Chatgpt Linux commands Developer's Guide

Table of Contents

> Introduction 3

> Supported OS 3

> Project Structure 3

1. CLI ENV 4
2. Spring Server 4
3. OpenAi Server 4

> the skills of use 4

1. ObjectMapper 4

2. ResTemplate 4

3. How to maintain chatGPT Question Session 5

4. Handling of CORS policies that occur during deployment 5

> API document 6

3.1 Ask a question 6

3.2 Termination of Session 9

3.3 Checking Grammar 9

3.4 Creating a Voice File Script 12

3.5 Emotional analysis 14

3.6 Translation of sentences 16

3.7 Sentence Summary 19

3.8 Extract Top Keywords 21

> reference 22

> Introduction

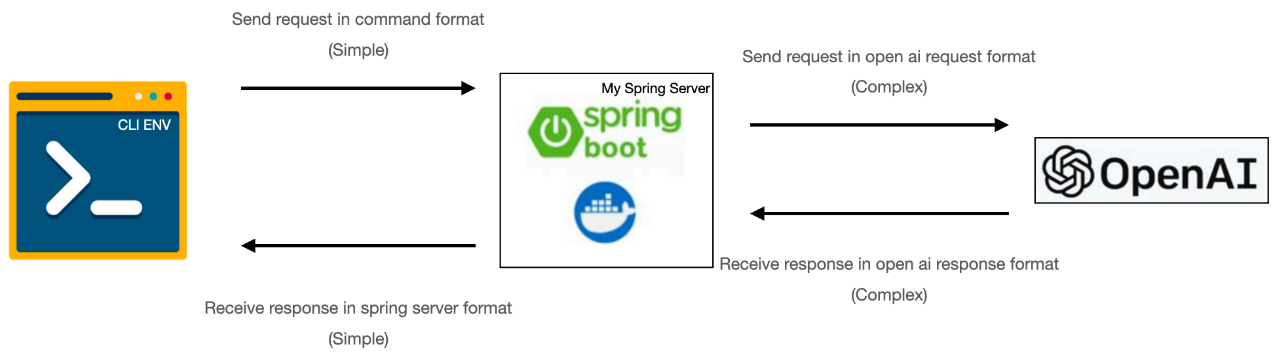
- Provides the Linux environment cli command to communicate with the open ai server in a Linux environment.
- By leveraging the various api provided by the Open ai server, users can easily use a variety of Linux command functions.

> Supported OS

	Amazon Linux	Mac OS	Ubuntu
Server	O	O	O
Amazon Web Service(AWS)	O	O	O
Google Cloud Platform(GCP)	X	O	O
Microsoft Azure	X	O	O

It is available in Linux, Mac OS, and Ubuntu environments, which are natively Linux-based operating systems. If you want to use the service in an environment other than the OS, you can modify the shell script in the shell file..

> Project Structure


CLI ENV
Send request in command format
(Simple)
Receive response in spring server format
(Simple)
My Spring Server
spring boot
Send request in open ai request format
(Complex)
Receive response in open ai response format
(Complex)
OpenAI

1. CLI ENV

- It is an environment in which a user executes a desired command.
- When you send a request in question format, it shows the response to the question.
- UI Roles

2. Spring Server

- The user receives the requested question and parses it into a request packet in the desired format (OpenAiRequestEntity class)
- The response to the OpenAi Server request sent by the server is parsed into a user-friendly response packet (OpenAiResponseEntity class)

3. OpenAi Server

- It is actually a server operated by Open Ai.(Server with chatGPT service)
- The server provides api for several chat GPT services.
- More api can be found at (https://platform.openai.com).

> the skills of use

---

1. ObjectMapper

- When using the JSON format, it is a technique used to serialize responses and deserialize requests.
- That is, it provides a function of performing conversion between JSON data and a Java object.
- JIt is a class of Jackson Libraries.
- When converting JSON data to Java objects, the readValue() method of the ObjectMapper class is used.
- When converting Java objects to JSON data, use the writeValue() method of the ObjectMapper class

2. ResTemplate

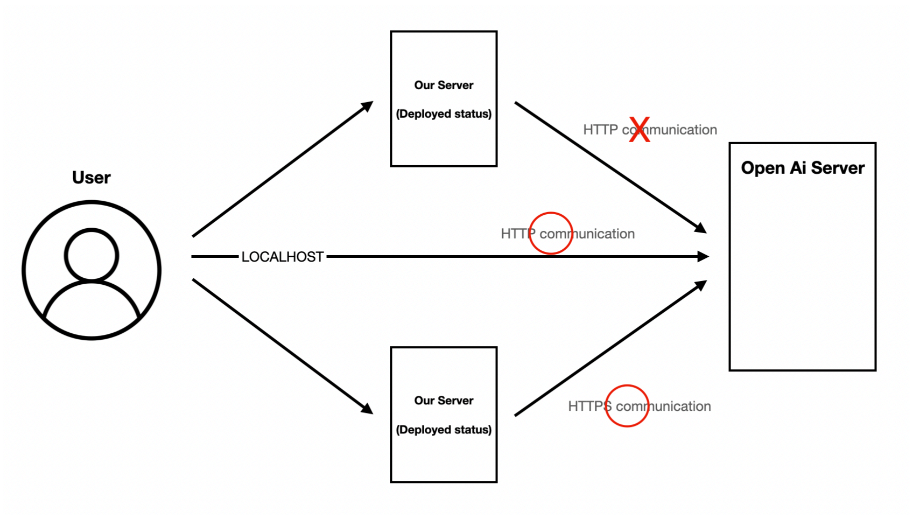
- It is a technology used to send HTTP requests for RESTful web services and receive responses.
- In other words, it is a technology used when an external API is used in the spring to send a request to the server and receive a response to the request.

- RestTemplate provides a variety of methods for creating and processing HTTP requests in a simple manner.
- Headers, parameters, text, etc. required for the request can be set.
- When receiving a response, the status code, header, text, etc. of the response may be checked response may be checked.
- In the future, we will also add a technology that uses WebClient instead of that technology. (From Spring 5, we recommend using WebClient instead of RestTemplate.)

3. How to maintain chatGPT Question Session

- When you ask a question on chatgpt, in the CLI environment, there is a problem that the content of the previous question is not connected to the next question.
- Method 1: Connect the questions. -> This also does not work normally because the answer to the question is not connected.
- Method 2: Connect questions + answers to those questions. -> Check that it is operating normally.

4. Handling of CORS policies that occur during deployment



- By default, the OpenAI server is open to HTTPS communication.
- So if you want to communicate with OpenAI using HTTP, you cannot communicate with OpenAI.
- However, because the OpenAI server allowed communication with LOCALHOST, it can communicate using LOCALHOST (LOCALHOST uses HTTP by default).
- We solved the problem through HTTPS communication.

> API document

3.1 Ask a question

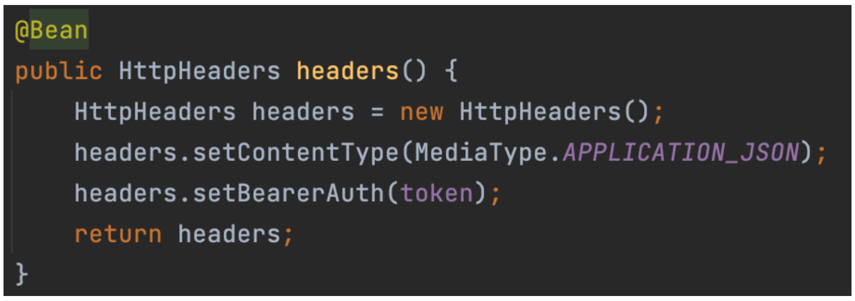
```
curl https://api.spring-chatgpt-communication.com/v1/chat/ \
  -H "Content-Type: text/plain" \
  -d '{
    what is open source?
   }'
```

The above curl command was created through alias as follows.

```
chatAsk -m "what is open source ?"
```

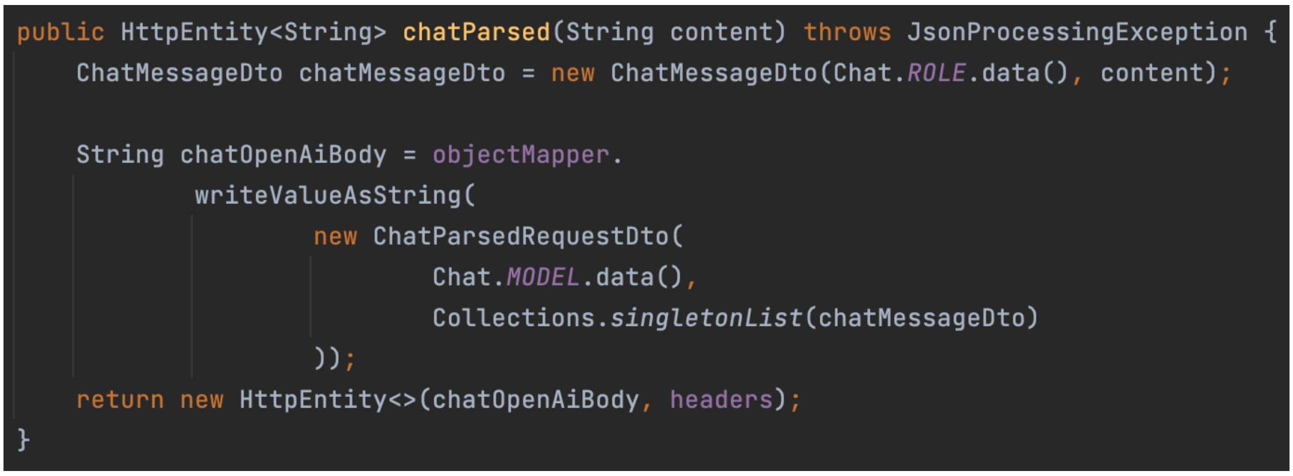
You can use the chatAsk command to ask questions.

Create an HTTP request packet to send to the open ai server



```
@Bean
public HttpHeaders headers() {
    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    headers.setBearerAuth(token);
    return headers;
}
```

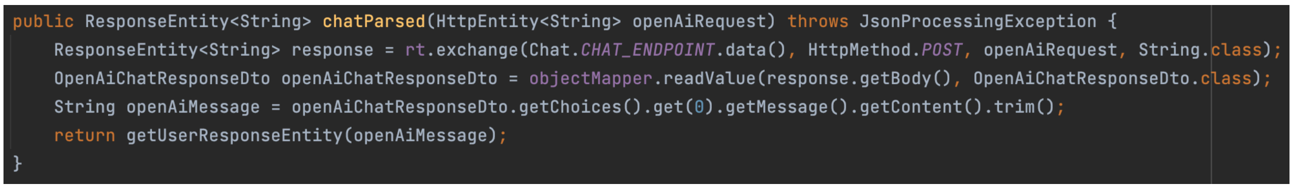
Create the HTTP request header portion. Media type is application/json type, and write the token value provided by open ai in Bear.



```java
public HttpEntity<String> chatParsed(String content) throws JsonProcessingException {
    ChatMessageDto chatMessageDto = new ChatMessageDto(Chat.ROLE.data(), content);

    String chatOpenAiBody = objectMapper.
            writeValueAsString(
                    new ChatParsedRequestDto(
                            Chat.MODEL.data(),
                            Collections.singletonList(chatMessageDto)
                    ));
    return new HttpEntity<>(chatOpenAiBody, headers);
}
```

Create the HTTP request body part. The data that will be included in the Body part is written in the model name and message list format.



```java
public ResponseEntity<String> chatParsed(HttpEntity<String> openAiRequest) throws JsonProcessingException {
    ResponseEntity<String> response = rt.exchange(Chat.CHAT_ENDPOINT.data(), HttpMethod.POST, openAiRequest, String.class);
    OpenAiChatResponseDto openAiChatResponseDto = objectMapper.readValue(response.getBody(), OpenAiChatResponseDto.class);
    String openAiMessage = openAiChatResponseDto.getChoices().get(0).getMessage().getContent().trim();
    return getUserResponseEntity(openAiMessage);
}
```

The request url, HTTP method of HTTP request packet is set and the request is sent to the Open ai server using the restTemplate. The completed HTTP request packet is finally as follows.

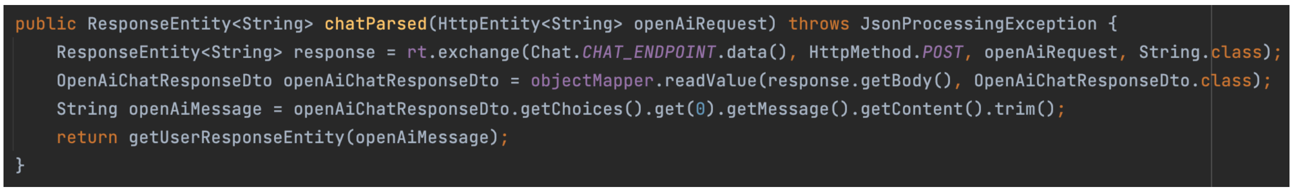
HTTP request packet

```
curl https://api.openai.com/v1/chat/completions \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $OPENAI_API_KEY" \
  -d '{
    "model": "gpt-3.5-turbo",
    "messages": [{"role": "user", "content": "what is open source?"}]
  }'
```

HTTP response packet

```
{
  "id": "chatcmpl-123",
  "object": "chat.completion",
  "created": 1677652288,
  "choices": [{
    "index": 0,
    "message": {
      "role": "assistant",
      "content": "\n\n Open source refers to a software licensing model where the source c
Freedom: Open source software provides the freedom for users to access, modify, and distri
",
    },
    "finish_reason": "stop"
  }],
  "usage": {
    "prompt_tokens": 9,
    "completion_tokens": 12,
    "total_tokens": 21
  }
}
```

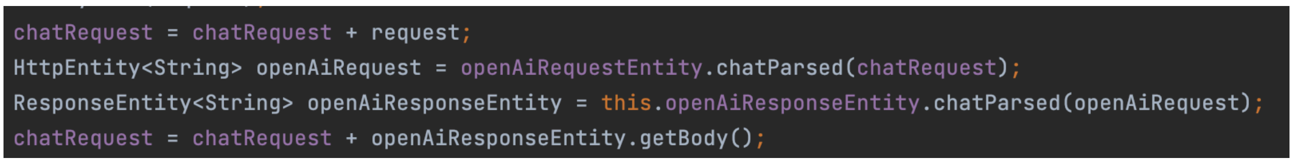
You have set up the OpenAiChatResponseDto class for that response.



```java
public ResponseEntity<String> chatParsed(HttpEntity<String> openAiRequest) throws JsonProcessingException {
    ResponseEntity<String> response = rt.exchange(Chat.CHAT_ENDPOINT.data(), HttpMethod.POST, openAiRequest, String.class);
    OpenAiChatResponseDto openAiChatResponseDto = objectMapper.readValue(response.getBody(), OpenAiChatResponseDto.class);
    String openAiMessage = openAiChatResponseDto.getChoices().get(0).getMessage().getContent().trim();
    return getUserResponseEntity(openAiMessage);
}
```

It then provides the content value that the user needs through Java's reference operation.

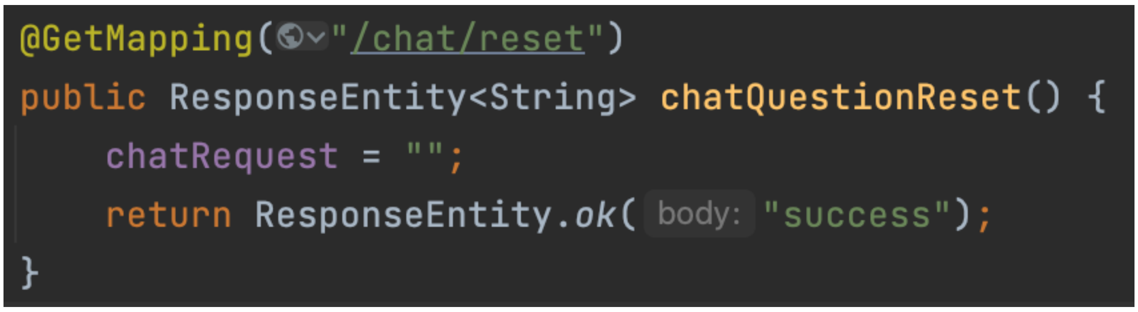
Session retention proceeds as follows.



```java
chatRequest = chatRequest + request;
HttpEntity<String> openAiRequest = openAiRequestEntity.chatParsed(chatRequest);
ResponseEntity<String> openAiResponseEntity = this.openAiResponseEntity.chatParsed(openAiRequest);
chatRequest = chatRequest + openAiResponseEntity.getBody();
```

You can add questions + answers to your questions, while sending them all to the open ai server for continuous questioning sessions.

3.2 Termination of Session



```java
@GetMapping("/chat/reset")
public ResponseEntity<String> chatQuestionReset() {
    chatRequest = "";
    return ResponseEntity.ok( body: "success");
}
```

Through the Ask Questions API, the question session will be terminated by initializing all the questions that were previously accumulated and the question session was maintained.

3.3 Checking Grammar

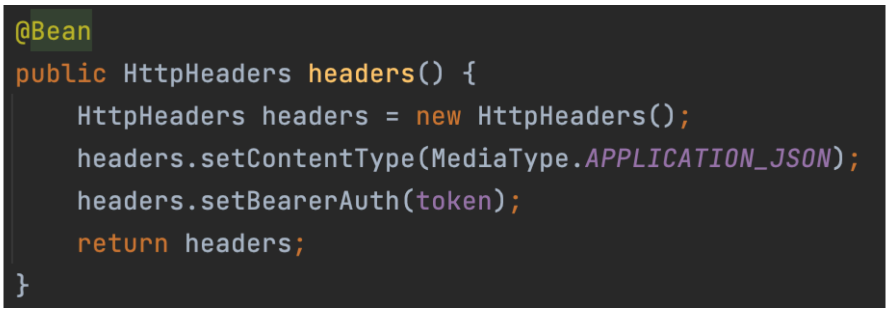
```
curl https://api.spring-chatgpt-communication.com/v1/gc \
  -H "Content-Type: text/plain" \
  -d '{
    god morning
  }'
```

The above curl command was created through alias as follows.

```
chatAsk -gc "god morning"
```

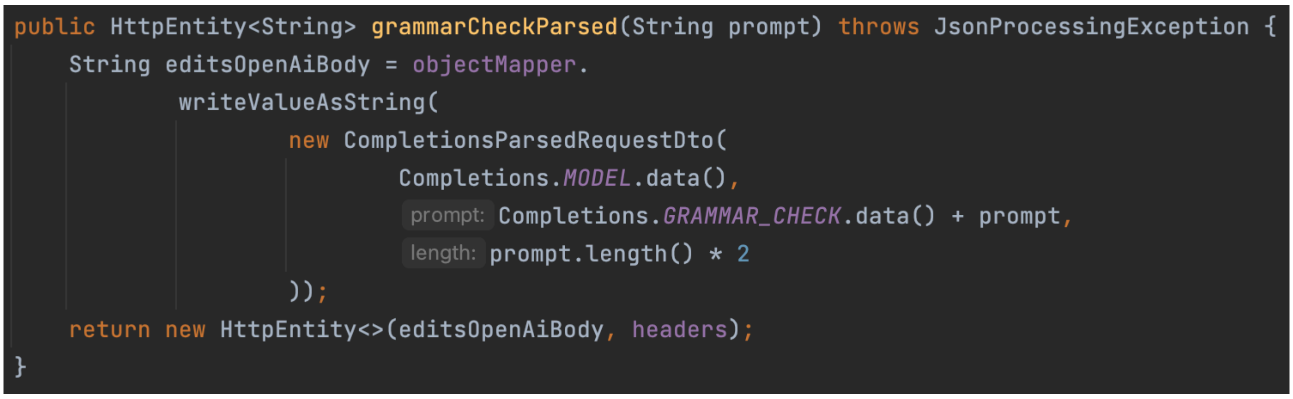
You can use the chatAsk command to check the grammar.

Create an HTTP request packet to send to the open ai server



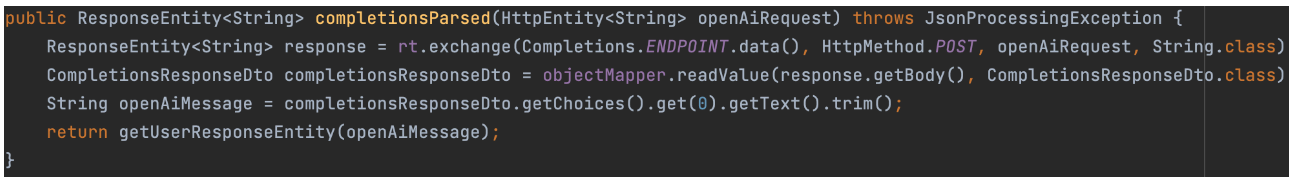
```java
@Bean
public HttpHeaders headers() {
    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    headers.setBearerAuth(token);
    return headers;
}
```

Create the HTTP request header portion. Media type is application/json type, and write the token value provided by open ai in Bear.



```java
public HttpEntity<String> grammarCheckParsed(String prompt) throws JsonProcessingException {
    String editsOpenAiBody = objectMapper.
            writeValueAsString(
                    new CompletionsParsedRequestDto(
                            Completions.MODEL.data(),
                            prompt: Completions.GRAMMAR_CHECK.data() + prompt,
                            length: prompt.length() * 2
                    ));
    return new HttpEntity<>(editsOpenAiBody, headers);
}
```

Create the HTTP request body part. For the data to be included in the Body part, the length information of the model name, prompt, and prompt is created.



```java
public ResponseEntity<String> completionsParsed(HttpEntity<String> openAiRequest) throws JsonProcessingException {
    ResponseEntity<String> response = rt.exchange(Completions.ENDPOINT.data(), HttpMethod.POST, openAiRequest, String.class)
    CompletionsResponseDto completionsResponseDto = objectMapper.readValue(response.getBody(), CompletionsResponseDto.class)
    String openAiMessage = completionsResponseDto.getChoices().get(0).getText().trim();
    return getUserResponseEntity(openAiMessage);
}
```

The request url, HTTP method of HTTP request packet is set and the request is sent to the Open ai server using the restTemplate. The completed HTTP request packet is finally as follows.

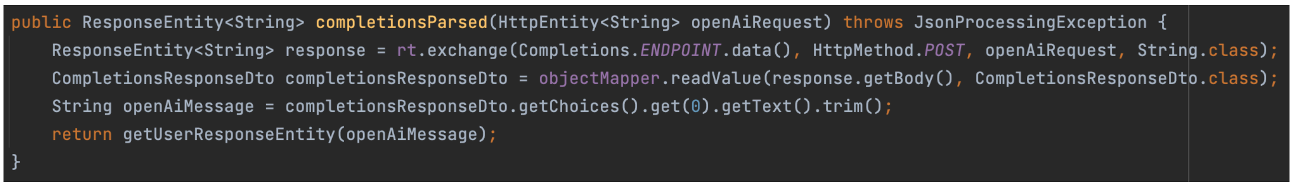
HTTP request packet

```
curl https://api.openai.com/v1/edits \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $OPENAI_API_KEY" \
  -d '{
    "model": "text-davinci-edit-001",
    "input": "god morning",
    "instruction": "Fix the spelling mistakes"
  }'
```

HTTP response packet

```
{
  "object": "edit",
  "created": 1589478378,
  "choices": [
    {
      "text": "good morning",
      "index": 0,
    }
  ],
  "usage": {
    "prompt_tokens": 25,
    "completion_tokens": 32,
    "total_tokens": 57
  }
}
```

You have set up the CompletionResponseDto class for that response.



```java
public ResponseEntity<String> completionsParsed(HttpEntity<String> openAiRequest) throws JsonProcessingException {
    ResponseEntity<String> response = rt.exchange(Completions.ENDPOINT.data(), HttpMethod.POST, openAiRequest, String.class);
    CompletionsResponseDto completionsResponseDto = objectMapper.readValue(response.getBody(), CompletionsResponseDto.class);
    String openAiMessage = completionsResponseDto.getChoices().get(0).getText().trim();
    return getUserResponseEntity(openAiMessage);
}
```

It then provides the text value that the user needs through Java's reference operation.

3.4 Creating a Voice File Script

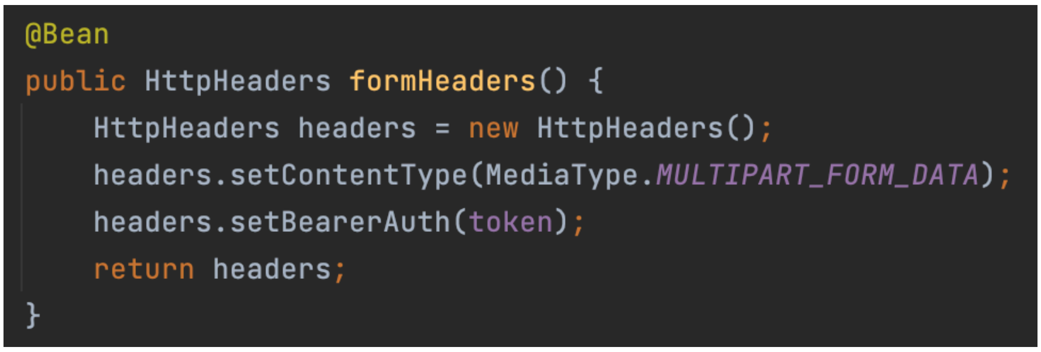
```
curl https://api.spring-chatgpt-communication.com/v1/at \
  -H "Content-Type: multipart/form-data" \
  -F file="@/path/to/file/audio.mp3" \
```

The above curl command was created through alias as follows.

```
chatAsk -at "@Desktop/음성파일.m4a"
```

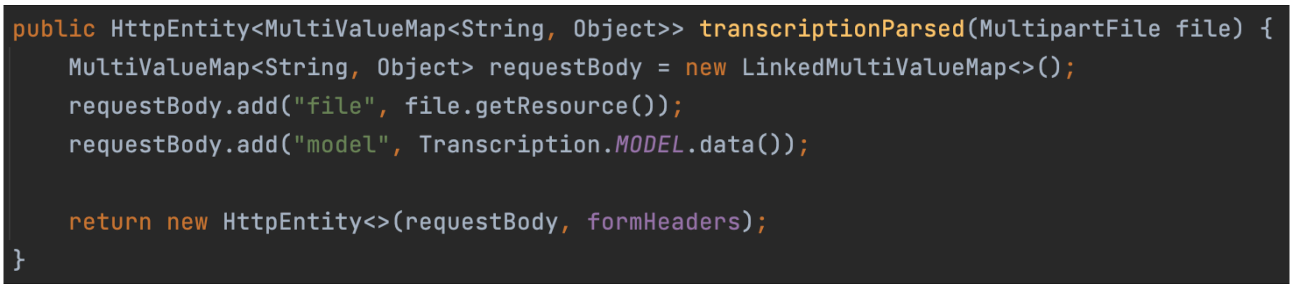
You can use the chatAsk command to create a voice file script.

Create an HTTP request packet to send to the open ai server



```
@Bean
public HttpHeaders formHeaders() {
    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.MULTIPART_FORM_DATA);
    headers.setBearerAuth(token);
    return headers;
}
```

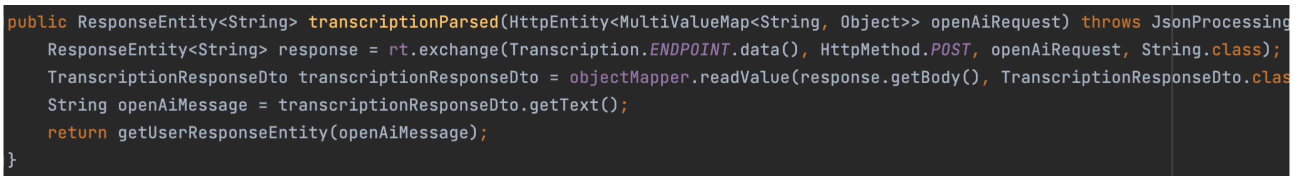
Create the HTTP request header portion. The Media type is the MULTIPART_FORM_DATA type and writes the value of the token provided by open ai in Bear.



```java
public HttpEntity<MultiValueMap<String, Object>> transcriptionParsed(MultipartFile file) {
    MultiValueMap<String, Object> requestBody = new LinkedMultiValueMap<>();
    requestBody.add("file", file.getResource());
    requestBody.add("model", Transcription.MODEL.data());

    return new HttpEntity<>(requestBody, formHeaders);
}
```

Create the HTTP request body part. The data that will be included in the Body part will be a voice file and a model name.



```java
public ResponseEntity<String> transcriptionParsed(HttpEntity<MultiValueMap<String, Object>> openAiRequest) throws JsonProcessing
    ResponseEntity<String> response = rt.exchange(Transcription.ENDPOINT.data(), HttpMethod.POST, openAiRequest, String.class);
    TranscriptionResponseDto transcriptionResponseDto = objectMapper.readValue(response.getBody(), TranscriptionResponseDto.clas
    String openAiMessage = transcriptionResponseDto.getText();
    return getUserResponseEntity(openAiMessage);
}
```

The request url, HTTP method of HTTP request packet is set and the request is sent to the Open ai server using the restTemplate. The completed HTTP request packet is finally as follows.

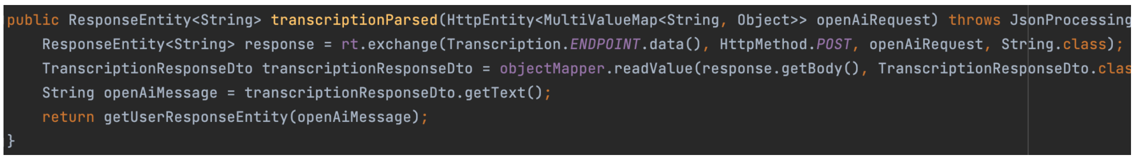
HTTP request packet

```
curl https://api.openai.com/v1/audio/transcriptions \
  -H "Authorization: Bearer $OPENAI_API_KEY" \
  -H "Content-Type: multipart/form-data" \
  -F file="@/path/to/file/audio.mp3" \
  -F model="whisper-1"
```

HTTP response packet

```
{
  "text": "Hello, my name is Wolfgang and I come from Germany.
}
```

You have set up a transcriptionResponseDto class for that response.



```java
public ResponseEntity<String> transcriptionParsed(HttpEntity<MultiValueMap<String, Object>> openAiRequest) throws JsonProcessing
    ResponseEntity<String> response = rt.exchange(Transcription.ENDPOINT.data(), HttpMethod.POST, openAiRequest, String.class);
    TranscriptionResponseDto transcriptionResponseDto = objectMapper.readValue(response.getBody(), TranscriptionResponseDto.clas
    String openAiMessage = transcriptionResponseDto.getText();
    return getUserResponseEntity(openAiMessage);
}
```

It then provides the text value that the user needs through Java's reference operation.

3.5 Emotional analysis

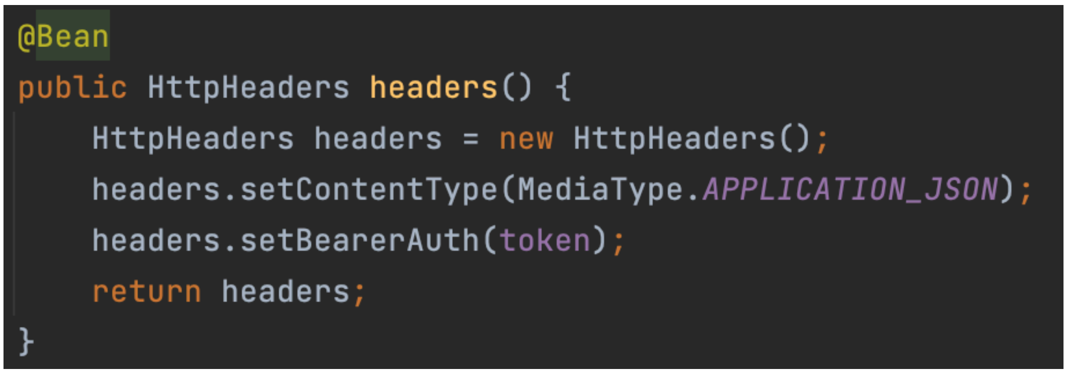
```
curl https://api.spring-chatgpt-communication.com/v1/mood \
  -H "Content-Type: text/plain" \
  -d '{
    i am so happy
  }'
```

The above curl command was created through alias as follows.

```
chatAsk -m "i am so happy"
```

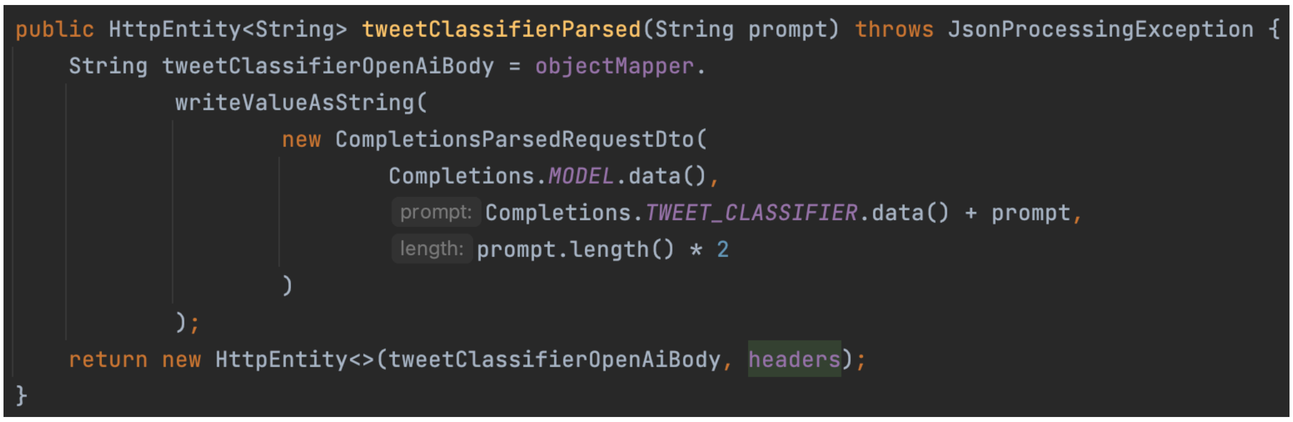
Use the chatAsk command to perform sentence emotion analysis.

Create an HTTP request packet to send to the open ai server



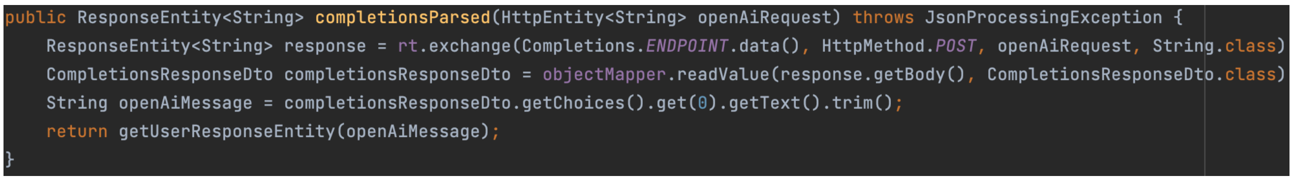
```java
@Bean
public HttpHeaders headers() {
    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    headers.setBearerAuth(token);
    return headers;
}
```

Create the HTTP request header portion. Media type is application/json type, and write the token value provided by open ai in Bear.



```
public HttpEntity<String> tweetClassifierParsed(String prompt) throws JsonProcessingException {
    String tweetClassifierOpenAiBody = objectMapper.
            writeValueAsString(
                    new CompletionsParsedRequestDto(
                            Completions.MODEL.data(),
                            prompt: Completions.TWEET_CLASSIFIER.data() + prompt,
                            length: prompt.length() * 2
                    )
            );
    return new HttpEntity<>(tweetClassifierOpenAiBody, headers);
}
```

Create the HTTP request body part. For the data to be included in the Body part, information of model name, sentence and prompt, and length X2 of prompt is created.



```
public ResponseEntity<String> completionsParsed(HttpEntity<String> openAiRequest) throws JsonProcessingException {
    ResponseEntity<String> response = rt.exchange(Completions.ENDPOINT.data(), HttpMethod.POST, openAiRequest, String.class)
    CompletionsResponseDto completionsResponseDto = objectMapper.readValue(response.getBody(), CompletionsResponseDto.class)
    String openAiMessage = completionsResponseDto.getChoices().get(0).getText().trim();
    return getUserResponseEntity(openAiMessage);
}
```

The request url, HTTP method of HTTP request packet is set and the request is sent to the Open ai server using the restTemplate. The completed HTTP request packet is finally as follows

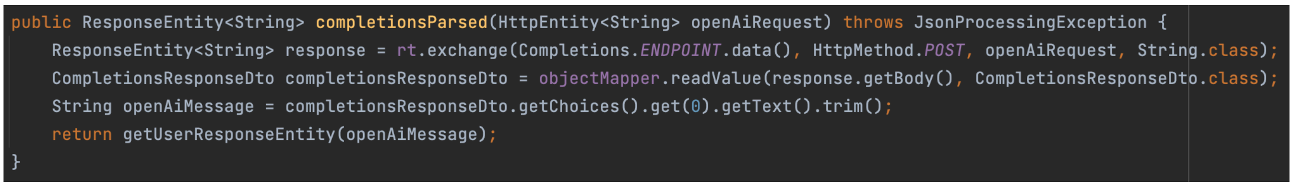
HTTP request packet

```
curl https://api.openai.com/v1/completions \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $OPENAI_API_KEY" \
  -d '{
  "model": "text-davinci-003",
  "prompt": "Decide whether a Tweet's sentiment is positive, neutral, or negative.\n\n
  Tweet: \"i am so happy\"\nSentiment:",
  "temperature": 0,
  "max_tokens": 60,
  "top_p": 1.0,
  "frequency_penalty": 0.5,
  "presence_penalty": 0.0
}'
```

HTTP response packet

```
{
    "text" : "positive"
}
```

You have set up the CompletionResponseDto class for that response.



```java
public ResponseEntity<String> completionsParsed(HttpEntity<String> openAiRequest) throws JsonProcessingException {
    ResponseEntity<String> response = rt.exchange(Completions.ENDPOINT.data(), HttpMethod.POST, openAiRequest, String.class);
    CompletionsResponseDto completionsResponseDto = objectMapper.readValue(response.getBody(), CompletionsResponseDto.class);
    String openAiMessage = completionsResponseDto.getChoices().get(0).getText().trim();
    return getUserResponseEntity(openAiMessage);
}
```

It then provides the text value that the user needs through Java's reference operation.

3.6 Translation of sentences

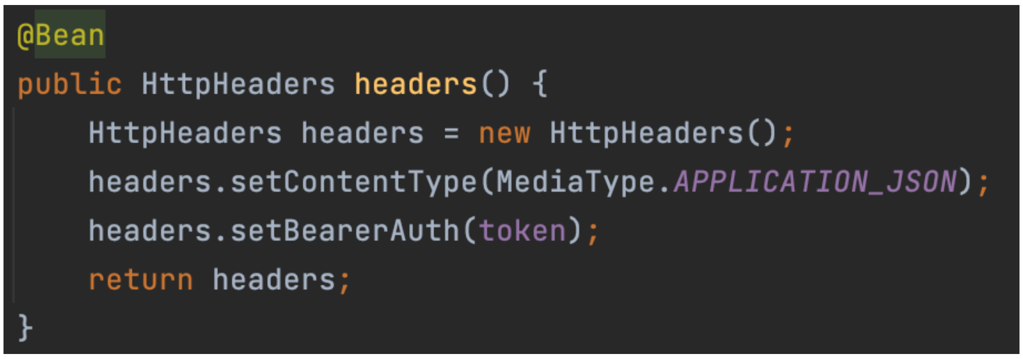
```
curl https://api.spring-chatgpt-communication.com/v1/trans \
  -H "Content-Type: text/plain" \
  -d '{
    hi, this is seo ji hyeon
  }'
```

The above curl command was created through alias as follows.

```
chatAsk -t "hi, this is seo ji hyeon"
```

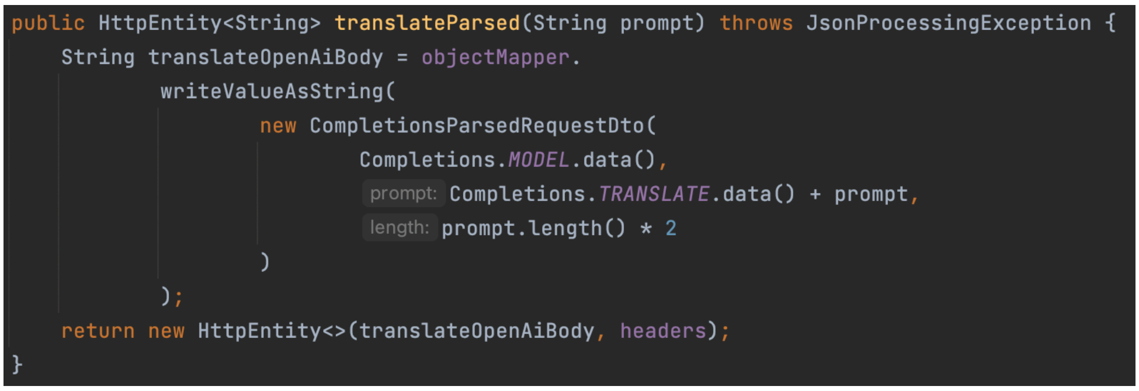
You can use the chatAsk command to translate sentences.

Create an HTTP request packet to send to the open ai server



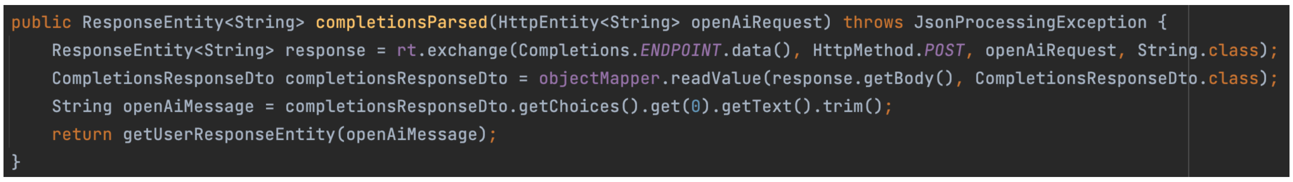
```java
@Bean
public HttpHeaders headers() {
    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    headers.setBearerAuth(token);
    return headers;
}
```

Create the HTTP request header portion. Mine type is application/json type and write the token value provided by open ai in Bear.



```java
public HttpEntity<String> translateParsed(String prompt) throws JsonProcessingException {
    String translateOpenAiBody = objectMapper.
            writeValueAsString(
                    new CompletionsParsedRequestDto(
                            Completions.MODEL.data(),
                            prompt: Completions.TRANSLATE.data() + prompt,
                            length: prompt.length() * 2
                    )
            );
    return new HttpEntity<>(translateOpenAiBody, headers);
}
```

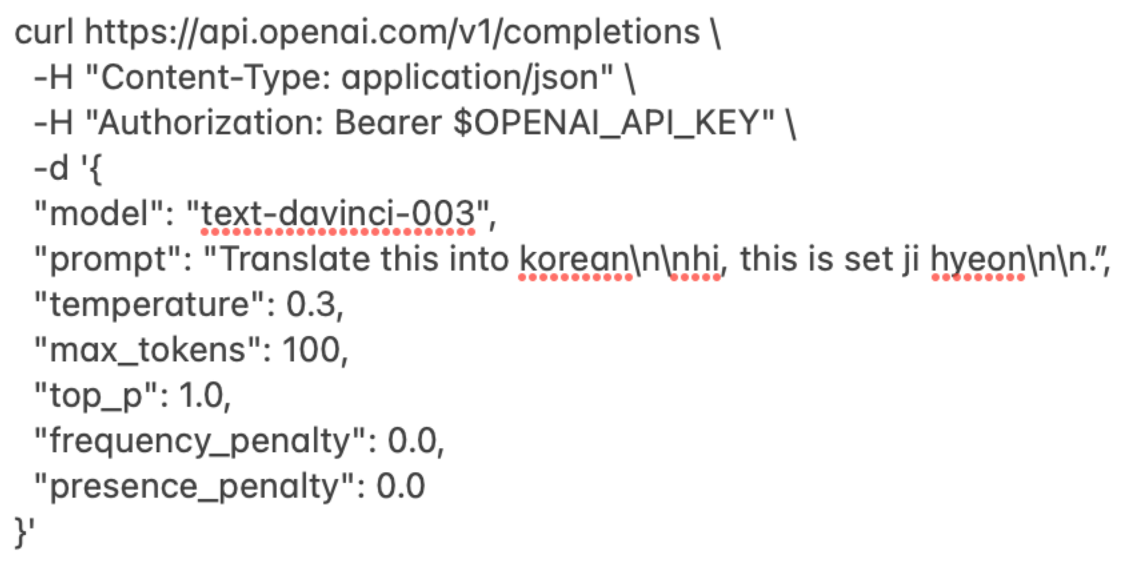
Create the HTTP request body part. For the data to be included in the Body part, information of model name, sentence and prompt, and length X2 of prompt is created.



```java
public ResponseEntity<String> completionsParsed(HttpEntity<String> openAiRequest) throws JsonProcessingException {
    ResponseEntity<String> response = rt.exchange(Completions.ENDPOINT.data(), HttpMethod.POST, openAiRequest, String.class);
    CompletionsResponseDto completionsResponseDto = objectMapper.readValue(response.getBody(), CompletionsResponseDto.class);
    String openAiMessage = completionsResponseDto.getChoices().get(0).getText().trim();
    return getUserResponseEntity(openAiMessage);
}
```

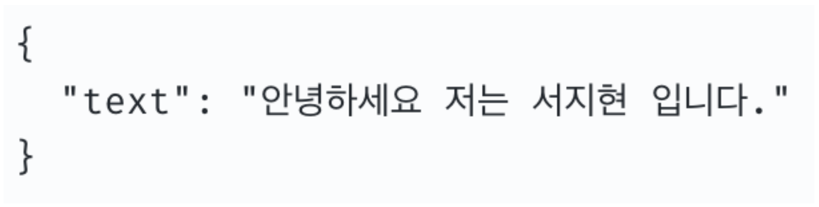
The request url, HTTP method of HTTP request packet is set and the request is sent to the Open ai server using the restTemplate. The completed HTTP request packet is finally as follows.

HTTP request packet



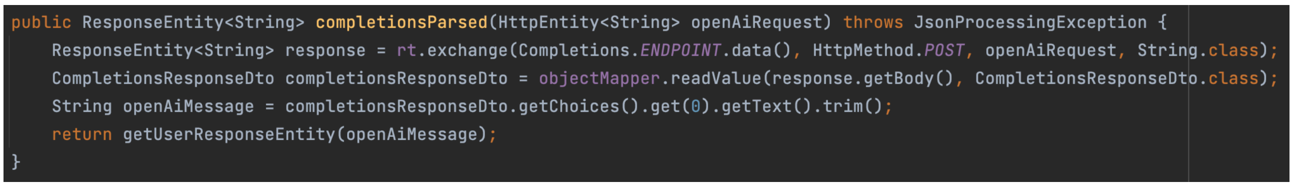
```
curl https://api.openai.com/v1/completions \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $OPENAI_API_KEY" \
  -d '{
  "model": "text-davinci-003",
  "prompt": "Translate this into korean\n\nhi, this is set ji hyeon\n\n.",
  "temperature": 0.3,
  "max_tokens": 100,
  "top_p": 1.0,
  "frequency_penalty": 0.0,
  "presence_penalty": 0.0
}'
```

HTTP response packet



```
{
   "text": "안녕하세요 저는 서지현 입니다."
}
```

You have set up the CompletionResponseDto class for that response.



```
public ResponseEntity<String> completionsParsed(HttpEntity<String> openAiRequest) throws JsonProcessingException {
    ResponseEntity<String> response = rt.exchange(Completions.ENDPOINT.data(), HttpMethod.POST, openAiRequest, String.class);
    CompletionsResponseDto completionsResponseDto = objectMapper.readValue(response.getBody(), CompletionsResponseDto.class);
    String openAiMessage = completionsResponseDto.getChoices().get(0).getText().trim();
    return getUserResponseEntity(openAiMessage);
}
```

It then provides the text value that the user needs through Java's reference operation.

3.7 Sentence Summary

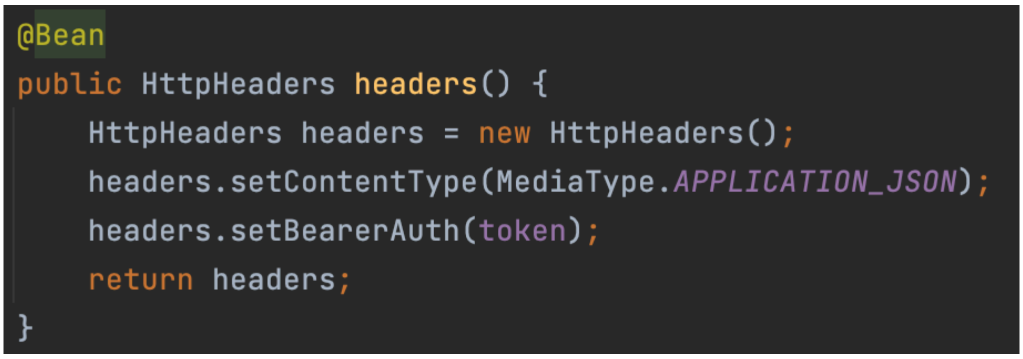
```
curl https://api.spring-chatgpt-communication.com/v1/summarize \
  -H "Content-Type: text/plain" \
  -d '{
        i love movie, i love soccer
  }'
```

The above curl command was created through alias as follows.

```
chatAsk -s "i love movie, i love soccer"
```

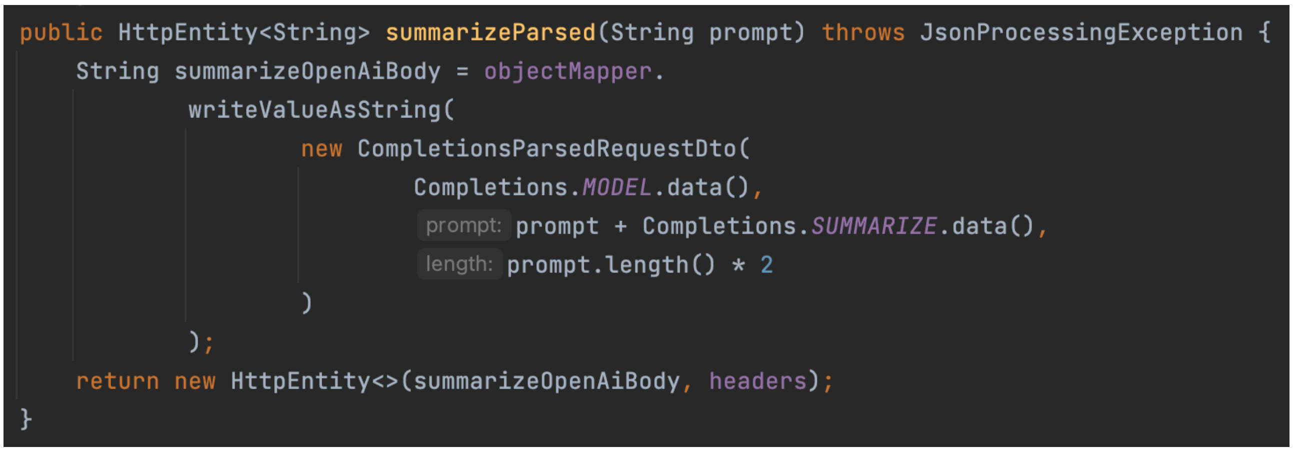
Use the chatAsk command to perform the sentence summary function.

Create an HTTP request packet to send to the open ai server



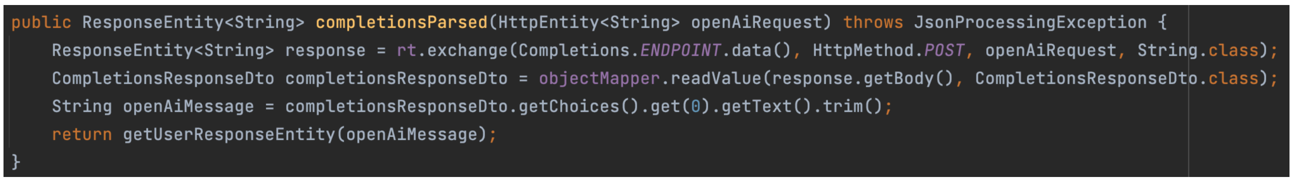
```java
@Bean
public HttpHeaders headers() {
    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    headers.setBearerAuth(token);
    return headers;
}
```

Create the HTTP request header portion. Mine type is application/json type and write the token value provided by open ai in Bear.



```java
public HttpEntity<String> summarizeParsed(String prompt) throws JsonProcessingException {
    String summarizeOpenAiBody = objectMapper.
            writeValueAsString(
                    new CompletionsParsedRequestDto(
                            Completions.MODEL.data(),
                            prompt: prompt + Completions.SUMMARIZE.data(),
                            length: prompt.length() * 2
                    )
            );
    return new HttpEntity<>(summarizeOpenAiBody, headers);
}
```

Create the HTTP request body part. For the data to be included in the Body part, information of model name, sentence and prompt, and length X2 of prompt is created.



```java
public ResponseEntity<String> completionsParsed(HttpEntity<String> openAiRequest) throws JsonProcessingException {
    ResponseEntity<String> response = rt.exchange(Completions.ENDPOINT.data(), HttpMethod.POST, openAiRequest, String.class);
    CompletionsResponseDto completionsResponseDto = objectMapper.readValue(response.getBody(), CompletionsResponseDto.class);
    String openAiMessage = completionsResponseDto.getChoices().get(0).getText().trim();
    return getUserResponseEntity(openAiMessage);
}
```

The request url, HTTP method of HTTP request packet is set and the request is sent to the Open ai server using the restTemplate. The completed HTTP request packet is finally as follows.

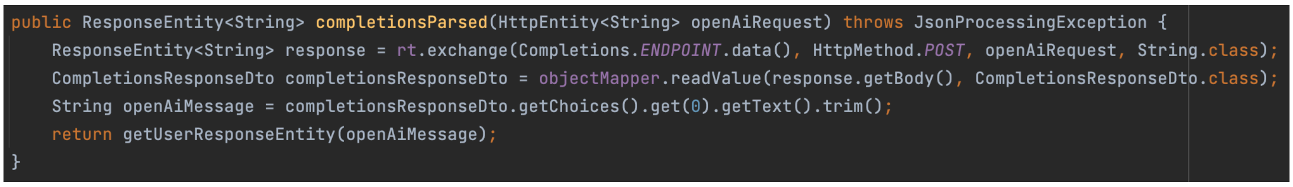
HTTP request packet

```
curl https://api.openai.com/v1/completions \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $OPENAI_API_KEY" \
  -d '{
  "model": "text-davinci-003",
  "prompt": "Summarize this for a second-grade student:\n\n i love movie and soccer",
  "temperature": 1,
  "max_tokens": 64,
  "top_p": 1.0,
  "frequency_penalty": 0.0,
  "presence_penalty": 0.0
}'
```

HTTP response packet

```
{
     "text" : "i love movie and soccer"
}
```

You have set up the CompletionResponseDto class for that response.그 후 자바의

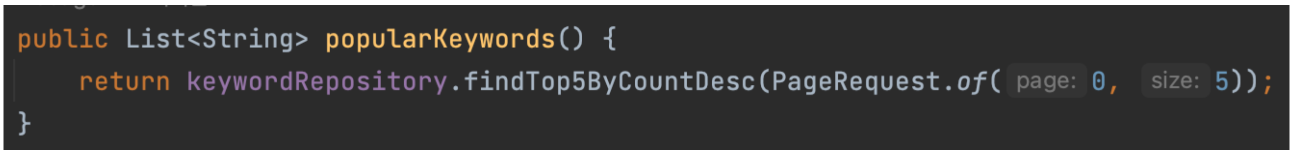


```java
public ResponseEntity<String> completionsParsed(HttpEntity<String> openAiRequest) throws JsonProcessingException {
    ResponseEntity<String> response = rt.exchange(Completions.ENDPOINT.data(), HttpMethod.POST, openAiRequest, String.class);
    CompletionsResponseDto completionsResponseDto = objectMapper.readValue(response.getBody(), CompletionsResponseDto.class);
    String openAiMessage = completionsResponseDto.getChoices().get(0).getText().trim();
    return getUserResponseEntity(openAiMessage);
}
```

Provides the text value that the user needs through the reference operation.

3.8 Extract Top Keywords

Ask questions api saves keywords to the database based on the sentences you ask. Top keyword extraction api is an api that provides the top 5 keywords most frequently asked by users among stored keywords.



```java
public List<String> popularKeywords() {
    return keywordRepository.findTop5ByCountDesc(PageRequest.of( page: 0,  size: 5));
}
```

Five top keywords can be invoked through the feature page-nation in SpringDataJpa as follows.

> reference

- https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/client/RestTemplate.html

- https://platform.openai.com/examples

Thank you.