and storage concerns in the Storage service; `ScreenshotService` should remain the orchestration layer.
- Use `takeResponsiveScreenshots()` when output is required for multiple viewport sizes rather than manually duplicating capture-and-upload calls.
- Use authenticated screenshot methods only when the target requires an authenticated browser session; ensure credentials, cookies, and session data are not included in returned results or logs.
- Preserve the `ScreenshotResult` return contract so callers consistently receive uploaded screenshot metadata and public URLs.
- Handle navigation, rendering, upload, and authentication failures with actionable errors because this service depends on external browser and Azure storage operations.

Relationships

- DEPENDS_ON → `PlaywrightService`
- DEPENDS_ON → `StorageService`
- DEPENDS_ON → `ScreenshotMetadataService`