ScreenshotService

September 4, 2026

ScreenshotService

Kind: Service

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

`ScreenshotService` manages screenshot capture workflows for the API, including creating individual captures, listing and retrieving screenshots, and deleting stored records. It also supports bulk capture operations and exposes job-status methods so callers can monitor asynchronous screenshot processing.

Methods

Method	Signature	Returns	Description
`captureScreenshot`	`captureScreenshot(dto: CaptureScreenshotDto, user: JwtPayload)`	`unknown`	Capture a single screenshot
`listScreenshots`	`listScreenshots(filters: ScreenshotFiltersDto, user: JwtPayload)`	`unknown`	List screenshots with filters
`getScreenshot`	`getScreenshot(id: string, user: JwtPayload)`	`unknown`	Get a single screenshot by ID
`deleteScreenshot`	`deleteScreenshot(id: string, user: JwtPayload)`	`unknown`	Delete a screenshot
`bulkCaptureScreenshots`	`bulkCaptureScreenshots(dto: BulkCaptureScreenshotsDto, user: JwtPayload)`	`unknown`	Bulk capture screenshots (creates a job)
`listJobs`	`listJobs(filters: ScreenshotJobFiltersDto, user: JwtPayload)`	`unknown`	List screenshot jobs

Method	Signature	Returns	Description
`getJob`	`getJob(id: string, user: JwtPayload)`	`unknown`	Get a screenshot job by ID

Dependencies

- `PrismaService`
- `AssetsService`

Where it refuses work

- `ScreenshotService` stops the work with `NotFoundException` when `!project` — “Project not found”, in 3 places.
- `ScreenshotService` stops the work with `NotFoundException` when `!screenshot` — “Screenshot not found”, in 2 places.
- `ScreenshotService` stops the work with `ForbiddenException` when `screenshot.organizationId !== user.organizationId` — “You do not have access to this screenshot”, in 2 places.
- `ScreenshotService` stops the work with `ForbiddenException` when `project.organizationId !== user.organizationId` — “You do not have access to this project”.
- `ScreenshotService` stops the work with `ForbiddenException` when `user.role !== UserRole.ADMIN && user.role !== UserRole.MAINTAINER` — “Only admins and maintainers can delete screenshots”.
- `ScreenshotService` stops the work with `NotFoundException` when `!job` — “Screenshot job not found”.

Diagram

```
sequenceDiagram
  participant Client
  participant Controller as Screenshot Controller
  participant Service as ScreenshotService
  participant Queue as Job Queue
  participant Storage as Screenshot Storage

  Client->>Controller: Request captureScreenshot()
  Controller->>Service: captureScreenshot(payload)
  Service->>Queue: Enqueue screenshot capture job
```

```
  Queue-->>Service: Job created
  Service-->>Controller: Job metadata
  Controller-->>Client: Capture job response

  Queue->>Storage: Generate and persist screenshot
  Client->>Controller: getJob(jobId)
  Controller->>Service: getJob(jobId)
  Service->>Queue: Retrieve job status
  Queue-->>Service: Status and result
  Service-->>Controller: Job details
  Controller-->>Client: Capture status
```

Usage

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

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

  async captureReportPreview(url: string) {
    const job = await this.screenshotService.captureScreenshot({
      url,
      viewport: {
        width: 1440,
        height: 900,
      },
    });

    return job;
  }

  async getCaptureStatus(jobId: string) {
    return this.screenshotService.getJob(jobId);
  }

  async listRecentScreenshots() {
    return this.screenshotService.listScreenshots();
  }

  async removeScreenshot(screenshotId: string) {
    await this.screenshotService.deleteScreenshot(screenshotId);
  }
}
```

AI Coding Instructions

- Keep screenshot capture operations asynchronous; use `getJob()` or `listJobs()` to inspect queued or completed bulk and individual capture work.
- Validate capture inputs, especially target URLs and viewport options, before calling `captureScreenshot()` or `bulkCaptureScreenshots()`.
- Use `listScreenshots()` and `getScreenshot()` for persisted screenshot data; do not assume that a completed job always includes the full stored screenshot payload.
- When adding new capture options, update both single and bulk capture paths to keep their request contracts consistent.
- Ensure deletion flows use `deleteScreenshot()` only after confirming the screenshot identifier and any related storage or retention requirements.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `AssetsService`