

# ChatGpt API 활용

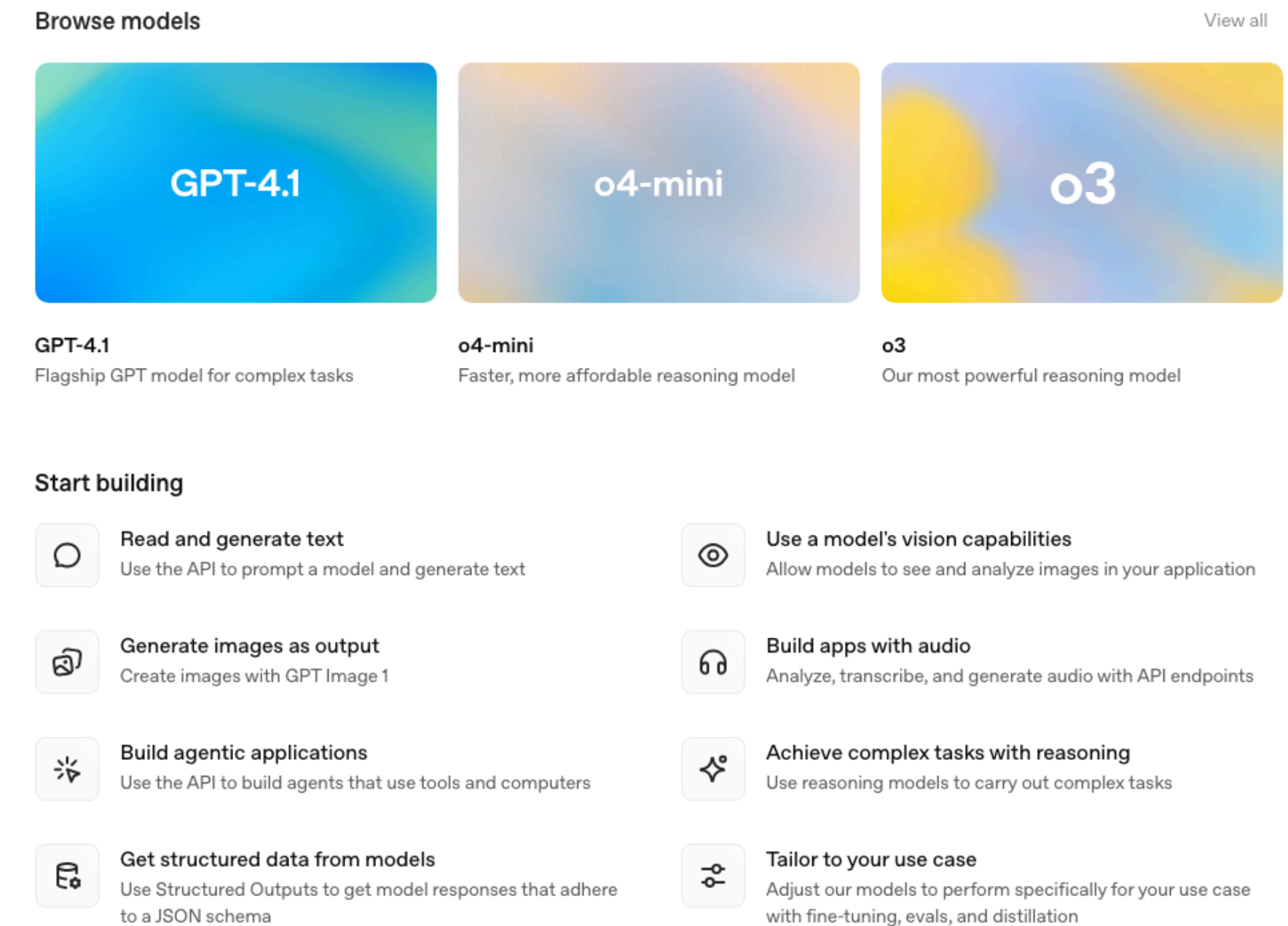
Day2. 텍스트 생성 모델

**OpenAI API**

# OpenAI API

## OpenAI API란?

- 대형 언어 모델(LLM)을 외부 애플리케이션이나 서비스에서 프로그래밍 방식으로 사용할 수 있도록 만든 인터페이스
- 텍스트, 이미지, 음성 생성 등 다양한 기능 제공
- 용도에 맞는 모델을 선택하여 사용할 수 있음
- 다양한 RESTful API endpoint를 제공



# OpenAI API

## OpenAI API Endpoint

API	주소 (endpoint)	용도
Chat Completions	/v1/chat/completions	GPT 모델과 대화
Completions	/v1/completions	고전적인 텍스트 생성 (단일 프롬프트 -> 단일 출력)
Responses	/v1/responses	대규모 응답 구조, 톨 호출, JSON 구조 출력 등 고급 기능 제공
Embeddings	/v1/embeddings	텍스트 -> 벡터로 변환 (유사도 검색, RAG 등에서 사용)
Moderations	/v1/moderations	입력 텍스트의 유해성, 욕석, 혐오 등 필터링
Whisper	/v1/audio/transcriptions	음성 -> 텍스트 (STT)
TTS	/v1/audio/speech	텍스트 -> 음성 (TTS)
Files	/v1/files	문서 업로드 및 처리
Images	/v1/images/generations	이미지 생성 (DALL·E)
Assistants	/v1/assistants	툴 기반 멀티모달 챗봇 (파일, 코드 실행 등)
vector_stores	v1/vector_stores	임베딩된 문서들을 저장하고 검색에 활용
batches	v1/batches	비동기식 대량 작업
models	v1/models	사용할 수 있는 모델 확인 (gpt-4, gpt-3.5 등)

# OpenAI API

## RESTful API

- Postman을 활용하여 요청을 보내고 응답을 확인해봅시다

The screenshot displays the Postman interface for a POST request to the OpenAI API endpoint: `https://api.openai.com/v1/chat/completions`. The request is configured with the following details:

- Method:** POST
- Headers:** 11 headers are listed, with 9 hidden. The visible headers are:

Key	Value	Description
Authorization	Bearer <openai api-key>	
Content-Type	application/json	
- Body:** The body is set to raw JSON. The JSON payload is:

```
1 {  
2   "model": "gpt-4",  
3   "messages":  
4     [{"role": "user", "content": "let me know usage of gpt api"}]  
5 }  
6 }
```

Additional interface elements include a 'Send' button, 'Params', 'Authorization', 'Scripts', 'Settings', 'Cookies', and 'Beautify' options.

# OpenAI API

## OpenAI API

- OpenAI API Response

https://api.openai.com/v1/chat/completions

POST https://api.openai.com/v1/chat/completions

Params Authorization Headers (11) Body • Scripts Settings Cookies

Body Cookies (2) Headers (26) Test Results 200 OK • 15.02 s • 2.42 KB

```
{
  "id": "chatcmpl-ByG3ho8lGZpFn43aacavbYVxDhGn6",
  "object": "chat.completion",
  "created": 1753701109,
  "model": "gpt-4-0613",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "OpenAI's GPT-3 offers a General Language Understanding AI API, which developers can use to incorporate natural language processing capabilities into their applications. The API could be used for a variety of tasks such as:\n\n1. Content Generation and Enhancement: Computers can create meaningful and high-quality written content, generate creative writing, improve written content, animate characters for video games, etc.\n\n2. Language Translation: Used for translating content from one language to another.\n\n3. Programming Help: Debugging code, writing code, etc.\n\n4. Learning and Education: Support in solving problems, teaching concepts, helping with homework, etc.\n\n5. Personal Assistant Functions: Answering queries, scheduling or reminding tasks, general planning, etc.\n\n6. Sentiment Analysis: Analyzing and understanding emotions and sentiments present in the text.\n\nTo use it, an application sends a prompt to the GPT-3 API and it returns a text generated by the model. Here is a Python example using the OpenAI API:\n\npython\nimport openai\n\nopenai.api_key = 'your-api-key'\n\nresponse = openai.Completion.create(\n  model='text-davinci-003',\n  prompt='Translate the following English text to French: '{}'\n)\n\nprint(response.choices[0].text.strip())\n\nPlease note, usage of the API involves some considerations such as safety and bias mitigation. API users are expected to use the guidelines provided by OpenAI for safe and responsible use. Repeated or flagrant violations may result in restrictions on the use of the API.",
      },
      "logprobs": null,
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 10,
    "completion_tokens": 200,
    "total_tokens": 210
  }
}
```

# OpenAI API

## OpenAI API

- Token
  - 입출력 토큰 개수에 따른 API 사용료
  - 모델별로 가능한 입출력 토큰 개수 제한이 있음
  - 1토큰 = 약 한글 1자 / 영어 단어 0.75개
    - “안녕하세요” -> 약 5토큰
    - “This is a sentence.” -> 약 5~6토큰

# OpenAI API

## OpenAI API

- max\_tokens
  - 응답에서 최대 몇 토큰 생성할지 설정
- frequency\_penalty
  - 반복된 표현 억제
- presence\_penalty
  - 새로운 개념 유도



# OpenAI API

## OpenAI API

- Role
  - system : 모델의 행동 지침
  - user : 사용자의 질문
  - assistant : 모델의 이전 응답

# OpenAI API

## OpenAI API

- Temperature
  - 모델의 창의성
  - 0 : 매우 안정적, 반복적인 답변
  - 1.0 : 매우 창의적이고 유연한 답변
  - 0.7 : 일반적으로 가장 많이 사용

# OpenAI API

## OpenAI API

- 파이썬 라이브러리를 제공. 쉽게 API 호출 가능.

```
3 from openai import OpenAI
4
5 with open('api-key', 'r') as f:
6     API_KEY = f.read().strip()
7
8 client = OpenAI(api_key=API_KEY)
9
10 completion = client.chat.completions.create(
11     model="gpt-4.1",
12     messages=[
13         {"role": "developer", "content": "You are a helpful assistant."},
14         {"role": "user", "content": "Hello!"}
15     ]
16 )
17
18 print(completion.choices[0].message)
19
```

```
19 import requests
20
21 url = "https://api.openai.com/v1/chat/completions"
22 headers = {
23     "Authorization": f"Bearer {API_KEY}",
24     "Content-Type": "application/json"
25 }
26 data = {
27     "model": "gpt-4.1",
28     "messages": [
29         {"role": "developer", "content": "You are a helpful assistant."},
30         {"role": "user", "content": "Hello!"}
31     ]
32 }
33
34 response = requests.post(url, headers=headers, json=data)
35
36 if response.status_code == 200:
37     result = response.json()
38     print(result)
39
```

# OpenAI API

## OpenAI API

```
1 from openai import OpenAI
2
3 with open('api-key', 'r') as f:
4     API_KEY = f.read().strip()
5
6 client = OpenAI(api_key=API_KEY)
7
8 completion = client.chat.completions.create(
9     model="gpt-4",
10    messages=[
11        {"role": "system", "content": "You are a helpful assistant."},
12        {"role": "user", "content": "Tell me a story about a dragon."}
13    ],
14    temperature=0.8,
15    top_p=0.95,
16    frequency_penalty=0.3,
17    presence_penalty=0.6,
18    max_tokens=300
19 )
20
21 print(completion.choices[0].message.content)
```

~

**`/v1/chat/completion`**

# OpenAI API

## /v1/chat/completion

- 사용자 프롬프트(지시문)을 바탕으로 자연어 텍스트를 생성
- 주요 파라미터
  - model : 사용할 GPT 모델 지정
  - messages : 대화 내용
  - max\_completion\_tokens : 최대 출력 토큰 수
  - store : 응답을 저장할지 (true로 설정해야 나중에 조회 가능)

# OpenAI API

## /v1/chat/completion

←

→

↺

platform.openai.com/docs/api-reference/chat/create

🔍

☆

📱

소연

업데이트 완료

⋮

P

Personal

Default project

Dashboard

Docs

API reference

소연

Q Search

🔍 K

Responses

Responses

Create a model response

Get a model response

Delete a model response

Cancel a response

List input items

The response object

The input item list

Streaming

Chat Completions

Chat Completions

Create chat completion

Get chat completion

Get chat messages

List Chat Completions

Update chat completion

Delete chat completion

The chat completion

Create chat completion

POST https://api.openai.com/v1/chat/completions

Starting a new project? We recommend trying Responses to take advantage of the latest OpenAI platform features. Compare Chat Completions with Responses.

Creates a model response for the given chat conversation. Learn more in the [text generation](#), [vision](#), and [audio](#) guides. Parameter support can differ depending on the model used to generate the response, particularly for newer reasoning models. Parameters that are only supported for reasoning models are noted below. For the current state of unsupported parameters in reasoning models, [refer to the reasoning guide](#).

Request body

messages array Required

A list of messages comprising the conversation so far. Depending on the [model](#) you use, different message types (modalities) are supported, like [text](#), [images](#), and [audio](#).

Show possible types

model string Required

Model ID used to generate the response, like `gpt-4o` or `o3`. OpenAI offers a wide range of models with different capabilities, performance characteristics, and price points. Refer to the [model guide](#) to browse and compare available models.

audio object or null Optional

Parameters for audio output. Required when audio output is requested with `modalities: ["audio"]`. [Learn more](#).

Show properties

frequency\_penalty number or null Optional Defaults to 0

Default

Image input

Streaming

Functions

Logprobs

Example request

gpt-4.1

python

1 from openai import OpenAI

2 client = OpenAI()

3

4 completion = client.chat.completions.create(

5 model="gpt-4.1",

6 messages=[

7 {"role": "developer", "content": "You are a hel

8 {"role": "user", "content": "Hello!"}

9 ]

10 )

11

12 print(completion.choices[0].message)

Response

1 {

2 "id": "chatcmpl-B9MBs8Cjcv0U2jLn4n570S5qMJKcT",

3 "object": "chat.completion",

4 "created": 1741569952,

5 "model": "gpt-4.1-2025-04-14",

6 "choices": [

7 {

8 "index": 0,

9 "message": {

10 "role": "assistant",

11 "content": "Hello! How can I assist you to

12 "refusal": null,

13 "annotations": []

14 },

15 "logprobs": null,

16 "finish\_reason": "stop"

<https://platform.openai.com/docs/api-reference/chat/create>

# ChatGPT Chatbot



# OpenAI API

## Chatbot

- 1일차에 구현한 채팅 서버에 챗봇을 적용하여 봅시다.
  - “@chatbot <메세지>” 를 입력하면 챗봇이 응답.
  - /v1/chat/completions api를 활용

# OpenAI API

## Chatbot

- 1. 사용자 메시지로 부터 @chatbot이 붙은 메시지인지 확인
- 2. 입력받은 사용자 메시지로 gpt api에 POST 요청
- 3. gpt api의 응답으로 부터 메시지만 꺼내와 client에 전송(broadcast)

# OpenAI API Chatbot

- 1. 사용자 메시지로 부터 @chatbot이 붙은 메시지인지 확인

```
--
87 @app.websocket("/ws/{nickname}")
88 async def websocket_endpoint(websocket: WebSocket, nickname: str):
89     await manager.connect(websocket, nickname)
90     try:
91         while True:
92
93             raw_data = await websocket.receive_text()
94             try:
95                 data = json.loads(raw_data)
96             except json.JSONDecodeError:
97                 print("Invalid JSON:", raw_data)
98                 continue
99
100             timestamp = datetime.utcnow()
101             data["timestamp"] = timestamp.isoformat()
102
103             messages_collection.insert_one({
104                 "nickname": data.get("nickname", nickname),
105                 "role": "user",
106                 "message": data.get("message", ""),
107                 "timestamp": timestamp
108             })
109
110             msg_type = data.get("type")
111             if msg_type == "text":
112                 await manager.broadcast(data)
113                 text = data.get("message", "")
114                 if text.startswith("@chatbot"):
115                     asyncio.create_task(handle_chatbot(text, nickname))
116
```

# OpenAI API

## Chatbot

- 2. 입력받은 사용자 메시지로 gpt api에 POST 요청

```
10 from openai import AsyncOpenAI
11 import asyncio
12
13 app = FastAPI()
14
15 with open('api-key', 'r') as f:
16     API_KEY = f.read().strip()
17
18 client = AsyncOpenAI(api_key=API_KEY)
19
```

```
52 async def chatbot_response(message: str) -> str:
53     completion = await client.chat.completions.create(
54         model="gpt-4.1",
55         messages=[
56             {"role": "system", "content": "You are a helpful assistant"},
57             {"role": "user", "content": message}
58         ]
59     )
60     return completion.choices[0].message.content
61
```

# OpenAI API

## Chatbot

- 3. gpt api의 응답으로 부터 메시지만 꺼내와 client에 전송(broadcast)

```
--
69 async def handle_chatbot(data, nickname):
70     query = data.strip()[len("@chatbot"):].strip()
71     reply = await chatbot_response(query)
72
73
74     await manager.broadcast({
75         "type": "text",
76         "nickname": "chatbot",
77         "message": reply,
78         "timestamp": datetime.utcnow().isoformat()
79     })
--
```

# OpenAI API Chatbot

- 결과

## Chatting

test

test [2025-07-28T14:01:24.864960] : @chatbot 안녕??

chatbot [2025-07-28T14:01:26.468195] : 안녕하세요! 😊 무엇을 도와드릴까요?

Input Messages

Send

채팅 내용을 기억하는 Chatbot

# OpenAI API

## Chatbot

- /v1/chat/completions api
  - 이전 대화를 기억하지 못함
    - store 파라미터의 경우 이전 응답을 기억만 하고 새로운 대화에 적용하지는 않음
  - api에 전달하지 않은 일반 채팅 대화 또한 당연히 기억하지 못함.
- 이전 대화를 기억하는 Chatbot을 만드려면?
  - 이전 대화를 database에 저장하고 있다가 api에 함께 보내기



# OpenAI API

## Chatbot

- 1. gpt api 응답 DB에 저장
- 2. gpt api에 요청을 보낼 때, DB에 저장되어 있는 메시지(사용자 메시지 + 지피티 응답) n개 꺼내어 함께 보내기.

# OpenAI API Chatbot

- 결과

## Chatting

s  Enter

s [2025-07-28T14:27:02.831842] : @chatbot 지피티 하이

chatbot [2025-07-28T14:27:04.274315] : 안녕하세요! 🖐️ 지피티와 대화해주셔서 반가워요. 오늘 어떤 이야기를 나눠볼까요? 도움이 필요한 게 있으면 언제든 말해 주세요! 😊

s [2025-07-28T14:27:08.487054] : @chatbot 내 이름은 소연이야

chatbot [2025-07-28T14:27:09.703145] : 안녕하세요, 소연이! 만나서 정말 반가워요 😊 앞으로 궁금한 게 있거나 도움이 필요하면 언제든 편하게 말씀해 주세요. 오늘은 어떤 이야기를 나눠볼까요?

s [2025-07-28T14:27:14.720514] : @chatbot 내이름이 뭐라고?

chatbot [2025-07-28T14:27:16.148089] : 네 이름은 소연이야! 😊 소연아, 앞으로 궁금한 게 있으면 언제든지 물어봐 줘!

s [2025-07-28T14:27:37.883704] : 사실 내이름은 민서 인데

s [2025-07-28T14:27:43.912108] : @chatbot 내이름이 뭐게?

chatbot [2025-07-28T14:27:45.625163] : 방금 “사실 내이름은 민서 인데”라고 말해준 걸로 봐서, 네 이름은 민서인 것 같아! 민서야, 만나서 반가워 😊

Input Messages

Send

# 챗봇과의 이전 대화를 기억하는 Chatbot

# OpenAI API

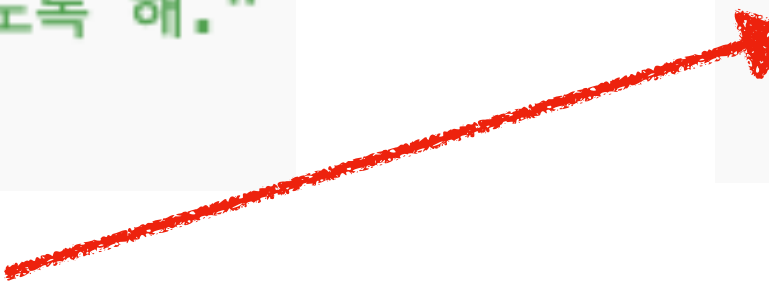
## Chatbot

- /v1/chat/completions api 대신 /v1/responses api를 사용
- /v1/responses api는 previous\_response\_id를 지정 가능

```
{  
  "model": "gpt-4.1",  
  "input": "내 이름은 민수야.",  
  "instructions": "대화를 이어가도록 해."  
}
```

```
{  
  "model": "gpt-4.1",  
  "input": "내 이름 기억나?",  
  "previous_response_id": "resp_abc123"  
}
```

response\_id = "resp\_abc123"



# OpenAI API

## Chatbot

- previous\_response\_id가 끊기지 않도록 체이닝

resp\_001 ← 최초 요청

resp\_002 ← previous\_response\_id = resp\_001

resp\_003 ← previous\_response\_id = resp\_002

...

resp\_010 ← previous\_response\_id = resp\_009

# OpenAI API Chatbot

- responses api

Personal

Default project

Dashboard

Docs

API reference

Search

K

API Reference

Introduction

Authentication

Debugging requests

Backward compatibility

Responses

Responses

Create a model response

Get a model response

Delete a model response

Cancel a response

List input items

The response object

The input item list

Streaming

Chat Completions

Chat Completions

Streaming

Webhooks

Webhook Events

Create a model response

POST https://api.openai.com/v1/responses

Creates a model response. Provide `text` or `image` inputs to generate `text` or `JSON` outputs. Have the model call your own `custom code` or use built-in `tools` like `web search` or `file search` to use your own data as input for the model's response.

Request body

**background**

boolean or null

Optional

Defaults to false

Whether to run the model response in the background. [Learn more.](#)

**include**

array or null

Optional

Specify additional output data to include in the model response. Currently supported values are:

`code_interpreter_call.outputs`

: Includes the outputs of python code execution in code interpreter tool call items.

`computer_call_output.output.image_url`

: Include image urls from the computer call output.

`file_search_call.results`

: Include the search results of the file search tool call.

`message.input_image.image_url`

: Include image urls from the input message.

`message.output_text.logprobs`

: Include logprobs with assistant messages.

`reasoning.encrypted_content`

: Includes an encrypted version of reasoning tokens in reasoning item outputs. This enables reasoning items to be used in multi-turn conversations when using the Responses API statelessly (like when the `store` parameter is set to `false`, or when an organization is enrolled in the zero data retention program).

**input**

string or array

Optional

Text, image, or file inputs to the model, used to generate a response.

Learn more:

[Text inputs and outputs](#)

Image inputs

Text input

Image input

File input

Web search

File search

Example request

python

```
1 from openai import OpenAI
2
3 client = OpenAI()
4
5 response = client.responses.create(
6     model="gpt-4.1",
7     input="Tell me a three sentence bedtime story abo
8 )
9
10 print(response)
```

Response

```
1 {
2   "id": "resp_67ccd2bed1ec8190b14f964abc0542670bb",
3   "object": "response",
4   "created_at": 1741476542,
5   "status": "completed",
6   "error": null,
7   "incomplete_details": null,
8   "instructions": null,
9   "max_output_tokens": null,
10  "model": "gpt-4.1-2025-04-14",
11  "output": [
12    {
13      "type": "message",
14      "id": "msg_67ccd2bf17f0819081ff3bb2cf6508e6",
15      "status": "completed",
16      "role": "assistant",
17      "content": [
18        {
19          "type": "output_text",
20          "text": "In a peaceful grove beneath a"
```

# OpenAI API

## Chatbot

- 1. 가장 최근의 response\_id 기억, gpt api 호출 시 넘겨주기
- 2. previous\_response\_id가 끊기지 않도록 체이닝

```
51  
52 latest_response_id: str | None = None  
53
```

# OpenAI API

## Chatbot

- 결과

### Chatting

g

g [2025-07-28T14:51:44.905159] : @chatbot 안녕??

chatbot [2025-07-28T14:51:46.173693] : 안녕하세요! 어떻게 도와드릴까요?

g [2025-07-28T14:51:50.114395] : @chatbot 내이름은 소연이야

chatbot [2025-07-28T14:51:51.544596] : 소연님, 만나서 반가워요! 오늘 어떻게 도와드릴까요?

g [2025-07-28T14:51:56.375464] : @chatbot 내이름이 뭐라고?

chatbot [2025-07-28T14:51:58.503293] : 소연님이라고 하셨죠! 맞나요?

g [2025-07-28T14:52:03.471950] : 사실 나는 민선데!

g [2025-07-28T14:52:11.402613] : @chatbot 내이름이 머라고!!!!

chatbot [2025-07-28T14:52:12.742402] : 죄송해요, 소연님! 제가 도와드릴까요?

Input Messages

Send



# OpenAI API

## Chatbot

- 텍스트 생성 모델을 어떻게 활용할 수 있을까??
  - 대화 내용 요약
  - 번역기
  - 상담봇
  - 캐릭터봇
- 여러가지 @태그 를 만들어 봅시다.

# Gemini API Chatbot

# Gemini API

## Chatbot

- **Gemini API**는 Google DeepMind의 LLM Gemini 시리즈를 사용할 수 있게 해주는 API
- API KEY 발급
  - <https://aistudio.google.com/apikey>

# Gemini API

## Chatbot

```
1 from google import genai
2 from google.genai import types
3
4 with open('genai-api-key', 'r') as f:
5     API_KEY = f.read().strip()
6
7
8 client = genai.Client(api_key=API_KEY)
9
10 response = client.models.generate_content(
11     model="gemini-2.5-flash",
12     contents="Hi?"
13 )
14 print(response)
15
```

# Gemini API

## Chatbot

```
16
17 response = client.models.generate_content(
18     model="gemini-2.5-flash",
19     contents="Hi?",
20     config=types.GenerateContentConfig(
21         thinking_config=types.ThinkingConfig(thinking_budget=0) # Disables thinking
22     ),
23 )
24 print(response)
25
```