FreshnessController

September 5, 2026

FreshnessController

Kind: Controller

Source: `atloria-monorepo/apps/api/src/technical-docs/freshness.controller.ts`

Owner "Docs freshness cockpit" — which published pages went stale when the project's
code changed (from technical_snapshots.staleInfo, written on every push by TechnicalDocsService.checkStaleness). Org-scoped + member-gated, mirroring ManualsInsightsController's authed surfaces. The public per-page reader badge lives separately in ManualsInsightsController.freshness.

`FreshnessController` powers the owner-facing Docs Freshness Cockpit, showing which published technical documentation pages became stale after project code changed. It exposes organization-scoped, member-gated freshness data derived from `technical_snapshots.staleInfo`, which is updated by `TechnicalDocsService.checkStaleness` on each push.

Diagram

```
graph LR
  Push[Repository push] --> Staleness[TechnicalDocsService.checkStaleness]
  Staleness --> Snapshot[(technical_snapshots.staleInfo)]
  Owner[Organization member] --> Guard[Authentication and org membership guards]
  Guard --> Controller[FreshnessController]
  Controller --> Snapshot
  Controller --> Cockpit[Docs Freshness Cockpit]
  Cockpit --> Pages[Published pages marked stale]
```

Usage

```
// Example owner-dashboard client request.
// Use the route registered for FreshnessController in the API module.
```

```
const response = await fetch(
  `${process.env.API_URL}/technical-docs/freshness`,
  {
    headers: {
      Authorization: `Bearer ${accessToken}`,
      'x-organization-id': organizationId,
    },
  },
);

if (!response.ok) {
  throw new Error('Unable to load documentation freshness data');
}

const freshness = await response.json();

// Render stale published pages in the Docs Freshness Cockpit.
for (const page of freshness.pages) {
  if (page.staleInfo?.isStale) {
    console.log(`${page.title} needs review:`, page.staleInfo);
  }
}
```

AI Coding Instructions

- Preserve the controller's organization-scoping and member-gated access pattern; do not expose cockpit-level freshness data through public routes.
- Treat `technical_snapshots.staleInfo` as the source of truth for staleness status, populated by `TechnicalDocsService.checkStaleness`.
- Keep freshness responses focused on published-page maintenance workflows, such as identifying pages needing review after code changes.
- Do not move or duplicate the public per-page reader freshness badge here; that behavior belongs in `ManualsInsightsController.freshness`.
- When adding filters or response fields, ensure they respect organization boundaries and do not leak snapshot data across organizations.

Relationships

- MODULE_DECLARES → `stale`
- MODULE_DECLARES → `reshoot`
- DEPENDS_ON → `FreshnessService`

Referenced By

- `TechnicalDocsModule` (MODULE_DECLARES)