PageRegenService

September 4, 2026

PageRegenService

Kind: Service

Source: `atloria-monorepo/apps/api/src/docs-pr/page-regen.service.ts`

Regenerates ONE doc page's markdown straight from the code entity graph, using the
exact same deterministic renderer the whole-project materializer uses — scoped to a single document. This is Rung 1 of the freshness ladder (scoped redraft).

Grounding is preserved: `renderEntityPage` renders only from the entity + its relationships + its stored AI enrichment, so every line traces to a code symbol.
No LLM, no whole-project rebuild.

`PageRegenService` regenerates a single documentation page directly from the code entity graph using the same deterministic renderer as the whole-project materializer. It performs the first “freshness ladder” step—a scoped redraft—without invoking an LLM or rebuilding documentation for the entire project. Generated content is grounded in the entity, its relationships, and stored AI enrichment.

Methods

Method	Signature	Returns
`regenerateFromCode`	`regenerateFromCode(documentId: string)`	`Promise<RegeneratedPage

Dependencies

- `PrismaService`

Where it refuses work

- `PageRegenService` stops the work with an early return when `!doc?.projectId || !doc.kgEntityId`.
- `PageRegenService` stops the work with an early return when `!snapshot`.
- `PageRegenService` stops the work with an early return when `!enrDoc?.content`.

Diagram

```
sequenceDiagram
  participant Caller as API / Docs Workflow
  participant Service as PageRegenService
  participant Graph as Code Entity Graph
  participant Renderer as renderEntityPage

  Caller->>Service: regenerateFromCode(entityId)
  Service->>Graph: Load entity and relationships
  Graph-->>Service: Entity graph data + AI enrichment
  Service->>Renderer: Render scoped entity page
  Renderer-->>Service: Deterministic markdown
  Service-->>Caller: RegeneratedPage or null
```

Usage

```
import { PageRegenService } from './page-regen.service';

@Injectable()
export class DocsRefreshService {
  constructor(
    private readonly pageRegenService: PageRegenService,
  ) {}

  async refreshPage(entityId: string) {
    const regenerated = await this.pageRegenService.regenerateFromCode(entityId);

    if (!regenerated) {
      throw new NotFoundException(
        `No documentation page could be regenerated for entity ${entityId}`,
      );
    }

    return {
      entityId,
      markdown: regenerated.markdown,
    };
  }
}
```

AI Coding Instructions

- Use `regenerateFromCode()` for targeted page refreshes; do not trigger a whole-project materialization for a single stale document.
- Preserve deterministic rendering by sourcing page content only from the entity graph, relationships, and persisted AI enrichment.
- Handle the `null` return value explicitly; it indicates that the requested entity/page could not be regenerated.
- Do not add LLM calls inside this service’s regeneration path, as scoped redrafts must remain grounded and reproducible.
- Keep renderer changes compatible with the shared `renderEntityPage` behavior used by full-project documentation generation.

Relationships

- DEPENDS_ON → `PrismaService`

Referenced By

- `DocsPrModule` (MODULE_PROVIDES)
- `DocsPrModule` (MODULE_EXPORTS)
- `DocsPrService` (DEPENDS_ON)