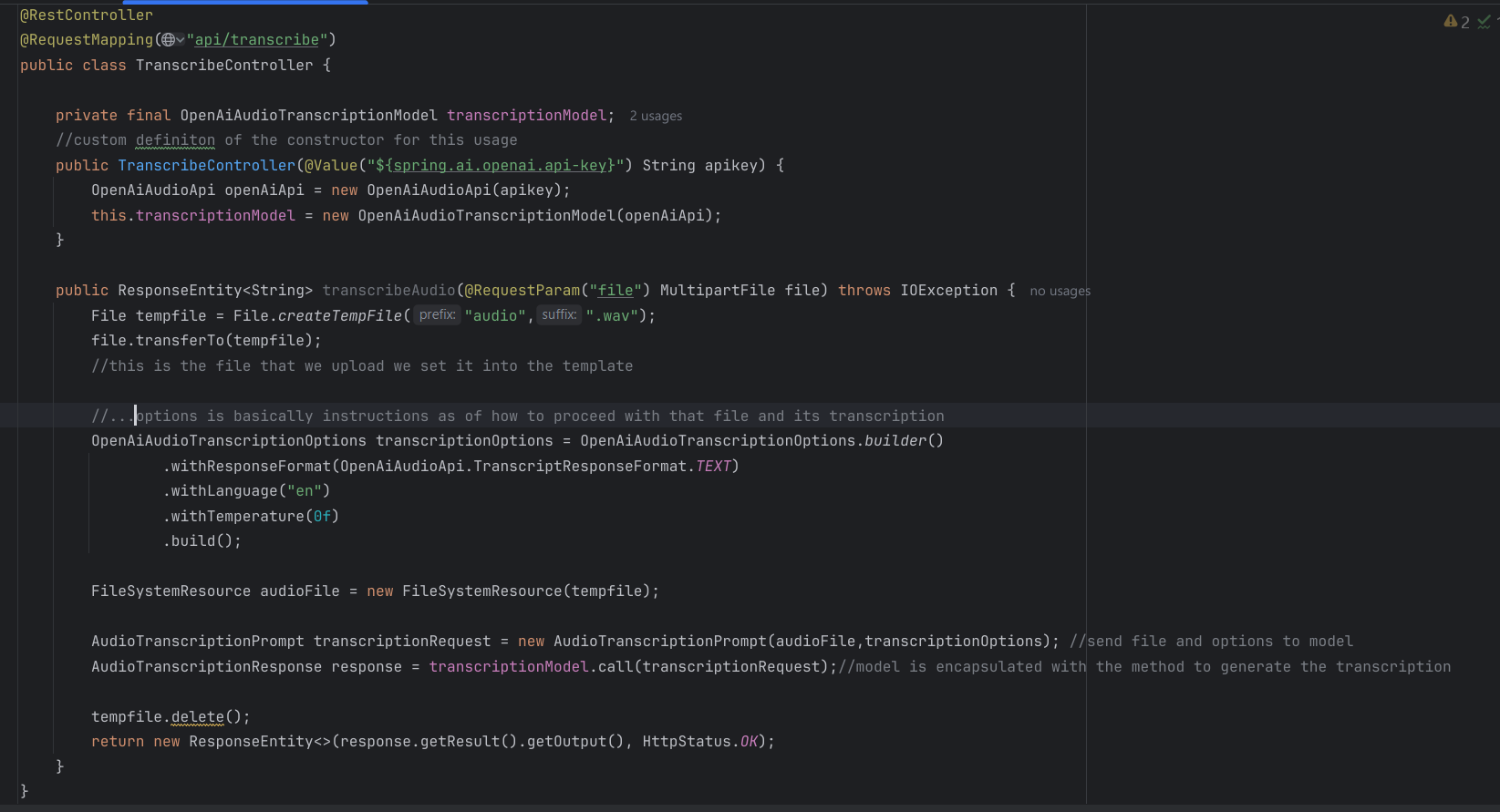
Audio Transcriber



**2. Constructor**

private final OpenAiAudioTranscriptionModel transcriptionModel;

public TranscribeController(@Value("${spring.ai.openai.api-key}") String apikey) {

OpenAiAudioApi openAiApi = new OpenAiAudioApi(apikey);

this.transcriptionModel = new OpenAiAudioTranscriptionModel(openAiApi);

}

* **Dependency Injection**:
  + The constructor uses the @Value annotation to inject the OpenAI API key, which is read from the application configuration (application.properties or application.yml) under the property spring.ai.openai.api-key.
  + Example in application.properties:
  + spring.ai.openai.api-key=YOUR\_API\_KEY
* **API Setup**:
  + OpenAiAudioApi: An instance of this class is created using the provided API key. This acts as the gateway to interact with OpenAI's audio transcription service.
  + OpenAiAudioTranscriptionModel: This is initialized using the OpenAiAudioApi instance, encapsulating the transcription logic.

**3. Transcription Endpoint**

public ResponseEntity<String> transcribeAudio(@RequestParam("file") MultipartFile file) throws IOException {

* **Method**: This method handles incoming HTTP requests to transcribe audio files.
* **@RequestParam("file") MultipartFile file**:
  + This annotation binds the HTTP request parameter named file to the MultipartFile object.
  + The MultipartFile represents the uploaded audio file.
* **ResponseEntity<String>**:
  + This allows the method to return a full HTTP response, including the body (transcription result) and status code.

**4. File Handling**

File tempfile = File.createTempFile("audio",".wav");

file.transferTo(tempfile);

* A temporary file is created with a prefix audio and suffix .wav.
* The uploaded MultipartFile is transferred to this temporary file using file.transferTo(tempfile). This step saves the uploaded file to the server for further processing.

**5. Transcription Options**

OpenAiAudioTranscriptionOptions transcriptionOptions = OpenAiAudioTranscriptionOptions.builder()

.withResponseFormat(OpenAiAudioApi.TranscriptResponseFormat.TEXT)

.withLanguage("en")

.withTemperature(0f)

.build();

* **OpenAiAudioTranscriptionOptions**:
  + This defines the settings for the transcription request.
  + **Response Format**: Specifies that the transcription result should be in plain text.
  + **Language**: Sets the language of the audio ("en" for English).
  + **Temperature**: Adjusts the randomness of the model's output. 0f means deterministic (least random).

**6. File Preparation for Transcription**

FileSystemResource audioFile = new FileSystemResource(tempfile);

AudioTranscriptionPrompt transcriptionRequest = new AudioTranscriptionPrompt(audioFile, transcriptionOptions);

* **FileSystemResource**: Wraps the temporary file for use in the transcription API.
* **AudioTranscriptionPrompt**: Combines the audio file and transcription options into a single request object.

**7. Call the Transcription Model**

AudioTranscriptionResponse response = transcriptionModel.call(transcriptionRequest);

* The transcriptionModel processes the transcription request and returns an AudioTranscriptionResponse.
* **response.getResult().getOutput()**:
  + Retrieves the transcription result (the text output of the audio).

**8. Clean Up and Response**

tempfile.delete();

return new ResponseEntity<>(response.getResult().getOutput(), HttpStatus.OK);

* **Temporary File Deletion**: The temporary file is deleted after processing to avoid clutter.
* **ResponseEntity**:
  + The transcription result is returned in the response body with an HTTP status code of 200 OK.

**How It Works**

1. A client sends an HTTP request with an audio file to the endpoint (e.g., POST /api/transcribe).
2. The uploaded file is saved temporarily on the server.
3. Transcription options are defined (e.g., language, format).
4. The file and options are sent to the OpenAI API using the OpenAiAudioTranscriptionModel.
5. The transcription response is retrieved, the temporary file is deleted, and the transcription result is sent back to the client.

# Necessary settings for connectivity

# 

# The properties you've shared are part of your **Spring Boot application's configuration**. These values are read at runtime and injected into your application, allowing it to connect to OpenAI's API and configure audio transcription. Below is an explanation of each property and why it's necessary:.

