ContentVisibilityController

September 4, 2026

ContentVisibilityController

Kind: Controller

Source: `atloria-monorepo/apps/api/src/project/content-visibility.controller.ts`

Owner API for D4 content segmentation (MAINTAINER/ADMIN). Sets per-document/category
visibility rules, manages the project's ReaderIdentityConfig, and mints "Preview as…" reader tokens. Org ownership is verified in the service from the caller's JWT org. Concrete guard imports (not the ../auth barrel) to avoid the require-cycle → undefined decorator trap.

`ContentVisibilityController` exposes owner-only APIs for configuring D4 content segmentation within a project. It delegates organization ownership checks to the service using the caller’s JWT organization, manages document/category visibility rules and `ReaderIdentityConfig`, and mints scoped “Preview as…” reader tokens for testing audience-specific access.

Diagram

```
graph LR
  Client[Maintainer / Admin client] --> JWT[JWT + role guard]
  JWT --> Controller[ContentVisibilityController]
  Controller --> Service[Content visibility service]
  Service --> Ownership[Verify JWT org owns project]
  Ownership --> Rules[Document/category visibility rules]
  Ownership --> Identity[ReaderIdentityConfig]
  Ownership --> Preview[Preview-as reader token]
  Rules --> D4[D4 content delivery]
  Identity --> D4
  Preview --> D4
```

Usage

```
// Example client-side request pattern.
// Use the concrete route and DTO field names defined by ContentVisibilityController.

const response = await fetch(
  `${API_BASE_URL}/projects/${projectId}/content-visibility`,
  {
    method: 'PATCH',
    headers: {
      Authorization: `Bearer ${maintainerJwt}`,
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      documentId: 'doc_123',
      visibility: {
        audience: 'partner',
        region: 'us',
      },
    }),
  },
);

if (!response.ok) {
  throw new Error(`Unable to update visibility: ${await response.text()}`);
}

// Preview-token endpoints can then be used by maintainers/admins to verify
// how the project appears for a selected reader identity.
const updatedRule = await response.json();
console.log(updatedRule);
```

AI Coding Instructions

- Import authentication guards directly from their concrete modules; do not use the `../auth` barrel, which can introduce a require cycle and leave decorators undefined.
- Keep authorization enforcement in the service layer: derive the caller organization from the JWT and verify it owns the target project before changing rules or minting tokens.
- Restrict all controller routes to `MAINTAINER` and `ADMIN` access; preview tokens must remain scoped to the requested project and reader identity.
- Reuse the project's existing DTO validation and response conventions when adding visibility-rule or `ReaderIdentityConfig` endpoints.
- Treat "Preview as…" tokens as test/preview credentials only; do not let them bypass normal D4 visibility evaluation.

Relationships

- MODULE_DECLARES → `listRules`
- MODULE_DECLARES → `setDocumentRule`
- MODULE_DECLARES → `setCategoryRule`
- MODULE_DECLARES → `getReaderIdentity`
- MODULE_DECLARES → `updateReaderIdentity`
- MODULE_DECLARES → `mintPreviewToken`
- DEPENDS_ON → `ContentVisibilityService`

Referenced By

- `ProjectModule` (MODULE_DECLARES)