StateResolverService

September 4, 2026

StateResolverService

Kind: Service

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

`StateResolverService` is a NestJS service responsible for resolving the state required before a screenshot job can proceed. It centralizes state lookup and normalization, returning a `StateResolutionResult` that downstream screenshot-processing components can consume.

Methods

Method	Signature	Returns	Description
`resolveState`	`resolveState(page: Page, config: StateResolutionConfigDto)`	`Promise<StateResolutionResult>`	Resolve a record in a specific workflow state.

Where it refuses work

- `StateResolverService` stops the work with an early return when `!strategy`.
- `StateResolverService` stops the work with an early return when `!match`.
- `StateResolverService` stops the work with an early return when `hasCells > 0`.
- `StateResolverService` stops the work with an early return when `count > 0`.
- `StateResolverService` stops the work with an early return when `trimmed.toLowerCase().includes(targetLower)`.

When something fails

- `StateResolverService` handles failure in 5 places: it discards it silently in 3, logs it and continues in 1, and turns it into a return value in 1.

Diagram

```
sequenceDiagram
    participant Worker as Screenshot Worker
    participant Resolver as StateResolverService
    participant Sources as State Sources
    participant Result as StateResolutionResult

    Worker->>Resolver: resolveState()
    Resolver->>Sources: Load required state
    Sources-->>Resolver: Raw state data
    Resolver->>Resolver: Validate and normalize state
    Resolver-->>Result: Return resolved state
    Result-->>Worker: Continue screenshot workflow
```

Usage

```
import { Injectable } from '@nestjs/common';
import { StateResolverService } from './services/state-resolver.service';

@Injectable()
export class ScreenshotProcessor {
  constructor(
    private readonly stateResolverService: StateResolverService,
  ) {}

  async processScreenshot(): Promise<void> {
    const state = await this.stateResolverService.resolveState();

    // Use the resolved state to configure and execute screenshot work.
    console.log('Resolved screenshot state:', state);
  }
}
```

AI Coding Instructions

- Keep state-resolution logic inside `StateResolverService`; callers should consume the returned `StateResolutionResult` rather than duplicate lookup or normalization logic.
- Treat `resolveState()` as asynchronous and always `await` its result before starting dependent screenshot operations.

- Preserve the `StateResolutionResult` contract when extending the service, since downstream worker components may rely on its shape.
- Handle unavailable or invalid state explicitly, using NestJS-compatible error handling where resolution cannot produce a valid result.
- Inject this service through NestJS dependency injection instead of constructing it manually.

Referenced By

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