Content Generator Application Documentation (Back End)

Overview

The Content Generator application is a Spring Boot-based REST API designed to generate content using OpenAI's GPT-3.5 model. The application receives content generation requests, processes them, and returns the generated content. It interacts with a MongoDB database to store the content generation requests and responses.

Components

1. ContentController

- **Description:** This is the controller class responsible for handling HTTP requests related to content generation.
- **Key Methods:**
  - generateContent(@RequestBody ContentRequest request): Handles POST requests to generate content. It processes the incoming request, interacts with the ContentService, and returns the generated content.
- **Endpoints:**
  - POST /api/v1/content/generate: Receives a ContentRequest JSON object and returns a ContentRequest with the generated content.

2. ContentRequest (Model)

- **Description:** This is a model class representing the content generation request and its attributes.
- **Attributes:**
  - id: The unique identifier for the content request (generated by MongoDB).
  - tone: The tone of the content (e.g., formal, informal).
  - targetAudience: The intended audience for the content (e.g., children, adults).
  - format: The format of the content (e.g., blog post, email).
  - length: The length of the content (e.g., short, medium, long).
  - generatedContent: The generated content after processing the request.

3. ContentRepository (Repository)

- **Description:** Interface for interacting with the MongoDB database. It extends MongoRepository to handle CRUD operations on ContentRequest objects.
- **Methods:** Inherited from MongoRepository, which provides standard operations like saving and retrieving documents.

4. ContentService (Service)

- **Description:** The service class responsible for the business logic, including content generation and database interactions.
- **Key Methods:**
  - generateContent(ContentRequest request): Validates the incoming ContentRequest, builds a prompt, interacts with OpenAI's API, and saves the content request with the generated content.
  - validateRequest(ContentRequest request): Validates the request to ensure all required fields (tone, target audience, format, and length) are present.
  - buildPrompt(ContentRequest request): Builds the prompt string to be sent to OpenAI's API based on the request attributes.
  - escapeJsonString(String input): Escapes special characters in the input string to prevent JSON injection.

CCGFM by Rushikesh

- fetchFromOpenAI(String prompt): Sends the prompt to OpenAI's API and fetches the generated content in response.

- **External Integration:**
  - The service interacts with the OpenAI API to generate content by sending POST requests to https://api.openai.com/v1/chat/completions.

5. ContentGeneratorApplication (Main Class)

- **Description:** The entry point of the Spring Boot application. This class starts the application and enables MongoDB repository support.
- **Key Annotations:**
  - @SpringBootApplication: Marks this class as the main entry point for the Spring Boot application.
  - @EnableMongoRepositories: Enables MongoDB repositories for the application.

Database

The application uses MongoDB for storing the content requests. The ContentRequest model is mapped to the content_requests collection in MongoDB. It stores all the fields, including the generated content.

Error Handling

- The application validates the incoming content generation requests to ensure all required fields are present. If any required field is missing, an IllegalArgumentException is thrown.
- If the interaction with the OpenAI API fails or no content is returned, an IOException is thrown, and a RuntimeException is raised to propagate the failure.
- Specific error messages are provided in cases of invalid input or API interaction failures.

Security

The application uses an API key (openai.api.key) stored in the application.properties file for authentication when making requests to the OpenAI API. This API key is required for all interactions with OpenAI's services.

External Dependencies

- **OkHttpClient:** Used to send HTTP requests to the OpenAI API.
- **Jackson (ObjectMapper):** Used for JSON parsing and handling the response from OpenAI.

Configuration

- The application requires the OpenAI API key to be configured in the application.properties file:

vbnet

Copy code

```
openai.api.key=your_api_key_here
```

This API key is used in ContentService to authenticate requests to OpenAI's API.

CCGFM by Rushikesh