DocsPrService

September 4, 2026

DocsPrService

Kind: Service

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

Orchestrates one docs-fix PR for a project: resolve the stale pages (from the shipped staleness signal) → regenerate each from code (Rung 1) → open or update a single deduped PR in the customer's repo. Nothing merges automatically; the PR is the review
surface. Every path is guarded so a project with the feature OFF, no repo, nothing stale, or an un-consented App simply no-ops — the repo is never touched on failure.

`DocsPrService` orchestrates documentation-fix pull requests for a project. It identifies stale docs pages from the shipped staleness signal, regenerates eligible pages from code, and creates or updates one deduplicated PR in the customer repository. All entry points are guarded so disabled features, missing repositories, empty stale-page sets, or missing app consent result in a safe no-op.

Methods

Method	Signature	Returns	Description
`onModuleInit`	`onModuleInit()`	`void`	Wire the webhook's signals through the durable queue (survives restarts, retries).
`runForProject`	`runForProject(projectId: string, opts: { actorUserId?: string })`	`Promise<DocsPrRunResult>`	

Method	Signature	Returns	Description
`handlePrClosed`	`handlePrClosed(repoUrl: string, prNumber: number, merged: boolean)`	`Promise<void>`	A docs PR was merged or closed on the remote → reflect it in our tracking, and on merge clear the project's staleness (the docs are now in sync).

Dependencies

- `PrismaService`
- `PageRegenService`
- `GitWriteAdapter`
- `DocsPrTrigger`
- `DocsPrQueue`
- `AgentRegenService`
- `GitCloneService`
- `GithubAppService`
- `PreviewService`
- `ChangelogDraftsService` *(optional)*

Where it refuses work

- `DocsPrService` stops the work with an early return when `!config?.enabled`.
- `DocsPrService` stops the work with an early return when `config.provider !== 'github'`.
- `DocsPrService` stops the work with an early return when `!project?.repoUrl`.
- `DocsPrService` stops the work with an early return when `stale.docIds.length === 0`.
- `DocsPrService` stops the work with an early return when `regenerated.length === 0`.
- `DocsPrService` stops the work with an early return when `allChanged.length === 0`.

When something fails

- `DocsPrService` handles failure in 2 places: it logs it and continues in 1, and lets it reach the caller in 1.

Diagram

```
sequenceDiagram
  participant Caller as Scheduler / API
  participant Service as DocsPrService
  participant Config as Project Configuration
  participant Signal as Staleness Signal
  participant Generator as Docs Generator
  participant Repo as Customer Repository
  participant GitHub as GitHub App / PR API

  Caller->>Service: runForProject(projectId)
  Service->>Config: Check docs PR feature and repo access

  alt Feature disabled, no repo, or no consent
    Config-->>Service: Not eligible
    Service-->>Caller: No-op result
  else Project eligible
    Service->>Signal: Resolve stale pages

    alt No stale pages
      Signal-->>Service: Empty set
      Service-->>Caller: No-op result
    else Stale pages found
      Signal-->>Service: Stale page list
      loop Each stale page
        Service->>Generator: Regenerate page from code (Rung 1)
        Generator-->>Service: Generated content
      end
      Service->>Repo: Prepare deduplicated docs-fix branch
      Service->>GitHub: Create or update single PR
      GitHub-->>Service: PR details
      Service-->>Caller: DocsPrRunResult
    end
  end

  GitHub->>Service: handlePrClosed(pr)
  Service->>Repo: Clean up or reconcile closed PR state
```

Usage

```
import { DocsPrService } from './docs-pr.service';

@Injectable()
export class DocsPrJob {
```

```
  constructor(private readonly docsPrService: DocsPrService) {}

  async run(projectId: string) {
    const result = await this.docsPrService.runForProject(projectId);

    if (result.status === 'noop') {
      return;
    }

    console.info('Docs PR processed', {
      projectId,
      pullRequestUrl: result.pullRequestUrl,
    });
  }

  async onPullRequestClosed(payload: { projectId: string; pullRequestNumber: number }) {
    await this.docsPrService.handlePrClosed(payload);
  }
}
```

AI Coding Instructions

- Preserve the no-op-first behavior: never touch a customer repository until feature flags, repository configuration, stale pages, and GitHub App consent have all been validated.
- Keep generated documentation changes scoped to stale pages and continue using the Rung 1 code-derived regeneration path.
- Maintain the single deduplicated PR model; update an existing docs-fix PR rather than opening duplicate PRs for the same project.
- Treat PR creation and updates as review-only workflows—do not add automatic merge behavior.
- When changing PR lifecycle handling, ensure `handlePrClosed()` keeps deduplication and repository state consistent for future runs.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `PageRegenService`
- DEPENDS_ON → `GitWriteAdapter`
- DEPENDS_ON → `DocsPrTrigger`
- DEPENDS_ON → `DocsPrQueue`
- DEPENDS_ON → `AgentRegenService`

- DEPENDS_ON → `GitCloneService`
- DEPENDS_ON → `GithubAppService`
- DEPENDS_ON → `PreviewService`
- DEPENDS_ON → `ChangelogDraftsService`

Referenced By

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