SnippetsService

September 4, 2026

SnippetsService

Kind: Service

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

B4 — per-project reusable content (snippets).

- CRUD over the Snippet table (org tenancy enforced by the controller's ResourceOrgGuard on projectId + projectId filters on every query here).
- The serve-time expansion entry points every seam calls (expandMarkdown / getSnippetMap + expandIncludesWithMap for corpus builds).
- Change fan-out: the git substrate is nudged directly (outbox + commit queue — docs-repo does not depend on this module, so no cycle); RAG/search/export invalidation is done by LISTENER hooks registered by the modules that own those caches (they already import SnippetsModule for expansion, so registration flows in their dependency direction).

`SnippetsService` manages reusable, per-project content snippets and provides CRUD operations scoped by `projectId`. It also expands snippet includes in Markdown for serve-time rendering and corpus builds, while notifying the Git substrate and registered cache/search/export listeners when snippets change.

Methods

Method	Signature	Returns	Description
`onSnippetChange`	`onSnippetChange(listener: (event: SnippetChangeEvent) => void)`	`void`	Register an invalidation hook (called after create/update/delete).
`list`	`list(projectId: string)`	`unknown`	
`get`	`get(projectId: string, id: string)`	`unknown`	
`create`	`create(projectId: string, organizationId: string, dto: CreateSnippetDto, createdById: string)`	`unknown`	
`update`	`update(projectId: string, id: string, dto: UpdateSnippetDto)`	`unknown`	
`remove`	`remove(projectId: string, id: string)`	`unknown`	

Method	Signature	Returns	Description
`references`	`references(projectId: string, name: string)`	`unknown`	Pages currently referencing a snippet (manager UI "used by" + safety check).
`getSnippetMapIfNeeded`	`getSnippetMapIfNeeded(projectId: string, contents: Array<string`	null	undefined>)`
`getSnippetMap`	`getSnippetMap(projectId: string)`	`Promise<Map<string, string>>`	Load a project's snippet map once (for corpus builders expanding many pages).
`expandMarkdown`	`expandMarkdown(projectId: string, markdown: string)`	`Promise<string>`	Expand a single markdown document (cheap no-op when it has no includes).
`expandMarkdownDetailed`	`expandMarkdownDetailed(projectId: string, markdown: string)`	`Promise<ExpandIncludesResult>`	Detailed expansion (missing/cycle diagnostics) for previews/authoring UI.

Dependencies

- `PrismaService`
- `OutboxService`
- `DocsRepoQueue`

Where it refuses work

- `SnippetsService` stops the work with `NotFoundException` when `!snippet`.
- `SnippetsService` stops the work with `ConflictException` when `existing`.
- `SnippetsService` stops the work with `ConflictException` when `clash`.
- `SnippetsService` stops the work with `BadRequestException` when `!SnippetsService.NAME_RE.test(name)` — “Snippet name must be a slug: lowercase letters/digits, then letters/digits/hyphen/undersc…”.
- `SnippetsService` stops the work with an early return when `!contents.some((c) => hasIncludeDirective(c ?? ''))`.
- `SnippetsService` stops the work with an early return when `!hasIncludeDirective(markdown)`.

When something fails

- `SnippetsService` handles failure in 2 places: it logs it and continues in all 2.

Diagram

```
sequenceDiagram
  participant Client
```

```
  participant Controller
  participant Service as SnippetsService
  participant DB as Snippet Table
  participant Git as Git Outbox / Commit Queue
  participant Listeners as Search/RAG/Export Listeners

  Client->>Controller: Create or update snippet
  Controller->>Controller: ResourceOrgGuard(projectId)
  Controller->>Service: create/update(projectId, payload)
  Service->>DB: Query/write scoped by projectId
  DB-->>Service: Snippet result
  Service->>Git: Queue repository change
  Service->>Listeners: Emit snippet-change hook
  Service-->>Controller: Snippet result
  Controller-->>Client: Response

  Client->>Service: expandMarkdown(projectId, markdown)
  Service->>DB: Load project snippet map
  DB-->>Service: Snippet content
  Service-->>Client: Expanded Markdown
```

Usage

```
import { SnippetsService } from './snippets.service';

export class DocumentRenderer {
  constructor(private readonly snippetsService: SnippetsService) {}

  async renderProjectDocument(projectId: string, markdown: string) {
    // Expands snippet include directives using snippets belonging only
    // to the requested project.
    return this.snippetsService.expandMarkdown(projectId, markdown);
  }

  async createReusableFooter(projectId: string) {
    return this.snippetsService.create(projectId, {
      name: 'standard-footer',
      content: '---\nContact the documentation team for support.',
    });
  }
}
```

AI Coding Instructions

- Always scope snippet reads, writes, and reference checks by `projectId`; organization access is enforced by the controller's `ResourceOrgGuard`, but service queries must still retain project filtering.
- Use `expandMarkdown()` for request-time Markdown rendering and `getSnippetMap()` with the include-expansion utilities for bulk or corpus-generation workflows.
- Call the existing snippet-change fan-out path after mutations so Git outbox/commit processing and registered search, RAG, and export invalidation listeners remain synchronized.
- Do not introduce dependencies from the docs-repository layer back into this module; Git changes are intentionally pushed through the outbox and commit queue to avoid dependency cycles.
- Preserve snippet expansion behavior when changing include syntax or caching logic, including handling missing snippets and avoiding unnecessary map loading via `getSnippetMapIfNeeded()`.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `OutboxService`
- DEPENDS_ON → `DocsRepoQueue`

Referenced By

- `DocVersionExportService` (DEPENDS_ON)
- `DocumentIndexingService` (DEPENDS_ON)
- `ProjectService` (DEPENDS_ON)
- `SnippetsController` (DEPENDS_ON)
- `SnippetsModule` (MODULE_PROVIDES)
- `SnippetsModule` (MODULE_EXPORTS)
- `TechnicalDocsMaterializerService` (DEPENDS_ON)