PreviewService

September 4, 2026

PreviewService

Kind: Service

Source: `atloria-monorepo/apps/api/src/change-request/preview.service.ts`

`PreviewService` provides the backend capabilities required to generate and validate change-request preview access, build preview URLs, and expose preview data. It supplies both a structured manifest of changed documents and an overlay payload that preview clients can use to render unpublished or proposed content changes.

Methods

Method	Signature	Returns	Description
`mintPreviewToken`	`mintPreviewToken(crId: string, exp: number)`	`string`	
`verifyPreviewToken`	`verifyPreviewToken(crId: string, token: string`	undefined	null)`
`buildPreviewUrl`	`buildPreviewUrl(projectId: string, crId: string)`	`Promise<string`	null>`
`manifest`	`manifest(projectId: string, crId: string, token: string`	undefined, isMember: boolean)`	`Promise<{

```
crId: string;
title: string;
status: string;
changedSlugs: string[];
changed: Array<{ documentId: string; slug: string; title: string; audienceSlug: string }>;
```

}> `| Preview manifest: the CR title, its status, and the changed pages (slug + audience so the reader can route into them). | |` overlay `|` overlay(projectId: string, crId: string, token: string | undefined, isMember: boolean, resolver: PublicDocResolver, args:

OverlayArgs) | Promise<Record<string, unknown>>` | Serve one document with the preview overlaid: if the requested page's documentId is in this CR's previewFiles, return the PROPOSED content (rendered through … |

Dependencies

- `PrismaService`
- `ConfigService`

Where it refuses work

- `PreviewService` stops the work with `Error` when `!s` — “JWT_SECRET not configured — preview tokens cannot be signed”.
- `PreviewService` stops the work with `NotFoundException` when `!src` — “Preview not found”.
- `PreviewService` stops the work with `GoneException` when `TERMINAL_STATUSES.has(src.status)` — “This change request is no longer previewable (merged or closed)”.
- `PreviewService` stops the work with `ForbiddenException` when `!isMember && !this.verifyPreviewToken(crId, token)` — “Invalid or expired preview token”.
- `PreviewService` stops the work with an early return when `!token`.
- `PreviewService` stops the work with an early return when `!payload || !sig`.

When something fails

- `PreviewService` handles failure in 1 place: it turns it into a return value in all 1.

Diagram

```
sequenceDiagram
  participant Client
  participant Controller
  participant PreviewService
  participant ChangeRequestData
  participant PreviewApp

  Client->>Controller: Request preview for change request
  Controller->>PreviewService: mintPreviewToken()
  PreviewService-->>Controller: preview token
  Controller->>PreviewService: buildPreviewUrl()
  PreviewService-->>Controller: preview URL or null
```

```
  Controller-->>Client: Redirect/open preview URL

  Client->>PreviewApp: Open preview URL with token
  PreviewApp->>Controller: Request preview manifest/overlay
  Controller->>PreviewService: verifyPreviewToken()
  PreviewService-->>Controller: valid / invalid
  Controller->>PreviewService: manifest() / overlay()
  PreviewService->>ChangeRequestData: Load change request changes
  ChangeRequestData-->>PreviewService: Changed documents and content
  PreviewService-->>Controller: Preview payload
  Controller-->>PreviewApp: Manifest or overlay response
```

Usage

```
import { Controller, ForbiddenException, Get, Query } from '@nestjs/common';
import { PreviewService } from './preview.service';

@Controller('change-requests')
export class ChangeRequestPreviewController {
  constructor(private readonly previewService: PreviewService) {}

  @Get('preview-url')
  async getPreviewUrl() {
    const token = this.previewService.mintPreviewToken();
    const previewUrl = await this.previewService.buildPreviewUrl();

    if (!previewUrl) {
      return { previewUrl: null };
    }

    return {
      previewUrl: `${previewUrl}${previewUrl.includes('?') ? '&' : '?'}token=${token}`,
    };
  }

  @Get('preview-manifest')
  async getManifest(@Query('token') token: string) {
    if (!this.previewService.verifyPreviewToken()) {
      throw new ForbiddenException('Invalid preview token');
    }

    return this.previewService.manifest();
  }

  @Get('preview-overlay')
  async getOverlay(@Query('token') token: string) {
    if (!this.previewService.verifyPreviewToken()) {
      throw new ForbiddenException('Invalid preview token');
    }
```

```
    return this.previewService.overlay();
  }
}
```

AI Coding Instructions

- Use `mintPreviewToken()` only when creating preview access, and validate access with `verifyPreviewToken()` before returning preview-specific data.
- Handle `buildPreviewUrl()` returning `null`; this indicates that a preview destination is not currently available or configured.
- Use `manifest()` for preview navigation, change summaries, and document metadata; use `overlay()` when the preview client needs content-level overrides.
- Keep preview endpoints protected and avoid exposing manifest or overlay data through public, unauthenticated routes.
- Preserve the manifest contract, especially `changedSlugs` and the document metadata fields consumed by preview clients.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `configservice`

Referenced By

- `ChangeRequestController` (DEPENDS_ON)
- `ChangeRequestModule` (MODULE_PROVIDES)
- `ChangeRequestModule` (MODULE_EXPORTS)
- `DocsPrService` (DEPENDS_ON)
- `PublicProjectController` (DEPENDS_ON)