CommentService

September 4, 2026

CommentService

Kind: Service

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

Service for managing document comments with threading,

`CommentService` manages document comments, including threaded replies, updates, deletion, and resolution state. It acts as the backend orchestration layer for comment-related API operations, including mention extraction and retrieval of individual comment records.

Methods

Method	Signature	Returns	Description
`create`	`create(documentId: string, dto: CreateCommentDto, user: JwtPayload)`	`unknown`	Create a new comment on a document
`list`	`list(documentId: string, filters: CommentFiltersDto)`	`unknown`	List all comments for a document with optional filters
`update`	`update(id: string, dto: UpdateCommentDto, user: JwtPayload)`	`unknown`	Update a comment's content
`delete`	`delete(id: string, user: JwtPayload, isAdmin: unknown)`	`unknown`	Delete a comment (author only, or admin)
`resolve`	`resolve(id: string, user: JwtPayload)`	`unknown`	Resolve a comment thread
`unresolve`	`unresolve(id: string, user: JwtPayload)`	`unknown`	Unresolve a previously resolved comment thread

Method	Signature	Returns	Description
`addReply`	`addReply(commentId: string, dto: CreateReplyDto, user: JwtPayload)`	`unknown`	Add a reply to an existing comment
`getById`	`getById(id: string)`	`unknown`	Get a single comment by ID
`extractMentions`	`extractMentions(content: string)`	`string[]`	Extract

Dependencies

- `PrismaService`
- `DocumentGateway`
- `ActivityService`
- `NotificationService`

Where it refuses work

- `CommentService` stops the work with `NotFoundException` when `!comment` — “Comment not found”, in 5 places.
- `CommentService` stops the work with `NotFoundException` when `!document` — “Document not found”.
- `CommentService` stops the work with `ForbiddenException` when `comment.createdById !== user.sub` — “You can only edit your own comments”.
- `CommentService` stops the work with `ForbiddenException` when `!isAdmin && comment.createdById !== user.sub` — “You can only delete your own comments”.
- `CommentService` stops the work with `ForbiddenException` when `comment.resolved` — “Comment is already resolved”.
- `CommentService` stops the work with `ForbiddenException` when `!comment.resolved` — “Comment is not resolved”.

When something fails

- `CommentService` handles failure in 3 places: it logs it and continues in all 3.

Diagram

```
sequenceDiagram
    participant Client
```

```
    participant Controller as CommentController
    participant Service as CommentService
    participant Database

    Client->>Controller: Create or update comment request
    Controller->>Service: create() / update()
    Service->>Service: extractMentions(content)
    Service->>Database: Persist comment and mentions
    Database-->>Service: Comment record
    Service-->>Controller: Comment response
    Controller-->>Client: Updated comment data

    Client->>Controller: Add reply or resolve comment
    Controller->>Service: addReply() / resolve()
    Service->>Database: Update thread or resolution state
    Database-->>Service: Updated record
    Service-->>Controller: Updated comment thread
    Controller-->>Client: Comment response
```

Usage

```
import { CommentService } from './comment.service';

class CommentController {
  constructor(private readonly commentService: CommentService) {}

  async createComment(documentId: string, authorId: string, content: string) {
    return this.commentService.create({
      documentId,
      authorId,
      content,
    });
  }

  async replyToComment(commentId: string, authorId: string, content: string) {
    return this.commentService.addReply(commentId, {
      authorId,
      content,
    });
  }

  async resolveComment(commentId: string, userId: string) {
    return this.commentService.resolve(commentId, userId);
  }

  async getComment(commentId: string) {
    return this.commentService.getById(commentId);
  }
}
```

AI Coding Instructions

- Keep comment mutations within `CommentService`; controllers should validate requests and delegate business logic rather than directly accessing persistence.
- Preserve thread relationships when implementing `addReply()`—replies must remain associated with their parent comment and document.
- Use `extractMentions()` whenever comment content is created or updated so mention-related behavior stays consistent.
- Verify authorization and document access before allowing update, delete, resolve, unresolve, or reply operations.
- Ensure `list()` returns comments in a thread-friendly structure or ordering expected by the document collaboration UI.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `DocumentGateway`
- DEPENDS_ON → `ActivityService`
- DEPENDS_ON → `NotificationService`

Referenced By

- `CommentController` (DEPENDS_ON)
- `CommentModule` (MODULE_PROVIDES)
- `CommentModule` (MODULE_EXPORTS)