



나도! **스프링**으로 **인공지능**을 할 수 있다(인프1탄)

Spring  
AI

지식공유자 : 박매일

---

# 강의목차

- 01\_실습준비와 OpenAI Key 발급하기
- 02\_Spring AI 기초와 환경설정
- 03\_Spring AI를 이용한 OpenAI와 채팅하기
- 04\_DALL-E 3를 이용한 이미지 생성하기
- 05\_이미지 분석과 수학문제 풀기
- 06\_AI로 만드는 나만의 레시피
- 07\_음성을 텍스트로 변환 하기(STT)
- 08\_OpenAI로 만든 음성 서비스(TTS)
- 09\_사용자 친화형 검색 서비스(GPT+SQL)
- 10\_RAG기반 PDF내 문서 유사도 검색
- 11\_플롯 유사성 기반 영화 추천 시스템
- 12\_RAG기반 호텔 AI 챗봇 서비스

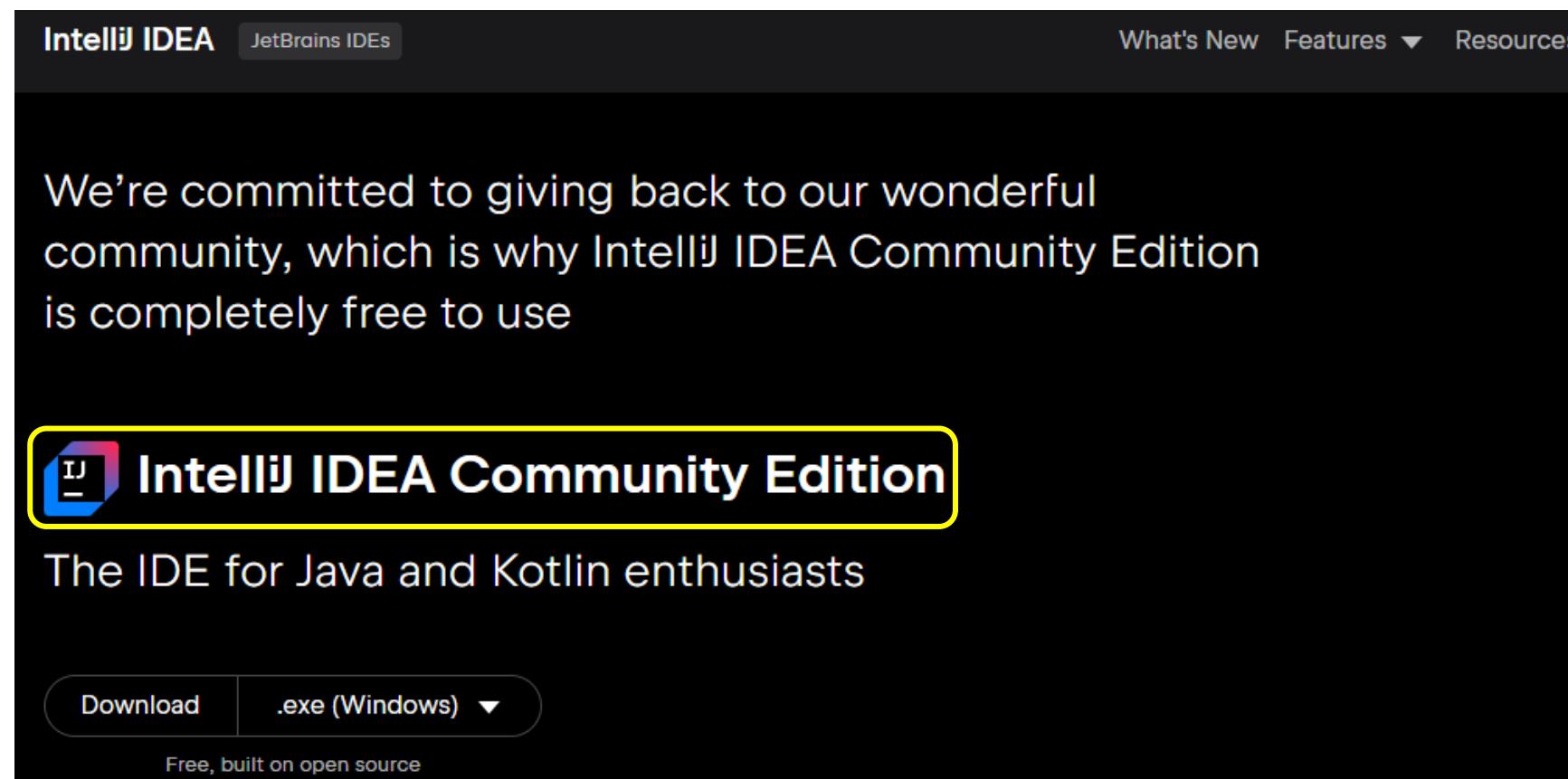
---

## 01\_실습준비와 OpenAI Key 발급하기

**목표:** OpenAI API 키 발급 방법을 학습하고 실습 환경에서 API를 테스트할 수 있도록 준비한다.

## 개발 TOOL 설치

<https://www.jetbrains.com/idea/download/?section=windows>



## Open AI Key 발급

<https://platform.openai.com/docs/overview>

A screenshot of a Google search results page. The search bar at the top contains the query "openai platform". Below the search bar, there are navigation links for "전체", "이미지", "뉴스", "쇼핑", "동영상", "웹", "지도", and "더보기". On the far right, there is a "도구" link. The main search results section shows a card for "OpenAI Platform" with the URL "https://platform.openai.com". Below the card, the title "OpenAI Platform" is displayed in purple, followed by a brief description: "Explore resources, tutorials, API docs, and dynamic examples to get the most out of OpenAI's developer platform."

# Open AI Key 발급

The screenshot shows the OpenAI API Keys page. At the top, there is a navigation bar with links to Playground, Dashboard, Docs, API reference, and a settings gear icon. To the right of the gear icon is a purple circular button with the Korean character '매일' (Daily) in white. A green bracket on the right side points from this button down to a red box around the 'Your profile' section of the user dropdown menu.

**Your profile**

Your details Security User API keys Legacy

**Name**  
The name associated with this account  
Park Maeil

**Email address**  
The email address associated with this account  
bmy19751975@gmail.com

**Phone number**  
The phone number associated with this account  
+821075041975

**Save**

Park Maeil  
bmy19751975@gmail.com  
Your profile  
Terms & policies  
Log out

# Open AI Key 발급

The screenshot shows the OpenAI Billing interface. On the left, a sidebar lists various sections: Personal, Default project, SETTINGS (Your profile), ORGANIZATION (General, API keys, Admin keys, Members, Projects), Billing (highlighted with a red box), Limits, Usage, Data controls, Verifications, PROJECT. The main area is titled "Billing" and "Pay as you go". It displays a credit balance of \$5.17. A note states "Auto recharge is off" and provides a link to "Enable auto recharge". Below this are buttons for "Add to credit balance" and "Cancel plan". A "Payment methods" button is highlighted with a yellow box and a red arrow points to it from the bottom. To the right, another "Billing" section shows a table of invoices with columns: INVOICE, STATUS, AMOUNT, CREATED. Two entries are listed: 6F89270A-0004 (Paid, \$6.60, 2024년 11월 5일 오후 10:25) and 6F89270A-0002 (Paid, \$5.50, 2024년 9월 29일 오후 10:26). The "Billing history" tab is selected.

INVOICE	STATUS	AMOUNT	CREATED
6F89270A-0004	Paid	\$6.60	2024년 11월 5일 오후 10:25
6F89270A-0002	Paid	\$5.50	2024년 9월 29일 오후 10:26

## Open AI Key 발급

P Personal / Default project

**SETTINGS**

Your profile

**ORGANIZATION**

General

**API keys**

Admin keys

Members

Projects

Billing

Limits

### Create new secret key

Owned by  
You Service account

This API key is tied to your user and can make requests against the selected project. If you are removed from the organization or project, this key will be disabled.

Name Optional

As an owner of this org  
Do not share your API key. It may automatically disable this key if it's shared with others.

Project  
Select project

To protect your account's security, OpenAI

Permissions  
 All  Restricted  Read Only

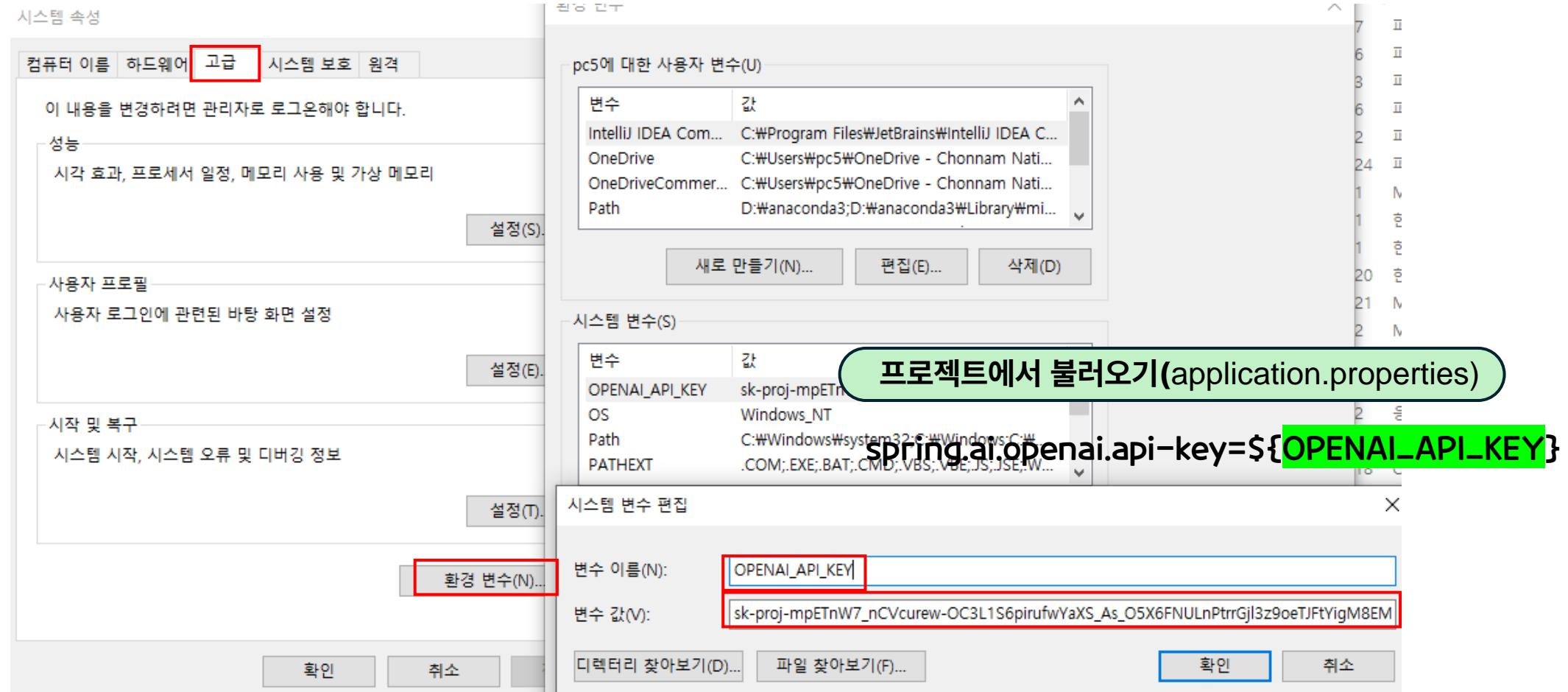
ACCESS CREATED BY  
Project Park Maeil

**+ Create new secret key**

**Cancel** **Create secret key**

**키 생성 후 반드시 복사(Copy) 해두자.**

# Open AI Key 세팅 - 1

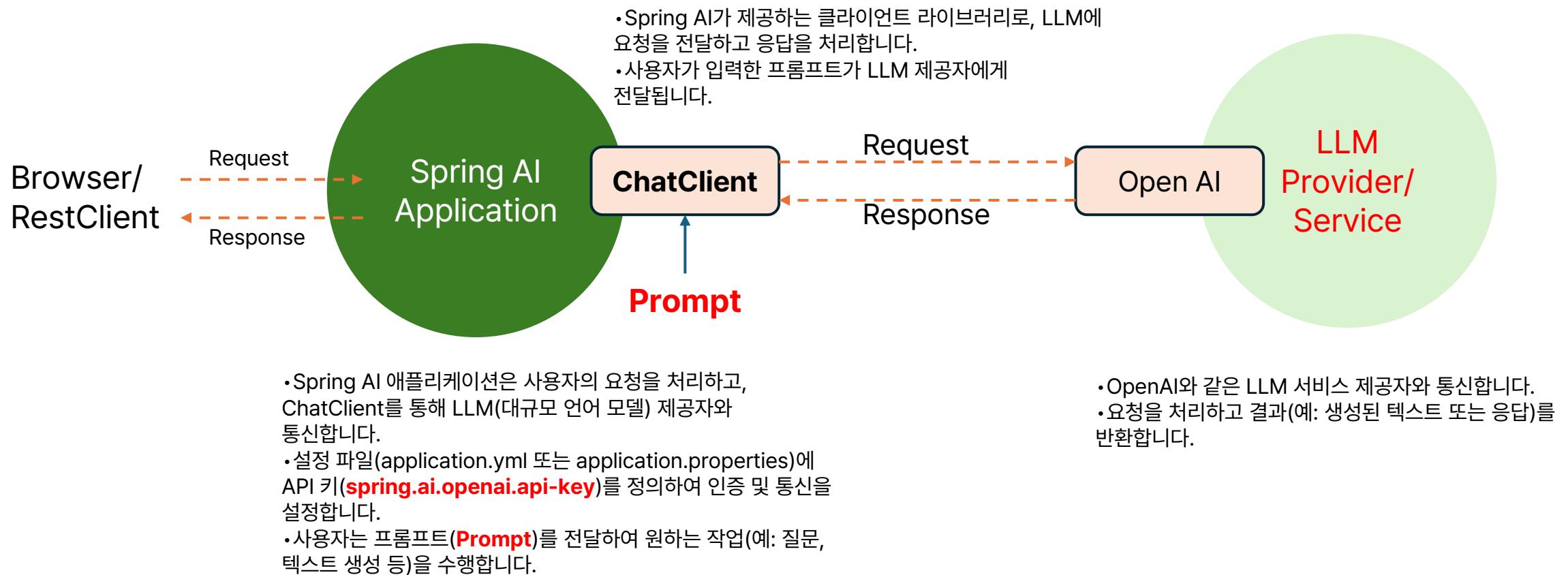


---

## 02\_Spring AI 기초와 환경설정

**목표:** Spring AI의 개념과 구조를 이해하고, 개발 환경을 설정하여 Spring AI 애플리케이션의 기초를 다진다.

## Spring AI의 Quick Start 예제



# Spring Boot Project 만들기

<https://start.spring.io/>

The screenshot shows the Spring Initializr web application interface. At the top, there's a navigation bar with a menu icon, the logo, and a search icon. Below the header, there are sections for Project, Language, and Dependencies.

**Project**: Maven (selected), Gradle - Groovy, Gradle - Kotlin.

**Language**: Java (selected), Kotlin, Groovy.

**Dependencies**: ADD DEPENDENCIES... CTRL + B

**Spring Web** WEB: Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

**OpenAI** AI: Spring AI support for ChatGPT, the AI language model and DALL-E, the Image generation model from OpenAI.

**Project Metadata**:

- Group: com.example
- Artifact: openai
- Name: openai
- Description: Demo project for Spring Boot
- Package name: com.example.openai
- Packaging: Jar (selected), War

Java: 23 (selected), 21, 17.

A file explorer window is shown at the bottom right, displaying a directory structure: 내 PC > 강의영상 (D:) > springai. Inside the 'springai' folder, there is an 'openai' folder and an 'openai.zip' file, which is highlighted with a blue selection bar.

## Project 실행 Open AI Key 설정 에러



## 1. application.properties에서 Key 불어오기

The screenshot shows the IntelliJ IDEA interface with the project structure on the left and the code editor on the right.

In the code editor, the `application.properties` file is open, displaying the following content:

```
spring.application.name=springbasic
spring.ai.openai.api-key=${OPENAI_API_KEY}
```

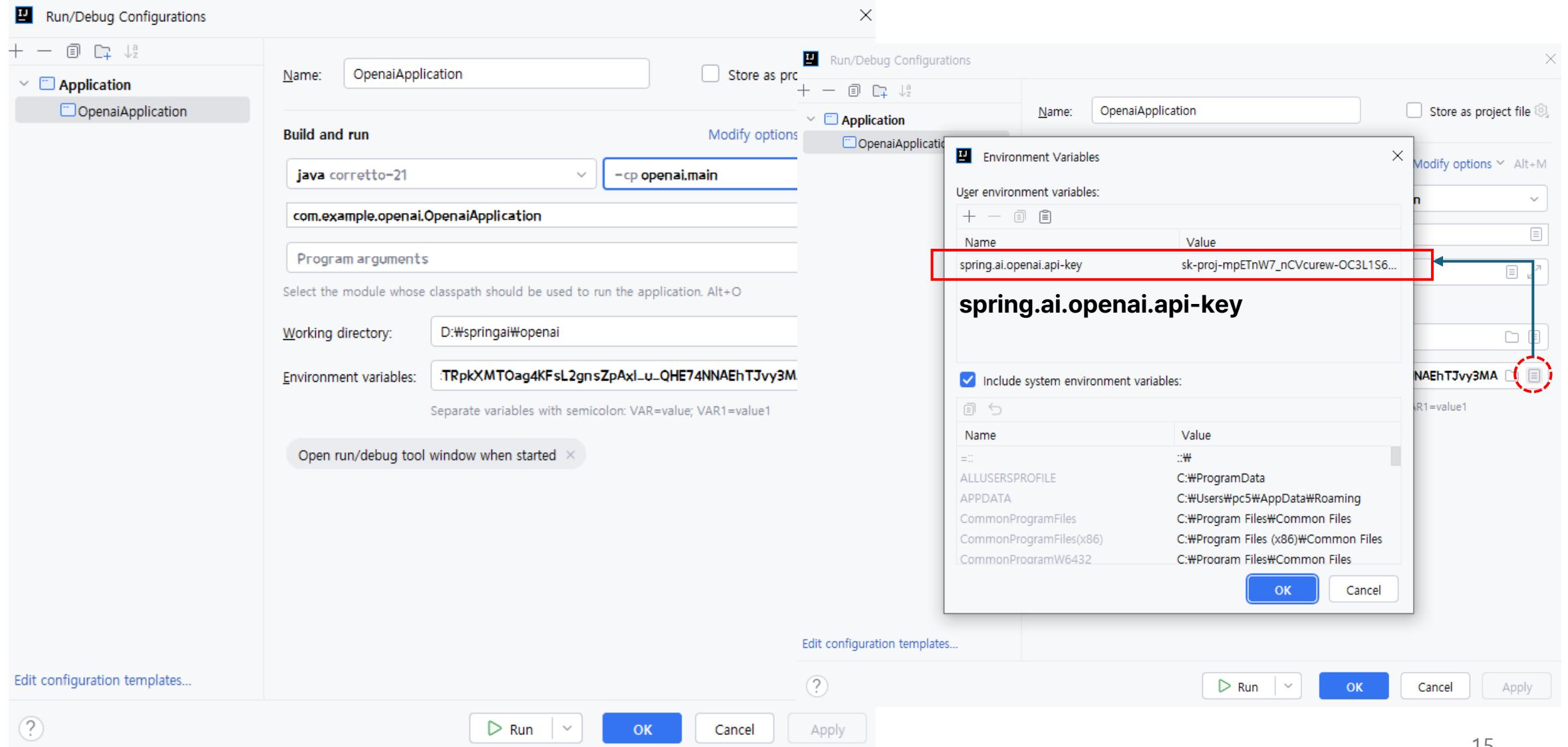
The line `spring.ai.openai.api-key=${OPENAI_API_KEY}` is highlighted with a red dashed box.

A system variable dialog box is displayed at the bottom right, titled "시스템 변수 편집". It contains the following information:

변수 이름(N):	OPENAI_API_KEY
변수 값(V):	:JTE8kCuBnJzJlY6Kv5xyMtP2tRS0vcTRpkXMT0ag4KFsl2gnsZpAxI_u_QHE74NNAEhTJvy3MA

Buttons at the bottom of the dialog include "확인" (Confirm) and "취소" (Cancel).

## 2. 프로젝트에서 Open AI Key 세팅



# Project 정상 실행

The screenshot shows a Java IDE interface with the following details:

- Project Structure:** The left sidebar displays the project structure under 'Project'. It includes a 'src' folder containing 'main' (with 'java' and 'com.example.openai' packages), 'resources' (with 'static' and 'templates' folders), and configuration files like 'application.properties' and 'build.gradle'. There are also 'test' and '.gitignore' files.
- Code Editor:** The main editor window shows the content of 'OpenaiApplication.java'. The code defines a main class 'OpenaiApplication' with a main method that runs the application using SpringApplication.
- Run Tab:** The 'Run' tab is selected, showing a run configuration for 'openai [OpenaiApplication.main()]'.
- Output Window:** The bottom pane shows the terminal output for the application's execution. It starts with a decorative ASCII art banner and then displays Spring Boot startup logs. The log concludes with the message 'Started OpenaiApplication in 3.99 seconds (process 16)'.

# API Reference

<https://docs.spring.io/spring-ai/reference/>

<https://platform.openai.com/docs/overview>

The screenshot shows the Spring AI project website at [https://docs.spring.io/spring-ai/reference/](#). The top navigation bar includes links for "Why Spring", "Learn", and "Projects". The left sidebar has a "Spring AI" header with a "1.0.0-SNAPSHOT" badge, a search bar, and a "Overview" section selected. Other sections in the sidebar include "AI Concepts", "Getting Started", "Chat Client API", "AI Models", "Vector Databases", "Observability", "Prompts", "Structured Output", "Function Calling", "Multimodality", "ETL Pipeline", "AI Model Evaluation", "Model Context Protocol (MCP)", "Service Connections", "Contribution Guidelines", and "Appendices". The main content area displays the "Spring AI / Overview" page, featuring the "spring AI" logo, a brief description of the project's goal to streamline AI development, and a note about connecting enterprise data and APIs to AI models. A diagram at the bottom illustrates the integration between an application, data, APIs, and generative AI.

The Spring AI project aims to streamline the development of applications that incorporate artificial intelligence functionality without unnecessary complexity.

The project draws inspiration from notable Python projects, such as LangChain and LlamaIndex, but Spring AI is not a direct port of those projects. The project was founded with the belief that the next wave of Generative AI applications will not be only for Python developers but will be ubiquitous across many programming languages.

**NOTE**

Spring AI addresses the fundamental challenge of AI integration: Connecting your enterprise Data and APIs with the AI Models .

The diagram shows a central yellow box containing a brain icon and the text "Generative AI". To its left is another yellow box labeled "Application" containing "Your Data" and "Your APIs". Two green arrows point from the "Application" box to the "Generative AI" box: one arrow points upwards labeled "To There" and another arrow points downwards labeled "To Here".

---

## What is an AI Model?(AI 모델이란?)

- AI 모델은 데이터를 통해 특정 작업을 수행할 수 있도록 학습된 수학적 표현 또는 프로그램입니다.
- 데이터를 분석하여 패턴을 인식하고, 이를 바탕으로 주어진 작업을 처리하는 방식으로 동작합니다.
- 인간이 학습하고 연습을 통해 기술을 익히듯, AI 모델도 예제를 학습하여 작업을 수행하는 "똑똑한 시스템"으로 볼 수 있습니다.

AI 모델은 데이터를 통해 학습된 문제 해결 도구입니다. 예를 들어, 이미지 분류, 자연어 처리, 음성 인식 등 다양한 작업에 사용될 수 있습니다.

- ▼ AI Models
  - Chat Models
  - Embedding Models
  - Image Models
  - Audio Models
  - Moderation Models

---

## GPT와 언어 모델(GPT and Language Models)

### 1. What is GPT? (GPT란 무엇인가?)

- GPT(Generative Pre-trained Transformer)는 OpenAI에서 개발한 언어 모델로, 자연어 처리(NLP) 작업에 사용됩니다.
- 사람처럼 텍스트를 생성하거나 이해할 수 있도록 훈련된 모델입니다.
- 기본적으로 대규모 텍스트 데이터를 학습하여 언어의 문맥과 의미를 이해합니다.

### 2. How GPT Works (GPT는 어떻게 작동하는가?)

- GPT는 Transformer라는 신경망 아키텍처를 기반으로 합니다.
- 입력된 텍스트를 토큰 단위로 나누고, 이를 바탕으로 문맥 정보를 학습하여 다음에 올 단어를 예측합니다.
- 사전 훈련(pre-training)과 이후 특정 작업에 맞춘 미세 조정(fine-tuning) 과정을 통해 높은 성능을 발휘합니다.

### 3. Applications of GPT (GPT의 응용 분야)

- **챗봇**: 사람과 자연스러운 대화를 할 수 있는 AI 기반 대화형 시스템.
- **콘텐츠 생성**: 글쓰기, 마케팅 콘텐츠, 보고서 작성 등에 활용.
- **언어 번역**: 다른 언어 간의 자연스러운 번역 제공.
- **문제 해결**: 질문 답변(Q&A), 요약, 코딩 지원 등 다양한 작업.

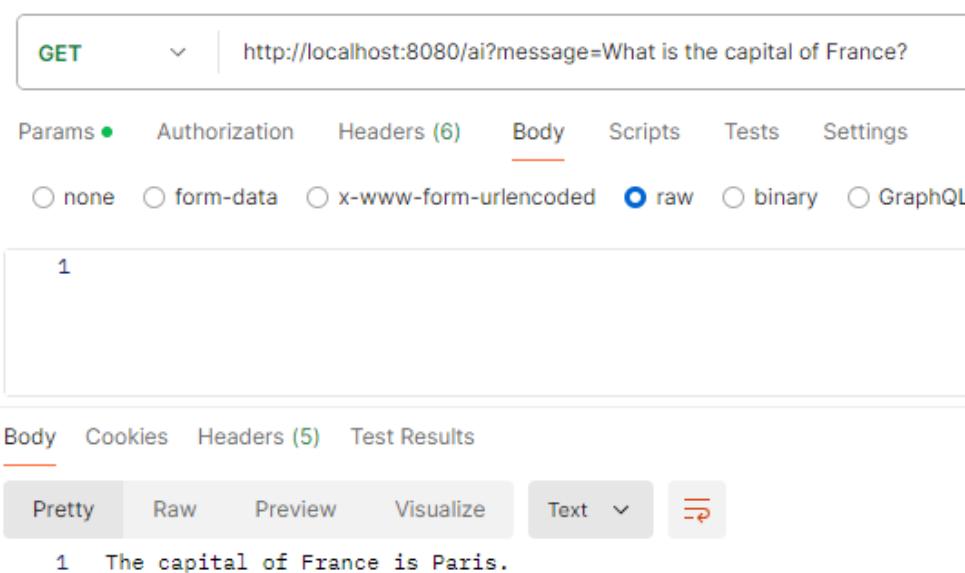
## What is an AI Model?(AI 모델이란?)

카테고리	설명	사용 예시
<b>Chat Models</b>	자연어 처리 및 대화 기능을 제공하는 LLM 기반 모델입니다.	AI 챗봇, Q&A 시스템
<b>Embedding Models</b>	텍스트를 벡터로 변환하여 유사도 검색 및 문서 검색에 사용됩니다.	문서 유사도 비교, RAG 서비스
<b>Image Models</b>	주어진 프롬프트로 이미지 생성 및 수정 기능을 제공합니다.	이미지 생성, 이미지 편집
<b>Audio Models</b>	음성을 텍스트로 변환(STT)하거나 텍스트를 음성으로 변환(TTS)합니다.	음성 인식, 음성 합성
<b>Moderation Models</b>	콘텐츠 검열 및 부적절한 텍스트/이미지를 감지하는 모델입니다.	텍스트 필터링, 부적절 콘텐츠 차단

## Chat Client API

Spring AI의 **ChatClient API**는 **AI 모델과의 통신**을 위한 **유창한(Fluent) API**를 제공합니다. 이 API는 **동기(synchronous)** 및 **스트리밍(streaming)** 프로그래밍 모델을 모두 지원하여 다양한 방식으로 사용할 수 있습니다.

# 예시: localhost:8080/**ai?message=What is the capital of France?**  
# 출력: "The capital of France is Paris."



**@RestController**

```
public class MyController {
    private final ChatClient chatClient;

    // 자동 구성된 ChatClient.Builder를 생성자 주입
    public MyController(ChatClient.Builder chatClientBuilder) {
        this.chatClient = chatClientBuilder.build();
    }
}
```

// 사용자가 입력한 message에 대한 AI 응답을 반환하는 엔드포인트

**@GetMapping("/ai")**

```
String generation(String message) {
    return this.chatClient.prompt() // 프롬프트 생성
        .user(message) // 사용자 메시지 설정
        .call() // AI 모델에 요청 전송
        .content(); // AI 모델의 응답을 문자열로 반환
}
```

---

## AI 모델의 다양한 역할(프롬프트 구조)

### 구성 요소

#### 1. System Role Example:

1. AI가 수행할 수 있는 다양한 역할의 예시를 나열합니다.
2. 예: Chatbot, Content Creator, Translator, Code Assistant 등.

#### 2. System Message Prompt:

1. AI의 역할을 정의하는 메시지입니다.
2. AI가 어떤 방식으로 행동해야 하는지 명확히 지시합니다.
3. 예: "You are a translator. Translate text between different languages."  
→ 당신은 번역가입니다. 다양한 언어 사이에서 텍스트를 번역하세요

#### 3. User Message Prompt:

1. 사용자가 AI에 요청할 실제 작업의 예입니다.
2. AI의 정의된 역할에 따라 실행해야 할 요청을 보여줍니다.
3. 예: "Translate the following English text to Spanish: 'Hello, how are you?'"  
→ 다음 영어 텍스트를 스페인어로 번역하세요: '안녕하세요, 어떻게 지내세요?'

# System Message

```

@Configuration
public class AppConfig {

    @Bean
    public ChatClient chatClientBuilder(ChatClient.Builder chatClientBuilder){
        // 당신은 교육 튜터입니다. 개념을 명확하고 간단하게 설명하세요
        return chatClientBuilder.defaultSystem("당신은 교육 튜터입니다. 개념을 명확하고 간단하게 설명하세요").build();
    }
}

@Service
public class ChatService {

    @Autowired
    private ChatClient chatClient;

    public String generateChatResponse2(){
        // 뉴턴의 운동 제2법칙을 간단하게 설명하세요.
        return chatClient.prompt()
            .user("뉴턴의 운동 제2법칙을 간단하게 설명하세요.")
            .call()
            .chatResponse()
            .getResult()
            .getOutput()
            .getContent();
    }
}

@RestController
public class ChatController {

    private final ChatService chatService;

    public ChatController(ChatService chatService) {
        this.chatService = chatService;
    }

    @GetMapping("/chat")
    public String chat(){
        return chatService.generateChatResponse2();
    }
}

```

**System Message**

**User Message**

## System Message



뉴턴의 운동 제2법칙은 물체의 가속도는 그 물체에 작용하는 힘과 질량에 따라 결정된다는 법칙입니다. 이것을 수식으로 표현하면  $F = ma$ 입니다. 여기서  $F$ 는 힘,  $m$ 은 질량,  $a$ 는 가속도입니다. 즉, 같은 힘이 작용할 때 질량이 큰 물체는 작은 물체보다 덜 가속되고, 같은 질량의 물체에 더 큰 힘이 작용하면 더 많이 가속됩니다. 간단히 말해, 힘은 질량과 가속도의 곱입니다.

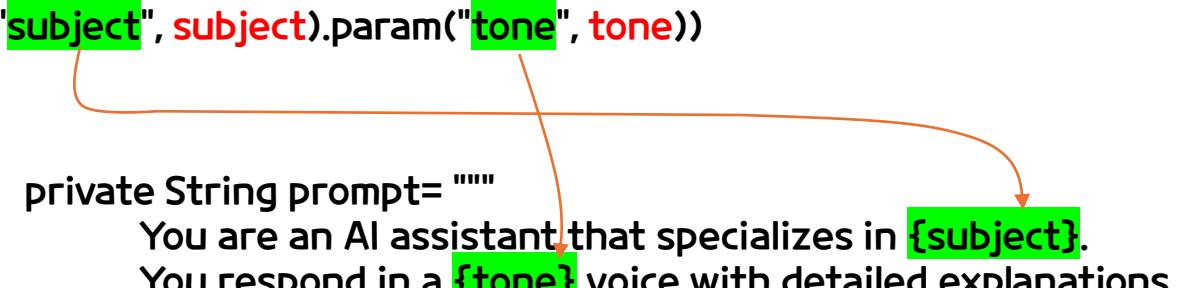
Default System Text with parameters : placeholder → { }

```
@Configuration  
public class AppConfig {  
  
    @Value("classpath:/prompt.txt")  
    private Resource resource;  
  
    private String prompt= """  
        You are an AI assistant that specializes in {subject}.  
        You respond in a {tone} voice with detailed explanations.  
    """;  
  
    @Bean  
    public ChatClient chatBuilder(ChatClient.Builder chatBuilder){  
        // 당신은 특정 {주제}에 특화된 AI 도우미입니다. 당신은 상세한 설명을 제공하는 {톤}과 음성으로 답변합니다.  
        return chatBuilder.defaultSystem("You are an AI assistant that specializes in {subject}." +  
            "You respond in a {tone} voice with detailed explanations.").build();  
    }  
}
```

Default System Text with parameters : placeholder → { }

```
@Service  
public class ChatService {  
  
    @Autowired  
    private ChatClient chatClient;  
  
    public String generateChatResponse(String subject, String tone, String message){  
        // 뉴턴의 운동 제2법칙을 간단하게 설명하세요.  
        return chatClient.prompt()  
            .user(message)  
            .system(sp->sp.param("subject", subject).param("tone", tone))  
            .call()  
            .chatResponse()  
            .getResult()  
            .getOutput()  
            .getContent();  
    }  
}
```

private String prompt= """  
 You are an AI assistant that specializes in {subject}.  
 You respond in a {tone} voice with detailed explanations.  
 """;



Default System Text with parameters : placeholder →{ }

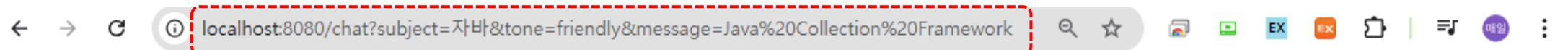
```
@RestController
public class ChatController {

    private final ChatService chatService;

    public ChatController(ChatService chatService) {
        this.chatService = chatService;
    }

    @GetMapping("/chat")
    public String chat(String subject, String tone, String message){
        return chatService.generateChatResponse(subject, tone, message);
    }
}
```

## Default System Text with parameters : placeholder →{ }



Java Collection Framework은 데이터 그룹을 저장하고 조작하기 위한 아키텍처를 제공하는 자바의 핵심 라이브러리입니다. 이 프레임워크는 다양한 인터페이스, 클래스 및 알고리즘을 포함하여 데이터 구조를 관리하고, 데이터를 효율적으로 처리할 수 있도록 돕습니다. ### 주요 인터페이스 1. **Collection**: 이 인터페이스는 기본적으로 데이터 그룹을 나타내며, 가장 상위에 위치한 인터페이스입니다. `List`, `Set`, `Queue` 같은 다른 인터페이스들이 이를 확장합니다. 2. **List**: 순서가 있는 요소의 컬렉션을 나타냅니다. 요소는 인덱스를 통해 접근할 수 있으며, 중복을 허용합니다. `ArrayList`, `LinkedList`, `Vector` 등이 구현체로 있습니다. 3. **Set**: 중복을 허용하지 않는 유일한 요소의 집합을 나타냅니다. `HashSet`, `LinkedHashSet`, `TreeSet` 등이 구현체로 있습니다. 4. **Queue**: 요소를 FIFO(First-In-First-Out) 방식으로 처리하는 컬렉션입니다. `LinkedList`, `PriorityQueue` 등이 구현체로 있습니다. 5. **Map**: 키-값 쌍으로 이루어진 컬렉션을 나타냅니다. `HashMap`, `LinkedHashMap`, `TreeMap`, `Hashtable` 등이 구현체로 있습니다. Map은 Collection의 하위 인터페이스가 아니지만, Java Collection Framework의 중요한 부분입니다. ### 주요 클래스 1. **ArrayList**: 크기가 가변적인 배열로 구현된 List입니다. 요소의 인덱스를 통해 접근이 빠릅니다. 2. **LinkedList**: 이중 연결 리스트로 구현된 List와 Queue의 기능을 모두 제공하는 클래스입니다. 요소의 삽입과 삭제가 빠릅니다. 3. **HashSet**: 해시 테이블을 사용하여 구현된 Set으로, 요소의 순서를 보장하지 않으며, 중복을 허용하지 않습니다. 4. **HashMap**: 해시 테이블을 사용하여 구현된 Map으로, 키-값 쌍의 저장을 도와줍니다. 순서를 보장하지 않습니다. 5. **TreeSet**: 정렬된 순서로 요소를 저장하는 Set입니다. `Comparable`이나 `Comparator`를 사용하여 요소를 정렬할 수 있습니다. 6. **TreeMap**: 키를 정렬된 순서로 저장하는 Map입니다. ### 알고리즘 Java Collection Framework는 데이터 조작을 위한 여러 알고리즘도 제공하며, `Collections` 클래스에 구현되어 있습니다. 예를 들어, 정렬, 검색, 교집합 등을 수행할 수 있습니다. ### 장점 - **일관성**: 모든 컬렉션 클래스가 공통의 인터페이스를 따르므로, 일관된 프로그래밍이 가능합니다. - **유연성**: 다양한 데이터 구조와 알고리즘을 제공하여, 특정 요구에 맞는 컬렉션을 쉽게 선택할 수 있습니다. - **성능**: 효율적인 데이터 처리 및 관리가 가능하도록 최적화된 알고리즘을 제공합니다. Java Collection Framework는 데이터 구조와 알고리즘을 쉽게 이해하고 사용할 수 있도록 도와주는 강력한 도구입니다. 이를 통해 개발자는 복잡한 데이터 구조를 직접 구현하지 않고도, 효율적인 프로그램을 작성할 수 있습니다.

---

## Returning a ChatResponse

ChatResponse는 Spring AI의 ChatClient API에서 AI 모델의 응답을 포맷하고 관리하기 위해 사용하는 데이터 구조

@RestController

```
@GetMapping("/chatjson")
public ChatResponse chatJson(@RequestParam("message") String message){
    return chatClient.prompt()
        .user(message)
        .call()
        .chatResponse();
}
```

## 응답받기 : ChatResponse(JSON)

```

@RestController
public class MyController {

    private final ChatClient chatClient;

    public MyController(ChatClient.Builder chatClientBuilder) {
        this.chatClient = chatClientBuilder.build();
    }

    // ChatResponse : JSON
    @GetMapping("/json")
    public ChatResponse chatJson(String message){
        return chatClient.prompt()
            .user(message)
            .call()
            .chatResponse();
    }
}
  
```

```

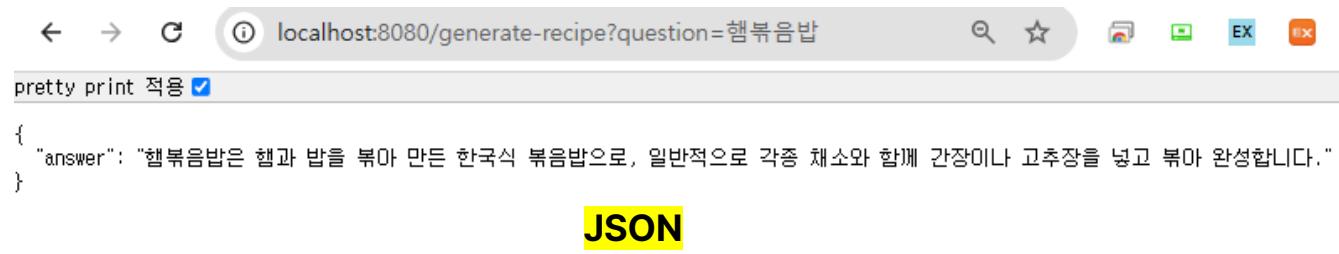
{
    "result": {
        "metadata": {
            "finishReason": "STOP",
            "contentFilterMetadata": null
        },
        "output": {
            "messageType": "ASSISTANT",
            "metadata": {
                "refusal": "",
                "finishReason": "STOP",
                "index": 0,
                "id": "chatcmpl-ATNQLSnTtVXrlbK4SqcpfQI4bQAet",
                "role": "ASSISTANT",
                "messageType": "ASSISTANT"
            },
            "toolCalls": [],
            "content": "Why don't scientists trust atoms? Because they make up everything!"
        }
    },
    "metadata": {
        "id": "chatcmpl-ATNQLSnTtVXrlbK4SqcpfQI4bQAet",
        "model": "gpt-4o-2024-08-06",
        "rateLimit": {
            "requestsLimit": 500,
            "requestsRemaining": 499,
            "tokensLimit": 30000,
            "tokensRemaining": 29973,
            "requestsReset": "PT0.12S",
            "tokensReset": "PT0.054S"
        },
        "usage": {
            "generationTokens": 12,
            "promptTokens": 16,
            "totalTokens": 28,
            "reasoningTokens": 0
        },
        "promptMetadata": [],
        "empty": false
    },
    "results": [
        {
            "metadata": {
                "finishReason": "STOP",
                "contentFilterMetadata": null
            },
            "output": {
                "messageType": "ASSISTANT",
                "metadata": {
                    "refusal": "",
                    "finishReason": "STOP",
                    "index": 0,
                    "id": "chatcmpl-ATNQLSnTtVXrlbK4SqcpfQI4bQAet",
                    "role": "ASSISTANT",
                    "messageType": "ASSISTANT"
                },
                "toolCalls": [],
                "content": "Why don't scientists trust atoms? Because they make up everything!"
            }
        }
    ]
}
  
```

## Returning an Entity

```
@RestController
public class RecipeController {

    @Autowired
    private ChatClient chatClient;

    @GetMapping("/generate-recipe")
    public Answer generateRecipe(String question) {
        return chatClient.prompt()
            .user(question)
            .call()
            .entity(Answer.class);
    }
}
```



```
public class Answer {
    private String answer;

    public Answer(String answer) {
        this.answer = answer;
    }

    public String getAnswer() {
        return answer;
    }

    public void setAnswer(String answer) {
        this.answer = answer;
    }

    @Override
    public String toString() {
        return "Answer{" +
            "answer=" + answer + 'W' +
            '}';
    }
}
```

## Prompt and Prompt template

```

@RestController
public class RecipeController {
    @Autowired
    private ChatClient chatClient;

    private final String recipeTemplate= """
        Answer for {foodname} for {question}?
        """;
}

@GetMapping("/generate-recipe")
public Answer generateRecipe(String foodName, String question) {
    return chatClient.prompt()
        .user(userSpec->userSpec.text(recipeTemplate)
            .param("foodname", foodName)
            .param("question", question)
        )
        .call()
        .entity(Answer.class);
}

```

question=주요 재료는 무엇인가요  
foodName=버거

localhost:8080/generate-recipe?question=%ED%95%91%EC%8A%A4%ED%8A%B8%ED%8E%88%ED%8A%A4%ED%8A%8C&foodName=%EB%8B%A8

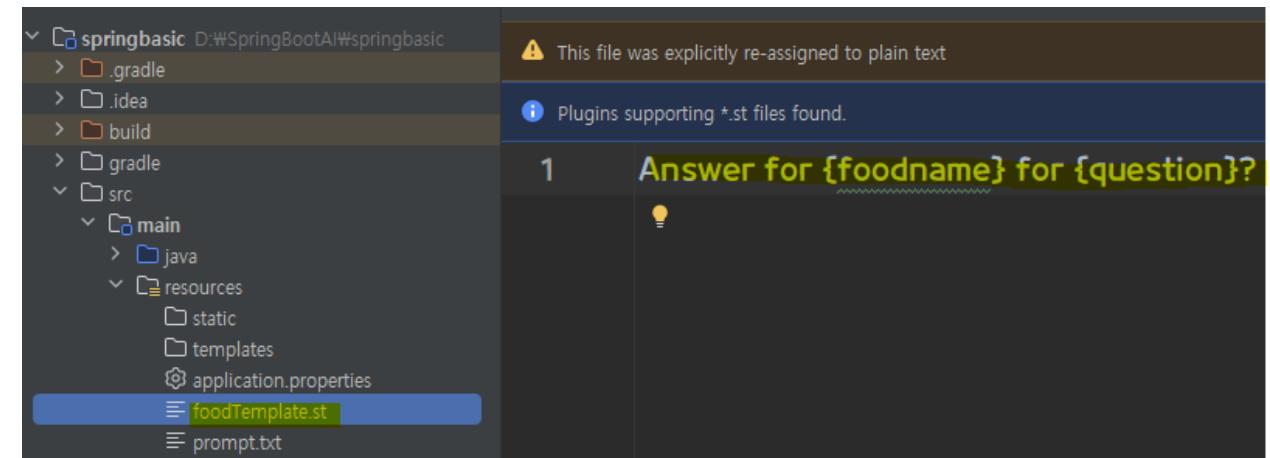
```
{
    "answer": "버거의 주요 재료는 다음과 같습니다: 빵(번), 쇠고기 패티, 치즈, 양상추, 토마토, 양파, 피클, 소스(예: 케첩, 마요네즈)."
}
```

## Resource template

```
@RestController
public class RecipeController {
    @Autowired
    private ChatClient chatClient;

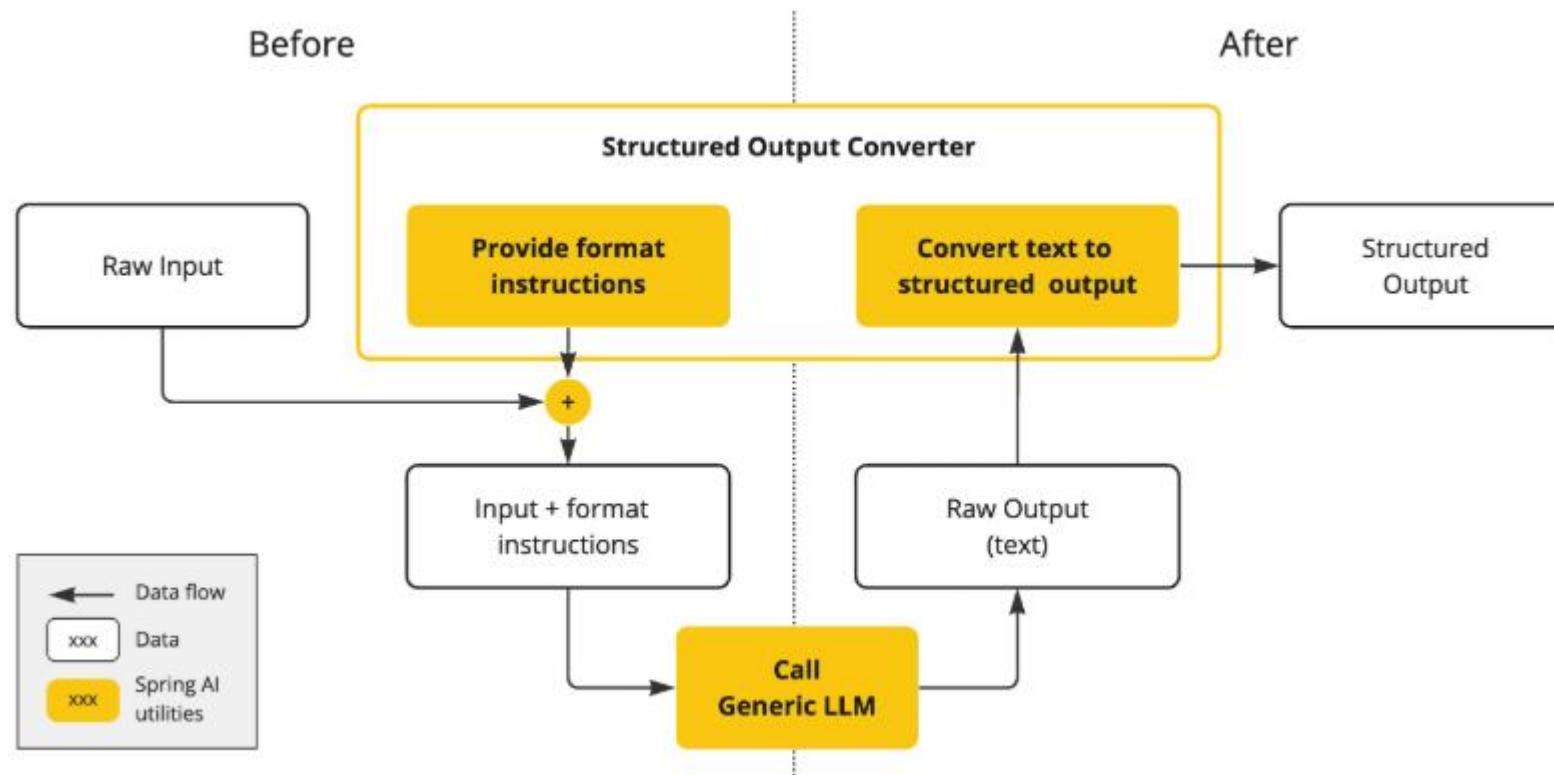
    @Value("classpath:/foodTemplate.st")
    private Resource foodResource;

    @GetMapping("/generate-recipe")
    public Answer generateRecipe(String foodName, String question) {
        return chatClient.prompt()
            .user(userSpec->userSpec.text(foodResource)
                .param("foodname", foodName)
                .param("question", question)
            )
            .call()
            .entity(Answer.class);
    }
}
```



## Structured Output

### Spring AI의 Structured Output Converter 아키텍처



## Structured Output

```

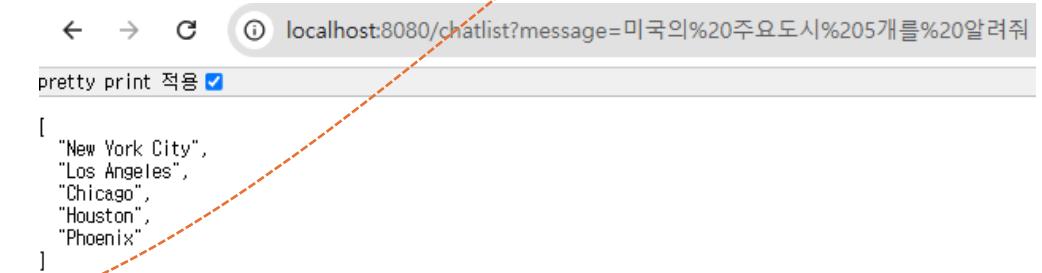
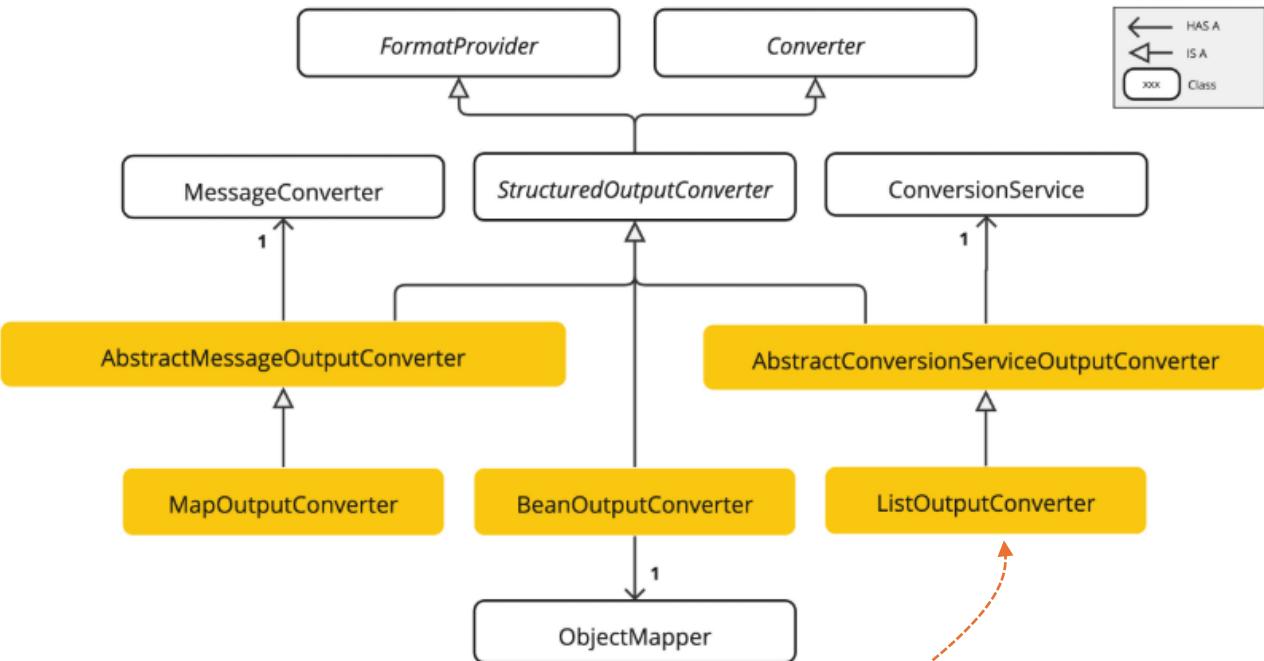
@RestController
public class ChatController {

    private ChatClient chatClient;

    public ChatController(ChatClient.Builder chatClient)
        this.chatClient= chatClient.build();
    }

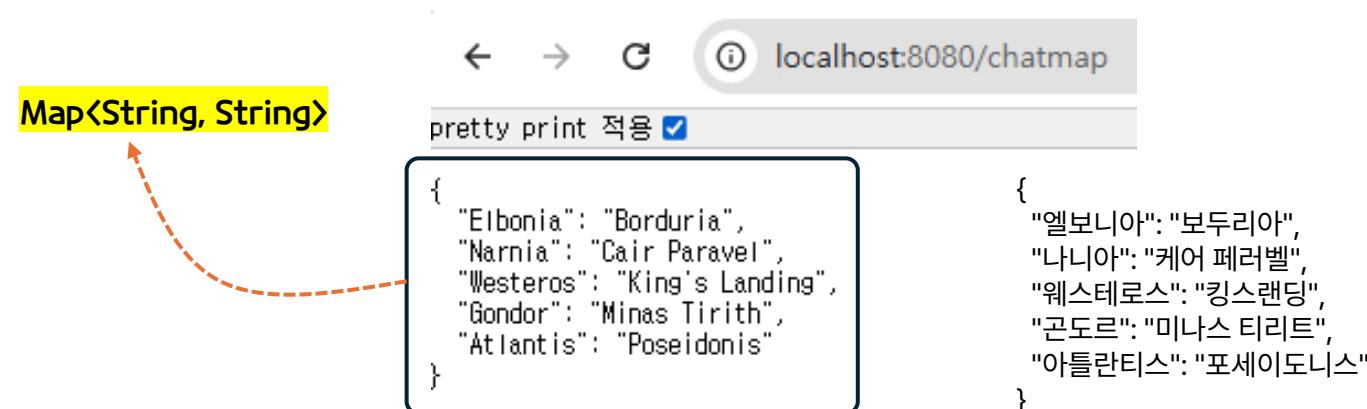
    @GetMapping("/chatlist")
    public List<String> chatList(@RequestParam("message") String message){
        return chatClient.prompt()
            .user(message)
            .call()
            .entity(new ListOutputConverter(new DefaultConversionService()));
    }
}

```



## Structured Output

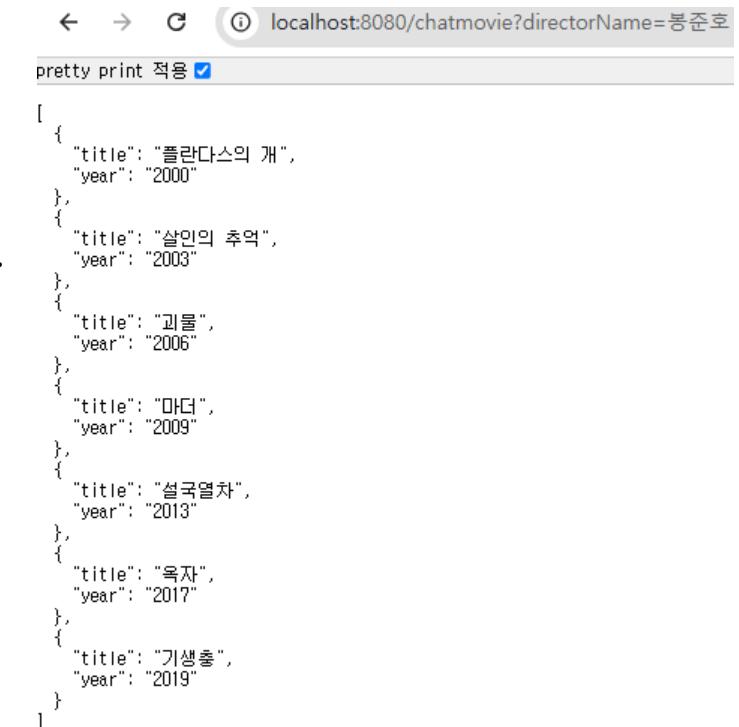
```
@GetMapping("/chatmap")
public Map<String, String> chatMap(@RequestParam(value = "message", defaultValue = "국가와 그 수도를 5개만 만들어줘?")  
String message){
    return chatClient.prompt()
        .user(message)
        .call()
        .entity(new ParameterizedTypeReference<Map<String, String>>() {});
}
```



## Structured Output

```
public record Movie(String title, String year) {  
}
```

```
@GetMapping("/chatmovie")  
public List<Movie> chatMovie(@RequestParam(value = "directorName") String directorName){  
    // "{directorName}가 감독한 영화 목록을 생성하세요. 감독이 알려지지 않은 경우, null을 반환하세요.  
    // 각 영화는 제목과 개봉 연도를 포함해야 합니다. {format}  
  
    String template= """  
        Generate a list of movies directed by {directorName}. If the director is unknown, return null.  
        한국 영화는 한글로 표기해줘.  
        Each movie should include a title and release year. {format}"  
        """;  
  
    List<Movie> movieList=chatClient.prompt()  
        .user(userSpec->userSpec.text(template)  
            .param("directorName", directorName)  
            .param("format", "json")  
        )  
        .call()  
        .entity(new ParameterizedTypeReference<List<Movie>>() {});  
  
    return movieList;  
}
```



---

## Spring AI Advisors

```
public String getResponse(String message){  
    return chatClient.prompt().user(message).call().content();  
}  
  
public void startChat(){  
    Scanner scanner=new Scanner(System.in);  
    System.out.println("Enter your message: ");  
    while(true){  
        String message=scanner.nextLine();  
        if(message.equals("exit")){  
            System.out.println("Exiting chat...");  
            break;  
        }  
        String response=getResponse(message);  
        System.out.println("Bot: "+ response);  
    }  
    scanner.close();  
}
```

---

## Spring AI Advisors

```
@Component
public class CommandLineAppStartupRunner implements CommandLineRunner {

    private final ChatService chatService;
    public CommandLineAppStartupRunner(ChatService chatService) {
        this.chatService = chatService;
    }

    @Override
    public void run(String... args) throws Exception {
        chatService.startChat();
    }
}

@Bean
public ChatClient chatBuilder(ChatClient.Builder chatBuilder){
    return chatBuilder.defaultAdvisors
        (new MessageChatMemoryAdvisor(new InMemoryChatMemory())).build();
}
```

---

**Github**

<https://github.com/parkmaeil/aispring.git>

---

## 03\_Spring AI를 이용한 OpenAI와 채팅하기

**목표:** Spring AI의 ChatClient를 사용하여 OpenAI의 GPT 모델과 채팅하는 기능을 구현한다.

---

Spring Boot + Spring AI

# OpenAI Asking ?

Spring AI

# Application UI

Ask AI

질문을 입력하세요:

인공지능에서 LLM이 뭐야

Ask

Ask with Options

LLM은 "Large Language Model"의 약자로, 대규모 언어 모델을 의미합니다. 이는 방대한 양의 텍스트 데이터를 학습하여 자연어를 이해하고 생성할 수 있는 인공지능 모델을 말합니다. LLM은 다양한 언어 작업, 예를 들어 번역, 요약, 질문 답변, 창의적인 글쓰기 등을 수행할 수 있습니다. 대표적인 LLM으로는 OpenAI의 GPT 시리즈, Google's BERT, 그리고 Facebook의 RoBERTa 등이 있습니다. 이러한 모델들은 수십억 개의 매개변수를 가지고 있어 복잡한 언어 패턴을 학습하고 인간과 유사한 수준의 언어 처리 능력을 보여줍니다.

# Application UI

## Ask AI

질문을 입력하세요:

질문 입력

Ask

Ask with Options

## Ask AI

질문을 입력하세요:

spring ai API를 만든 목적이 뭐야?

Ask

Ask with Options

Spring AI API는 주로 인공지능 및 머신러닝 기능을 애플리케이션에 쉽게 통합할 수 있도록 지원하기 위해 만들어졌습니다. 이를 통해 개발자들은 복잡한 AI 모델을 직접 구현하지 않고도, Spring 프레임워크를 사용하여 AI 기능을 손쉽게 활용할 수 있습니다. 이러한 API는 생산성을 높이고, 애플리케이션의 인텔리전스를 강화하며, 개발 시간을 단축시키는 데 목적이 있습니다. 구체적인 사용 사례로는 자연어 처리, 이미지 인식, 데이터 분석 등이 포함될 수 있습니다.

# 입력 폼

## Ask AI

질문을 입력하세요:

질문 입력

Ask

Ask with Options

```
<div class="container mt-5">
  <h2 class="text-center">Ask AI</h2>
  <div class="form-group">
    <label for="question">질문을 입력하세요:</label>
    <input type="text" id="question" class="form-control" placeholder="질문 입력">
  </div>
  <button class="btn btn-primary mt-2" onclick="getResponse()">Ask</button>
  <button class="btn btn-secondary mt-2" onclick="getResponseOptions()">Ask with Options</button>

  <div class="mt-4">
    <div id="response" class="alert alert-info" role="alert" style="display: none;"></div>
  </div>
</div>
```

# 질문입력 스크립트

```
<script>
    async function getResponse() {
        const question = document.getElementById('question').value;
        if (question.trim() === "") {
            alert('질문을 입력하세요');
            return;
        }

        const responseDiv = document.getElementById('response');
        responseDiv.innerText = '응답이 생성중입니다...';
        responseDiv.style.display = 'block';

        try {
            const response = await fetch('/ask?message=${encodeURIComponent(question)}');
            const result = await response.text();
            responseDiv.innerText = result;
        } catch (error) {
            console.error('Error:', error);
            responseDiv.innerText = '오류가 발생했습니다.';
        }
    }

    async function getResponseOptions() {
        const question = document.getElementById('question').value;
        if (question.trim() === "") {
            alert('질문을 입력하세요');
            return;
        }

        const responseDiv = document.getElementById('response');
        responseDiv.innerText = '응답이 생성중입니다...';
        responseDiv.style.display = 'block';

        try {
            const response = await fetch('/ask-ai?message=${encodeURIComponent(question)}');
            const result = await response.text();
            responseDiv.innerText = result;
        } catch (error) {
            console.error('Error:', error);
            responseDiv.innerText = '오류가 발생했습니다.';
        }
    }
</script>
```

# AskController

```
@Controller  
public class RouteController {  
  
    @GetMapping("/askview")  
    public String askview(){  
        return "ask";  
    }  
}
```

```
@RestController  
public class AskAIController {  
  
    private final ChatService chatService;  
    public AskAIController(ChatService chatService) {  
        this.chatService = chatService;  
    }  
  
    @GetMapping("/ask")  
    public String getResponse(@RequestParam("message") String message){  
        return chatService.getResponse(message);  
    }  
  
    @GetMapping("/ask-ai")  
    public String getResponseOptions(@RequestParam("message") String message){  
        return chatService.getResponseOptions(message);  
    }  
}
```

# ChatService

```
@Service
public class ChatService {

    private final ChatModel chatModel;
    public ChatService(ChatModel chatModel) {
        this.chatModel = chatModel;
    }

    public String getResponse(String message){
        return chatModel.call(message);
    }

    public String getResponseOptions(String message){
        ChatResponse response = chatModel.call(
            new Prompt(
                message,
                OpenAiChatOptions.builder()
                    .withModel("gpt-4o")
                    .withTemperature(0.4)
                    .build()
            ));
        return response.getResult().getOutput().getContent();
    }
}
```

Spring AI의 **OpenAiChatOptions**를 이용해 **Runtime Options**를 설정하는 방법을 설명하겠습니다. 이 옵션을 사용하면 OpenAI의 Chat 모델과 관련된 여러 설정(모델 종류, temperature, frequency penalty 등)을 구성할 수 있습니다.

---

## Github

<https://github.com/parkmaeil/askchat.git>

---

## 04\_DALL-E 3를 이용한 이미지 생성하기

**목표:** OpenAI의 DALL·E 모델을 활용해 입력된 텍스트로부터 이미지를 생성하는 방법을 익힌다.



Spring Boot + Spring AI

# Image Generation

Spring AI

# Application UI

## 이미지 생성

a cute golden retriever sitting in a park

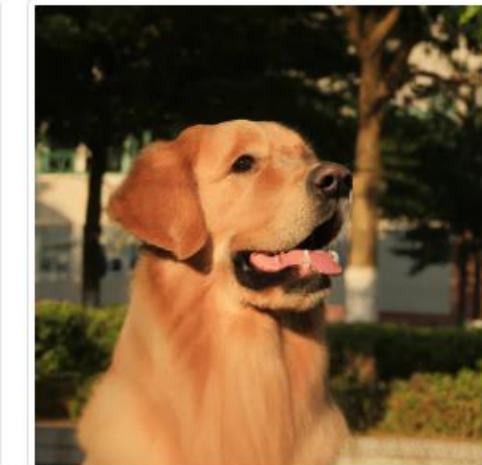
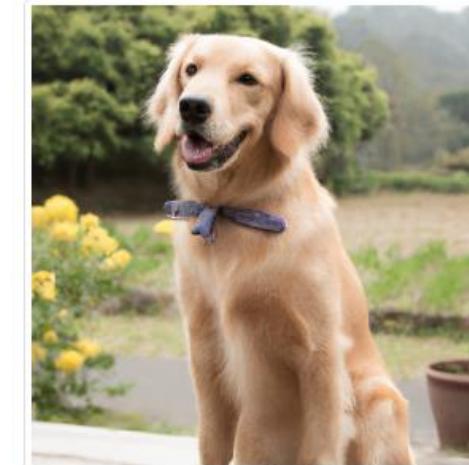
이미지 생성

모델 선택

DALL-E 2

이미지 개수 (1-3)

2



# Application UI

## 이미지 생성

상세 메시지를 입력하세요

이미지 생성

모델 선택

DALL-E 2

이미지 개수 (1-3)

1

## 이미지 생성

a cute golden retriever sitting in a park

이미지 생성

모델 선택

DALL-E 2

이미지 개수 (1-3)

2

이미지 생성 중입니다...

# Application UI

## 이미지 생성

a cute golden retriever sitting in a park

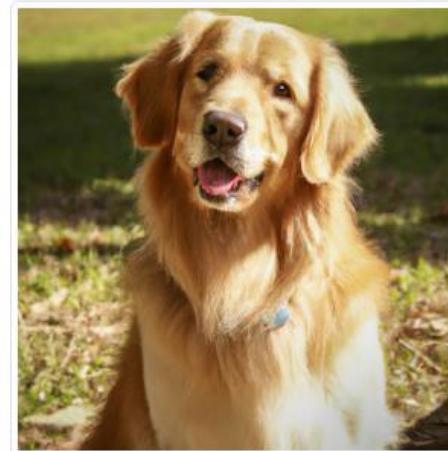
이미지 생성

모델 선택

DALL-E 2

이미지 개수 (1-3)

2



---

## Application UI

달리3을 이용해 이미지를 생성할 때에는 이러한 세부 사항을 반드시 고려해야 한다

1. 주제: 그림의 주제나 핵심 내용은 무엇인지 설명 한다
2. 시각적 요소: 그림에 포함될 주요 시각적 요소 설명 한다 (예: 배경, 주요 객체, 색상 등)
3. 상세 설명: 각 요소에 대한 상세한 설명 제공 (예: 특정 색상의 하늘, 구름의 모양, 나무의 종류 등)
4. 순서와 구조: 그림의 전체적인 구성이나 배열에 대한 설명을 제공한다
5. 스타일과 텍스처: 원하는 그림의 스타일과 텍스처를 설명한다 (예: 실사, 수채화 등)
6. 기타 특이사항: 그림에 추가하거나 특별히 고려해야 할 점이 있다면 설명 한다

이러한 양식을 따라 질문을 작성하시면, 원하는 그림을 더 정확하게 생성하는 데 도움이 될 것입니다.

# 이미지 생성 품

## 이미지 생성

상세 메시지를 입력하세요 이미지 생성

모델 선택 DALL-E 2

이미지 개수 (1-3) 1

```
<!-- 메시지 입력과 버튼을 한 줄로 배치 -->
<div class="input-group mb-3">
  <input type="text" class="form-control" id="message" placeholder="상세 메시지를 입력하세요">
  <button class="btn btn-primary" onclick="generateImage()">이미지 생성</button>
</div>

<!-- 모델 선택과 이미지 개수 선택을 한 줄로 배치 -->
<div class="row mb-3">
  <div class="col-md-6">
    <label for="model" class="form-label">모델 선택</label>
    <select class="form-select" id="model">
      <option value="dall-e-2">DALL-E 2</option>
      <option value="dall-e-3">DALL-E 3</option>
    </select>
  </div>
  <div class="col-md-6">
    <label for="n" class="form-label">이미지 개수 (1-3)</label>
    <input type="number" class="form-control" id="n" min="1" max="3" value="1">
  </div>
</div>
```

# 이미지 생성 스크립트

**프롬프트** : a cute golden retriever sitting in a park

**프롬프트**: 와이드 사진, 바닷가 모래사장에 클래식 피아노 한대가 있고, 한명의 남자가 피아노 치고 있다. 피아노는 20미터 거리고, 모두 열려져 있으며, 측면으로 바라봄. 모래사장은 가로로 펼쳐져 있으며, 바다는 잔잔한 파도가 있고, 주변에 사람들이 없다.

```

<script>
  async function generateImage() {
    const message = document.getElementById("message").value;
    const model = document.getElementById("model").value;
    const n = document.getElementById("n").value;

    // "이미지 생성 중" 메시지 표시
    const loadingAlert = document.getElementById("loadingAlert");
    loadingAlert.style.display = "block";

    try {
      const response = await fetch('/image', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json; charset=UTF-8'
        },
        body: JSON.stringify({
          message: message,
          model: model,
          n: parseInt(n)
        })
      });

      const imageUrls = await response.json();
      console.log(imageUrls);
      // 이미지 컨테이너를 비웁니다.
      const imageContainer = document.getElementById("imageContainer");
      imageContainer.innerHTML = "";

      // 이미지 URL로 이미지 요소 생성 및 다운로드 링크 추가
      imageUrls.forEach((url, index) => {
        const imgElement = document.createElement("img");
        imgElement.src = url;
        imgElement.classList.add("img-thumbnail", "m-2");
        imgElement.style.width = "300px"; // 이미지 너비 설정
        imageContainer.appendChild(imgElement);

        // 클릭 이벤트 추가
        imgElement.addEventListener("click", () => {
          // Spring Boot의 엔드포인트로 요청 보내기
          downloadImage(url);
        });
      });
    } catch (error) {
      console.error("이미지 생성 중 오류가 발생했습니다:", error);
      alert("이미지 생성에 실패했습니다. 다시 시도해 주세요.");
    } finally {
      // "이미지 생성 중" 메시지 숨김
      loadingAlert.style.display = "none";
    }
  }

  // 이미지 다운로드 함수
  function downloadImage(imageUrl) {
    const downloadUrl = '/download-file?url=${encodeURIComponent(imageUrl)}';
    window.location.href = downloadUrl; // 브라우저에서 파일 다운로드 트리거
  }
</script>

```

# ImageGenerationController

```
@Controller
public class RouteController {

    @GetMapping("/imageview")
    public String imageview(){
        return "image";
    }
}

@RestController
public class ImageGenerationController {

    private final ImageService imageService;

    public ImageGenerationController(ImageService imageService) {
        this.imageService = imageService;
    }

    @PostMapping(value = "/image", consumes = "application/json; charset=UTF-8")
    public List<String> image(@RequestBody ImageRequestDTO request) throws IOException {
        ImageResponse imageResponse = imageService.getImageGen(request);

        List<String> imageUrl = imageResponse.getResults().stream()
            .map(result -> result.getOutput().getUrl())
            .toList();
        return imageUrl;
    }
}

@Data
public class ImageRequestDTO {
    private String message;
    private String model;
    private int n;
}
```

# FileDownloadController

```
@Configuration  
public class AppConfig {  
    @Bean  
    public RestTemplate restTemplate() {  
        return new RestTemplate();  
    }  
}
```

```
@RestController  
public class FileDownloadController {  
  
    private final RestTemplate restTemplate;  
  
    public FileDownloadController(RestTemplate restTemplate) {  
        this.restTemplate = restTemplate;  
    }  
  
    @GetMapping("/download-file")  
    public ResponseEntity<byte[]> downloadFile(@RequestParam String url) {  
        try {  
            // URL을 URI 객체로 변환  
            URI uri = new URI(url);  
  
            // RestTemplate로 요청 보내기  
            ResponseEntity<byte[]> response = restTemplate.getForEntity(uri, byte[].class);  
  
            // URL에서 파일 이름 추출  
            String fileName = extractFileName(url);  
  
            // 파일 다운로드용 헤더 설정  
            HttpHeaders downloadHeaders = new HttpHeaders();  
            downloadHeaders.setContentType(MediaType.APPLICATION_OCTET_STREAM);  
            downloadHeaders.setContentDisposition(ContentDisposition.attachment().filename(fileName).build());  
  
            return new ResponseEntity<>(response.getBody(), downloadHeaders, HttpStatus.OK);  
        } catch (Exception e) {  
            // 상세 오류 메시지 반환  
            return ResponseEntity.badRequest()  
                .body("Failed to download file: " + e.getMessage().getBytes());  
        }  
    }  
  
    // 파일 이름 추출 메서드  
    private String extractFileName(String url) {  
        // URI에서 경로의 마지막 부분 추출  
        String path = URI.create(url).getPath();  
        return path.substring(path.lastIndexOf("/") + 1);  
    }  
}
```

# ImageService

```
@Service
public class ImageService {

    private final OpenAilmageModel openAilmageModel;

    public ImageService(OpenAilmageModel openAilmageModel) {
        this.openAilmageModel = openAilmageModel;
    }

    public ImageResponse getImageGen(ImageRequestDTO request){
        ImageResponse imageResponse=openAilmageModel
            .call(new ImagePrompt(request.getMessage(),
                OpenAilmageOptions.builder()
                    .withModel(request.getModel())
                    .withQuality("hd")
                    .withN(request.getN())
                    .withHeight(1024)
                    .withWidth(1024)
                    .build()))
            );
        return imageResponse;
    }
}
```

---

## Github

<https://github.com/parkmaeil/imagegen.git>

---

## 05\_이미지 분석과 수학문제 풀기

**목표:** 이미지 분석을 통해 시각적 데이터를 처리하고, 분석 결과를 바탕으로 수학 문제를 해결하는 AI 기능을 구현한다.



Spring Boot + Spring AI  
**Image에서 Text 추출**

# Application UI

Upload Image for Analysis

Choose and Analyze Image

Analysis Result



Analysis Text

이 이미지는 책 표지로 보이며, 다음과 같은 요소들이 포함되어 있습니다:

1. \*\*텍스트:\*\*

- "한 권으로 배우는 자바 마스터 가이드 북"
- "초보 개발자를 위한 JAVA"
- 저자 이름: 이병승
- "동영상 강의 무료 제공"
- "최신 대응 Java 20"
- "Q&A 네이버 카페 운영"
- 출판사: Youngjin.com (영진닷컴)

2. \*\*이미지:\*\*

- 달리는 사람의 일러스트가 중심에 있습니다. 사람이 역동적으로 달리는 모습이며, 주위에 색색의 리본과 같은 요소들이 함께 그려져 있습니다.

3. \*\*디자인 요소:\*\*

- 책 표지의 전반적인 색상은 흰색 배경에 파란색과 빨간색이 강조되어 있습니다.
- 다양한 폰트 크기와 색상으로 주요 정보를 강조하고 있습니다.

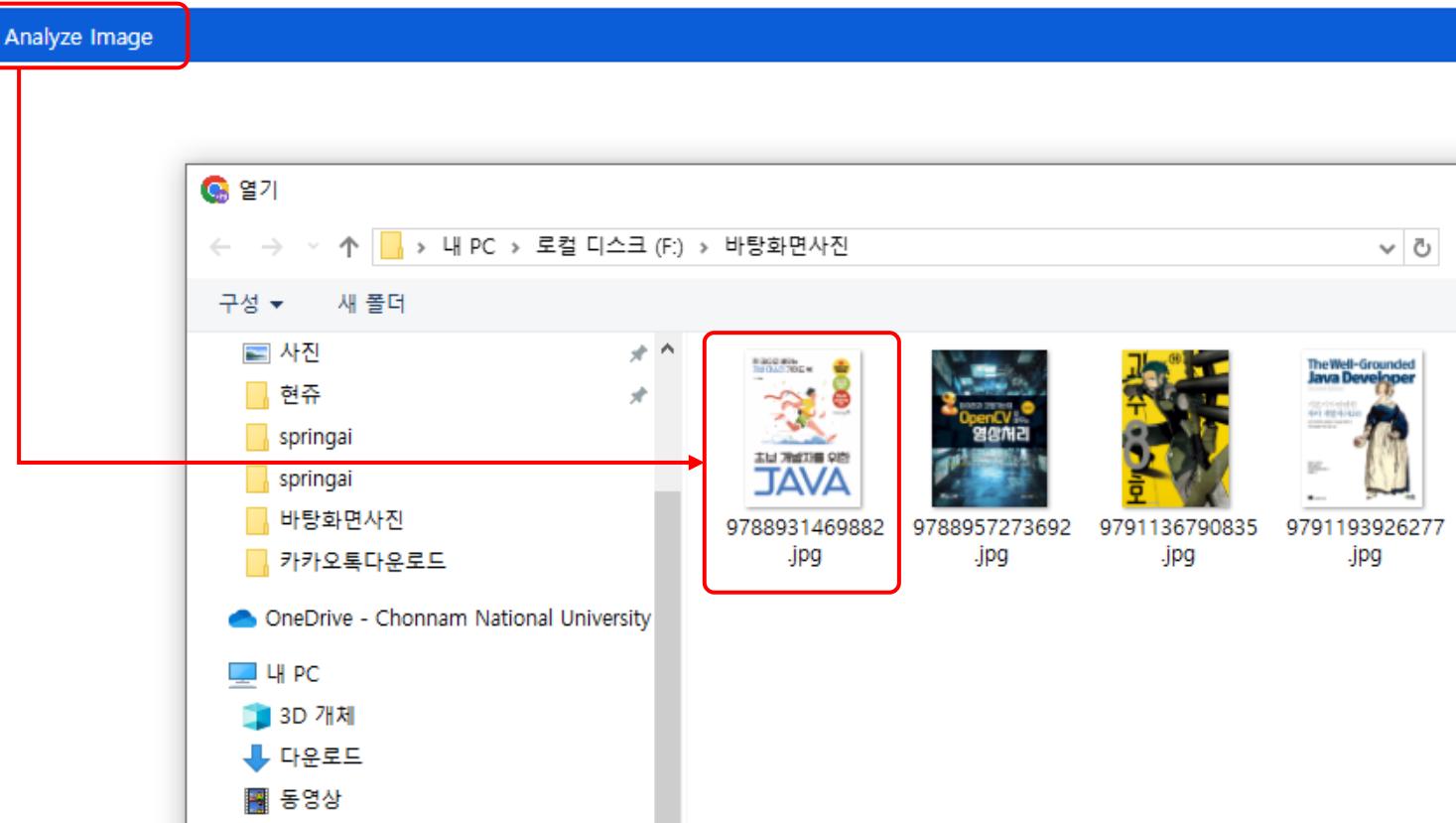
이 책은 Java 프로그래밍을 배우려는 초보 개발자를 위한 가이드 북으로 보입니다. 최신 Java 20 버전에 대응하며, 동영상 강의와 Q&A 네이버 카페를 통해 추가적인 학습 지원을 제공한다고 광고하고 있습니다.

Related YouTube Videos

# Application UI

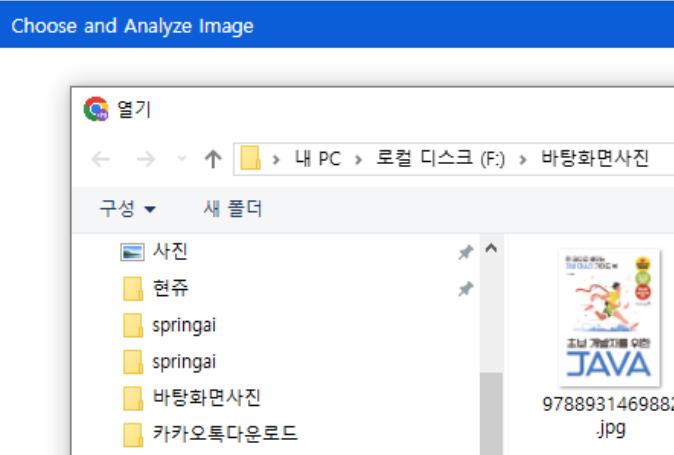
Upload Image for Analysis

Choose and Analyze Image



# 입력 폼

## Upload Image for Analysis



```
<h2>Upload Image for Analysis</h2>
<form id="imageUploadForm" enctype="multipart/form-data">
  <label class="btn btn-primary btn-file d-flex align-items-center">
    <input type="file" class="form-control d-none" id="imageFile" name="image" accept="image/*" required>
    Choose and Analyze Image
  </label>
</form>
```

# 이미지 업로드 스크립트

```
public class ImageAnalysisVO {  
    private String imageUrl;  
    private String analysisText;  
    private List<String> youtubeUrls;  
}
```

```
<script>  
    document.getElementById("imageFile").addEventListener("change", async function () {  
        const formData = new FormData();  
        formData.append("image", document.getElementById("imageFile").files[0]);  
        formData.append("message", "이 이미지에 무엇이 있나요?");  
  
        document.getElementById("loadingMessage").style.display = "block";  
        document.getElementById("resultSection").style.display = "none";  
  
        try {  
            const response = await fetch("/image-text/analyze", {  
                method: "POST",  
                body: formData  
            });  
  
            if (response.ok) {  
                const data = await response.json();  
                // Display the analysis results  
                document.getElementById("resultSection").style.display = "block";  
                document.getElementById("uploadedImage").src = data.imageUrl;  
                document.getElementById("analysisText").innerText = data.analysisText;  
            } else {  
                alert("Image analysis failed. Please try again.");  
            }  
        } catch (error) {  
            console.error("Error:", error);  
            alert("An error occurred. Please try again.");  
        } finally {  
            document.getElementById("loadingMessage").style.display = "none";  
        }  
    });  
</script>
```

# ImageTextGenController

```
@Controller
public class RouteController {

    @GetMapping("/imageview")
    public String imageview(){
        return "imageview";
    }
}
```

application.yml

```
upload:
  path: src/main/resources/static/uploads/
```

```
@Configuration
public class WebConfig implements WebMvcConfigurer {

    @Value("${upload.path}")
    private String uploadPath;

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("/uploads/**") // URL 패턴
            .addResourceLocations("file:" + uploadPath); // 파일 시스템의 실제 경로
    }
}
```

```
@RestController
@RequestMapping("/image-text")
public class ImageTextGenController {

    private final ImageTextGenService imageTextGenService;

    @Value("${upload.path}")
    private String uploadPath;

    public ImageTextGenController(ImageTextGenService imageTextGenService) {
        this.imageTextGenService = imageTextGenService;
    }

    @PostMapping("/analyze")
    public ResponseEntity<ImageAnalysisVO> getMultimodalResponse(
        @RequestParam("image") MultipartFile imageFile,
        @RequestParam(defaultValue = "이 이미지에 무엇이 있나요?") String message) throws IOException {

        // Ensure the upload directory exists
        File uploadDirectory = new File(uploadPath);
        if (!uploadDirectory.exists()) {
            uploadDirectory.mkdirs();
        }

        // Save the uploaded file to the specified upload path
        String filename = imageFile.getOriginalFilename();
        Path filePath = Paths.get(uploadPath, filename);
        Files.write(filePath, imageFile.getBytes());

        // Analyze the image
        String analysisText = imageTextGenService.analyzeImage(imageFile, message);

        String imageUrl = "/uploads/" + filename;
        ImageAnalysisVO response = new ImageAnalysisVO(imageUrl, analysisText, null);
        return ResponseEntity.ok(response);
    }
}
```

# ImageTextGenService

```
@Service
public class ImageTextGenService {

    private final ChatModel chatModel;

    @Value("classpath:/system.message")
    private Resource defaultSystemMessage;

    public ImageTextGenService(ChatModel chatModel) {
        this.chatModel = chatModel;
    }

    public String analyzeImage(MultipartFile inputFile, String message) {
        // MIME 타입 결정
        String contentType = inputFile.getContentType();
        if (!MimeTypeUtils.IMAGE_PNG_VALUE.equals(contentType) &&
            !MimeTypeUtils.IMAGE_JPEG_VALUE.equals(contentType)) {
            throw new IllegalArgumentException("지원되지 않는 이미지 형식입니다.");
        }
        try {
            // Media 객체 생성
            var media = new Media(MediaType.valueOf(contentType), inputFile.getResource());
            // 사용자 메시지 생성
            var userMessage = new UserMessage(message, media);
            // 시스템 메시지 생성
            var systemMessage = new SystemMessage(defaultSystemMessage);
            // AI 모델 호출
            return chatModel.call(userMessage, systemMessage);
        } catch (Exception e) {
            throw new RuntimeException("이미지 처리 중 오류가 발생했습니다.", e);
        }
    }
}
```

system.message  
이미지를 분석해서 텍스트로 만들어주고 자세하게 설명해줘



<https://github.com/parkmaeil/imagetextgen.git>

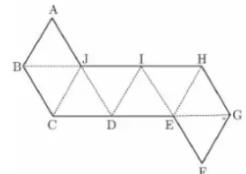
# Application UI 02

Upload Image for Analysis

Choose and Analyze Image

Analysis Result

5. 다음 그림과 같은 진개도를 이용하여 만들어지는 정다면체에 대한 설명으로 옳은 것은 몇 개인가? [4점]



- ① 정팔면체이다.
  - ② 모서리의 개수가 12개다.
  - ③ 꼭짓점 A와 꼭짓점 I는 겹친다.
  - ④ 모서리 AB와 모서리 CD는 평행이다.
  - ⑤ 모서리 AJ와 꼬인 위치에 있는 모서리는 4개다.
- ① 1개      ② 2개      ③ 3개  
④ 4개      ⑤ 5개

Analysis Text

문제 해답:

1. \*\*정팔면체이다.\*\*

- 정팔면체는 8개의 면을 가진 다면체입니다. 주어진 전개도는 정팔면체의 전개도입니다.
- 설명은 맞습니다. (참)

2. \*\*모서리의 개수가 12개다.\*\*

- 정팔면체의 모서리 개수는 12개입니다.
- 설명은 맞습니다. (참)

3. \*\*꼭짓점 A와 꼭짓점 I는 접한다.\*\*

- 전개도를 접었을 때, 꼭짓점 A와 I는 서로 접하지 않습니다.
- 설명은 틀립니다. (거짓)

4. \*\*모서리 AB와 모서리 CD는 평행이다.\*\*

- 전개도를 접었을 때 모서리 AB와 CD는 만나게 되므로 평행하지 않습니다.
- 설명은 틀립니다. (거짓)

5. \*\*모서리 AJ와 꼬인 위치에 있는 모서리는 4개다.\*\*

- 모서리 AJ와 꼬인 위치에 있는 모서리는 4개가 맞습니다.
- 설명은 맞습니다. (참)

각 설명이 참인지 거짓인지 분석한 결과, 참인 설명은 3개입니다.

핵심 키워드: 정팔면체, 모서리, 꼭짓점, 전개도

Related YouTube Videos



Related Video



Related Video



Related Video



Related Video



Related Video

# Application UI 02

Upload Image for Analysis

Choose and Analyze Image

Analysis Result

1.  $\sqrt[3]{24} \times 3^{\frac{2}{3}}$ 의 값은? [2점]

- ① 6    ② 7    ③ 8    ④ 9    ⑤ 10

Analysis Text

문제 해답:

주어진 표현식을 풀어보겠습니다:

$$\sqrt[3]{24} \times 3^{\frac{2}{3}}$$

1. 먼저  $\sqrt[3]{24}$ 를 구합니다. 24를 소인수 분해하면  $24 = 2^3 \times 3$ 이므로,

$$\sqrt[3]{24} = \sqrt[3]{2^3 \times 3} = 2 \times \sqrt[3]{3}$$

2. 다음으로  $3^{\frac{2}{3}}$ 은 다음과 같이 바꿀 수 있습니다:

$$3^{\frac{2}{3}} = (\sqrt[3]{3})^2$$

3. 이제 두 값을 곱합니다:

$$(2 \times \sqrt[3]{3}) \times (\sqrt[3]{3})^2 = 2 \times 3 = 6$$

따라서, 답은 ① 6입니다.

핵심 키워드: 세제곱근, 지수법칙, 소인수분해

LLM(Large Language Model)은 수학적 표기를 나타낼 때 주로 **LaTeX** 형식을 사용

Related YouTube Videos



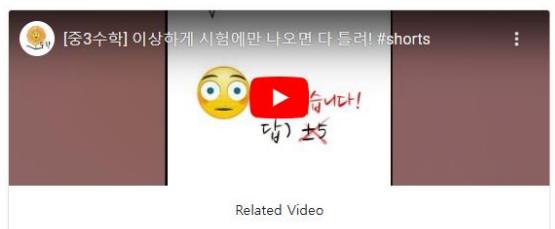
Related Video



Related Video



Related Video



Related Video



Related Video

```
<script src="https://cdn.jsdelivr.net/npm/mathjax@3/es5/tex-mml-chtml.js"></script>
```

→ **MathJax**는 웹페이지에서 수학식을 고품질의 수학 표기법으로 표시하기 위해 사용하는 JavaScript 라이브러리

```
document.getElementById("analysisText").innerHTML = "$a^2 + b^2 = c^2$";
MathJax.typeset(); // 동적으로 추가된 수학식을 렌더링
```

위 코드는 브라우저에서  $a^2 + b^2 = c^2$ 와 같은 수학식으로 렌더링됩니다.

# ImageTextGenController

```
@Controller
public class RouteController {

    @GetMapping("/imageview")
    public String imageview(){
        return "imageview";
    }
}
```

application.yml

```
upload:
  path: src/main/resources/static/uploads/
```

```
@Configuration
public class WebConfig implements WebMvcConfigurer {

    @Value("${upload.path}")
    private String uploadPath;

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("/uploads/**") // URL 패턴
            .addResourceLocations("file:" + uploadPath); // 파일 시스템의 실제 경로
    }
}
```

```
@RestController
@RequestMapping("/image-text")
public class ImageTextGenController {

    @Value("${upload.path}")
    private String uploadPath;

    private final ImageTextGenService imageTextGenService;

    public ImageTextGenController(ImageTextGenService imageTextGenService) {
        this.imageTextGenService = imageTextGenService;
    }

    @PostMapping("/analyze")
    public ResponseEntity<ImageAnalysisVO> getMultimodalResponse(
        @RequestParam("image") MultipartFile imageFile,
        @RequestParam(defaultValue = "이 이미지에 무엇이 있나요?") String message)
        throws IOException {
        // Ensure the upload directory exists
        File uploadDirectory = new File(uploadPath);
        if (!uploadDirectory.exists()) {
            uploadDirectory.mkdirs();
        }

        // Save the uploaded file to the specified upload path
        String filename = imageFile.getOriginalFilename();
        Path filePath = Paths.get(uploadPath, filename);
        Files.write(filePath, imageFile.getBytes());

        // Analyze the image
        String analysisText = imageTextGenService.analyzeImage(imageFile, message);
        String searchKeyword = imageTextGenService.extractKeyYouTubeSearch(analysisText);
        List<String> youtubeUrls = imageTextGenService.searchYouTubeVideos(searchKeyword);
        String imageUrl = "/uploads/" + filename;

        ImageAnalysisVO response = new ImageAnalysisVO(imageUrl, analysisText, youtubeUrls);
        return ResponseEntity.ok(response);
    }
}
```

# ImageTextGenService

## system.message

수학 문제를 자세한 풀이 과정으로 해답을 제공합니다. 수식은 수학에서 사용하는 기호로 표현해주세요.  
또한, 응답에 '핵심 키워드:'를 포함하여 YouTube 검색에 사용할 적절한 핵심 키워드를 명시해주세요.  
핵심 키워드는 문제에서 등장하는 단어로만 해주세요.  
예시 응답 형식: 문제 해답: <해답>, 핵심 키워드: <키워드>

```

@Service
public class ImageTextGenService {

    @Value("classpath:/system.message")
    private Resource defaultSystemMessage;

    private final ChatModel chatModel;
    public ImageTextGenService(ChatModel chatModel) {
        this.chatModel = chatModel;
    }

    public String analyzeImage(MultipartFile imageFile, String message) throws IOException {
        String contentType = imageFile.getContentType();
        if (!MimeTypeUtils.IMAGE_PNG_VALUE.equals(contentType) &&
            !MimeTypeUtils.IMAGE_JPEG_VALUE.equals(contentType)) {
            throw new IllegalArgumentException("지원되지 않는 이미지 형식입니다.");
        }

        var media = new Media(MediaType.valueOf(contentType), imageFile.getResource());
        var userMessage = new UserMessage(message, media);
        var systemMessage = new SystemMessage(defaultSystemMessage);
        return chatModel.call(userMessage, systemMessage);
    }

    public List<String> searchYouTubeVideos(String query) {
        String url = "https://www.googleapis.com/youtube/v3/search?part=snippet&type=video&q=EBS " +
                     query + "&order=relevance&key=본인의 Key를 입력하세요";

        RestTemplate restTemplate = new RestTemplate();
        ResponseEntity<String> response = restTemplate.getForEntity(url, String.class);
        System.out.println(response.getBody());
        List<String> videoUrls = new ArrayList<>();
        JSONObject jsonResponse = new JSONObject(response.getBody());
        JSONArray items = jsonResponse.getJSONArray("items");

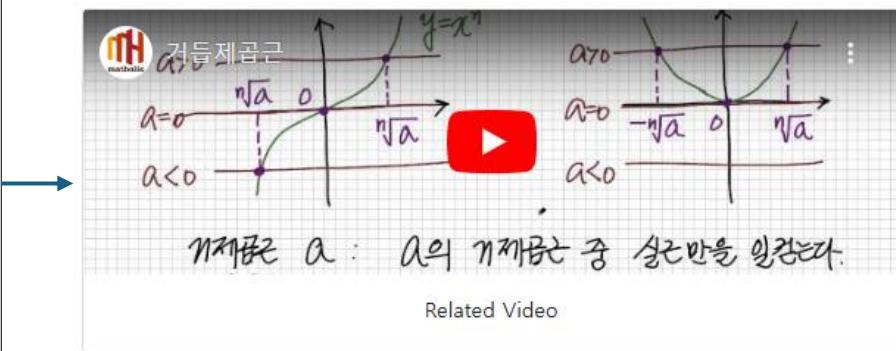
        for (int i = 0; i < items.length(); i++) {
            JSONObject item = items.getJSONObject(i);
            String videoid = item.getJSONObject("id").getString("videoid");
            videoUrls.add(videoid);
        }
        return videoUrls;
    }

    public String extractKeyYouTubeSearch(String analysisText) {
        String keyword=null;
        if(analysisText.indexOf("핵심 키워드:")!=-1){
            keyword=analysisText.substring(analysisText.indexOf("핵심 키워드:")).split(":")[1].trim();
        }
        return keyword;
    }
}

```

# 이미지 스크립트

```
// Display YouTube videos as cards
if (data.youtubeUrls && data.youtubeUrls.length > 0) {
  const youtubeVideosContainer = document.getElementById("youtubeVideos");
  data.youtubeUrls.forEach(videoid => {
    const cardHtml =
      <div class="col-md-6 mb-4">
        <div class="card h-100">
          <iframe
            src="https://www.youtube.com/embed/${videoid}"
            frameborder="0"
            allowfullscreen
            class="card-img-top"
            style="width: 100%; height: 200px;">
          </iframe>
          <div class="card-body text-center">
            <h6 class="card-title">Related Video</h6>
          </div>
        </div>
      </div>;
    youtubeVideosContainer.innerHTML += cardHtml;
  });
}
```



## YouTube Data API v3

- Youtube API 요청은 무료로 사용할 수 있는 대신 하루에 요청할 수 있는 할당량이 정해져 있다. 일반적으로 일간 할당량은 10,000이다.

프로젝트 선택

프로젝트 및 폴더 검색

최근	별표 표시됨	전체
<input checked="" type="checkbox"/> <a href="#">springboot-sns</a>		ID springboot-sns-421813
<input type="checkbox"/> <a href="#">spring-oauth2</a>		ID spring-oauth2-443111

구글 클라우드 콘솔 접속  
<https://console.cloud.google.com/>

Google Cloud

탐색 메뉴 ( )

▲ projects 할당량이 10개 남았습니다. 할당량 증가를 요청하거나 프로젝트를 삭제하세요. [자세히 알아보기](#)

[MANAGE QUOTAS](#)

프로젝트 이름 \* **youtube**

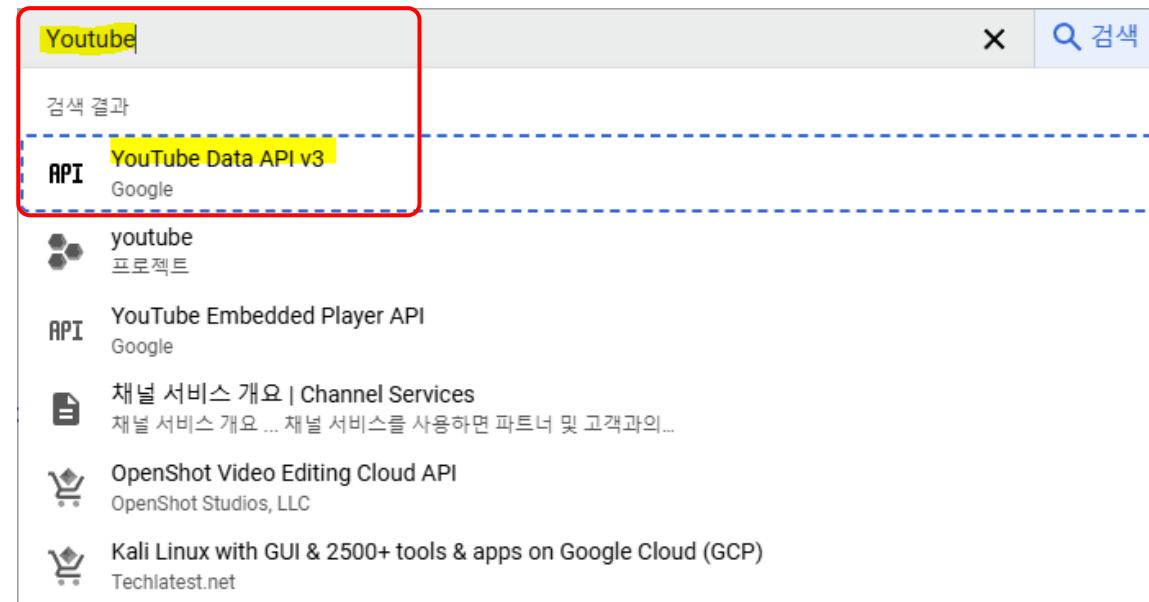
프로젝트 ID: clear-vision-445402-c9. 나중에 변경할 수 없습니다. [수정](#)

위치 \*  **조직 없음** [찾아보기](#)

상위 조직 또는 폴더

만들기      취소

## YouTube Data API v3



### YouTube Data API v3

[Google](#)

The YouTube Data API v3 is an API that provides access to YouTube data, such as videos, playlists,...

[사용](#)

[API 사용해 보기](#)

<https://developers.google.com/youtube/v3/docs?apix=true&hl=ko>

# YouTube Data API v3

The screenshot shows the official Google Cloud Platform documentation for the YouTube Data API v3. The URL is <https://cloud.google.com/youtube-data/api/v3/reference/search/list>. The page title is "YouTube Data API v3". A large red box highlights the "API 사용해 보기" (Try API) button. On the left, a sidebar menu lists various API methods under categories like "활동" (Activities), "자막" (Captions), "채널" (Channels), etc. A red box highlights the "list" method under the "검색" (Search) category. The main content area shows the "Search: list" method details, including its purpose ("API 요청에 지정된 쿼리 매개변수와 일치하는 검색결과 컬렉션을 반환합니다."), supported parameters ("video, channel, playlist"), and a note about usage limits ("할당량 영향"). Below this, a section titled "일반적인 사용 사례" (Common Use Cases) provides information on how to use the method.

**YouTube Data API v3**

[Google](#)

The YouTube Data API v3 is an API that provides access to YouTube data, such as videos, playlists,...

사용 API 사용해 보기

개요

▶ 활동

▶ 자막

▶ ChannelBanners

▶ 채널

▶ 채널 섹션

▶ 댓글

▶ 댓글 대화목록

▶ i18n언어

▶ i18n지역

▶ 구성원

▶ 멤버십 등급

▶ PlaylistImages

▶ PlaylistItems

▶ 재생목록

▶ 검색

개요

list

홈 > 제품 > YouTube > Data API > 참고자료 도움이 되었나요? ⌂ ⌂

## Search: list

API 요청에 지정된 쿼리 매개변수와 일치하는 검색결과 컬렉션을 반환합니다. 기본적으로 검색결과 집합은 일치하는 `video`, `channel`, `playlist` 리소스를 식별하지만 특정 유형의 리소스만 검색하도록 쿼리를 구성할 수도 있습니다.

★ 할당량 영향: 이 메서드를 호출하면 100단위의 할당량 비용이 적용됩니다.

### 일반적인 사용 사례

The table below shows common use cases for this method. You can click on a use case name to load sample parameter values in the APIs Explorer. Or you can see code samples for a use case in the fullscreen APIs Explorer by clicking on the code icon below a use case name. In the fullscreen UI, you can update parameter and property values and the code samples will dynamically update to reflect the values you enter.

# YouTube Data API v3

## 요청

### HTTP 요청

```
GET https://www.googleapis.com/youtube/v3/search
```

## 매개변수

다음 표에는 이 쿼리가 지원하는 매개변수가 나와 있습니다. 나열된 모든 매개변수는 쿼리 매개변수입니다.

### 매개변수

#### 필수 매개변수

**part**

**string**

**part** 매개변수는 API 응답이 포함하는 **search** 리소스 속성 하나 이상의 쉼표로 구분된 목록을 지정합니다. 매개변수 값을 **snippet**로 설정합니다.

## YouTube Data API v3

```
String url = "https://www.googleapis.com/youtube/v3/search?part=snippet&type=video&q=EBS " +  
    query + "&order=relevance&key=[REDACTED]";
```

part	string	part 매개변수는 API 응답이 포함하는 search 리소스 속성 하나 이상의 쉼표로 구분된 목록을 지정합니다. 매개변수 값을 snippet로 설정합니다.
order	string	order 매개변수는 API 응답에서 리소스를 정렬하는데 사용할 메서드를 지정합니다. 기본값은 relevance입니다.
type	string	type 매개변수는 특정 유형의 리소스만 검색하도록 검색어를 제한합니다. 값은 쉼표로 구분된 리소스 유형 목록입니다. 기본값은 video,channel,playlist입니다.
q	string	q 매개변수는 검색할 검색어를 지정합니다.

# API Key 발급

Google Cloud

springboot-sns

Youtube

**API API 및 서비스**

**사용자 인증 정보**

**+ 사용자 인증 정보 만들기**

**삭제**

**삭제된 사용자 인증 정보 복원**

**사용 설정된 API 및 서비스**

사용 설정한 API에 액세스하려면

**API 키**

할당량과 액세스 권한을 확인하기 위해 간단한 API 키로 프로젝트를 확인합니다.

**OAuth 클라이언트 ID**

앱에서 사용자 데이터에 액세스할 수 있도록 사용자 동의를 요청합니다.

**이름**

표시할 API 키가 없습니다.

**서비스 계정**

로봇 계정을 사용하여 서버 간의 앱 수준 인증을 사용 설정합니다.

**사용자 인증 정보 선택 도움말**

사용할 사용자 인증 정보의 유형을 결정할 수 있도록 몇 가지 질문을 합니다.

**API 키 생성 완료**

애플리케이션에서 이 키를 사용하려면 키를 `key=API_KEY` 파라미터로 전달하세요.

▲ 이 키는 제한되지 않습니다. 무단 사용을 방지하려면 API를 사용할 수 있는 위치와 대상을 제한하는 것이 좋습니다. 제한사항을 추가하려면 [API 키를 수정하세요](#). [자세히 알아보기](#)



<https://github.com/parkmaeil/imagemathgen.git>

---

## 06\_AI로 만드는 나만의 레시피

**목표:** 사용자의 입력을 기반으로 AI가 맞춤형 레시피를 생성하도록 구현하고 OpenAI의 텍스트 처리 능력을 활용한다.

---

Spring Boot + Spring AI

# Recipe Generation

AI와 함께하는 나만의 레시피 만들기. 똑똑하게 냉장고를 관리하고  
나의 레시피를 커뮤니티에 공유해 보세요

# Application UI

레시피 생성기

홈 레시피 찾기 문의하기

레시피 생성기

재료

예: 새우, 아보카도, 고수

요리 유형

예: 멕시코 요리

식이 제한

예: 페스코테리언, 유제품 없음

레시피 생성

# Application UI

레시피 생성기
홈 레시피 찾기 문의하기



**레시피 생성기**

**재료**

닭가슴살, 레몬, 타임, 로즈마리, 후추

**요리 유형**

프랑스 요리

**식이 제한**

고단백, 저탄수화물

**레시피 생성**

⟳ 레시피가 생성 중입니다...

요리 제목: 허브 레몬 향의 프랑스식 치킨 구이

### 재료 목록:

- 닭가슴살 2조각
- 레몬 1개 (즙과 제스트)
- 신선한 타임 1큰술
- 신선한 로즈마리 1큰술
- 올리브 오일 2큰술
- 소금 약간
- 신선한 후추 약간
- 마늘 2쪽 (다진 것)
- 화이트 와인 1/4컵 (선택 사항)
- 버터 1큰술 (선택 사항)

### 조리법:

# Application UI

요리 제목: 허브 레몬 향의 프랑스식 치킨 구이

### 재료 목록:

- 닭가슴살 2조각
- 레몬 1개 (즙과 제스트)
- 신선한 타임 1큰술
- 신선한 로즈마리 1큰술
- 올리브 오일 2큰술
- 소금 약간
- 신선한 후추 약간
- 마늘 2쪽 (다진 것)
- 화이트 와인 1/4컵 (선택 사항)
- 버터 1큰술 (선택 사항)

### 조리법:

1. \*\*재료 준비:\*\*

- 레몬은 껍질을 제스트로 갈아낸 후 반으로 잘라 즙을 짹니다.
- 신선한 타임과 로즈마리는 잘게 다져 준비합니다.
- 마늘은 곱게 다집니다.

2. \*\*닭가슴살 밑간:\*\*

- 닭가슴살을 깨끗이 씻은 후 키친타월로 물기를 제거합니다.
- 소금과 신선한 후추로 양면에 고루 밑간합니다.

## 참고할 수 있는 웹사이트

- <https://blog.naver.com/9810213/140205634675?viewType=pc>
- <https://m.blog.naver.com/ranaskitchen/220218975687>
- <https://blog.naver.com/jisie/221060540488>
- <https://arteflame.com/ko/blogs/%EC%9A%94%EB%A6%AC%EB%B2%95/%EC%99%84%EB%B2%BD%ED%95%9C-%EA%B5%AC%EC%9A%BA-%EC%9C%A1%EC%A6%99-%EC%B9%98%ED%82%A8-%EC%98%A8-%EB%8D%94-%EC%95%84%EB%A5%B4%ED%85%8C%ED%94%8C%EB%A0%88%EC%9E%84-%EA%B7%B8%EB%A6%B4>
- [https://signaturemg.co.kr/news\\_view.php?idx=174&cate=&cate\\_sub=](https://signaturemg.co.kr/news_view.php?idx=174&cate=&cate_sub=)
- <https://arteflame.com/ko/blogs/%EC%9A%94%EB%A6%AC%EB%B2%95/%EB%A0%88%EB%AA%AC-%ED%97%88%EB%B8%8C-%EC%B9%98%ED%82%A8>
- <https://maestropasta.cz/ko/%ED%92%8D%EB%AF%88-%EB%84%98%EC%B9%98%EB%8A%94-%EB%8B%2A%8EB%8B%2B%3-%EA%B0%99%EC%9D%80-%EB%A0%88%EB%AA%AC-%EC%B9%98%ED%82%A8%EA%B3%BC-%ED%92%8D%EB%8B%6%80%ED%95%9C-%ED%97%88%EB%B8%8C-%EC%86%8C%EC%8A%A4/>
- <https://arteflame.com/ko/blogs/%EC%9A%94%EB%A6%AC%EB%B2%95/%EC%89%BD%EA%B2%8C-%EA%B5%AC%EC%9A%BA-%EB%A0%88%EB%AA%AC-%EB%A7%88%EB%8A%98-%EC%B9%98%ED%82%A8-%EC%A0%80%EB%85%81>
- <https://m.cooknchefnews.com/news/view/1065563506545250>
- <https://arteflame.com/ko/blogs/%EC%9A%94%EB%A6%AC%EB%B2%95/%EC%B6%94%EC%88%98%EA%B0%90%EC%82%AC%EC%A0%88-%EB%A1%9C%ED%8B%80%EC%84%8B%80%EB%A6%AC-%EC%B9%98%ED%82%A8-%EC%95%84%EB%A5%B4%ED%85%8C%ED%94%8C%EB%A0%88%EC%9E%84>

# 레시피 생성 품

## 레시피 생성기

재료

닭가슴살, 레몬, 타임, 로즈마리, 후추

요리 유형

프랑스 요리

식이 제한

고단백, 저탄수화물

레시피 생성

- **Ingredients:** 피자에 필요한 재료 (치즈, 밀가루, 토마토)
- **Cuisine:** 서양 요리, 즉 피자 스타일
- **Dietary Restrictions:** 채식주의자용 피자이므로 고기를 제외한 재료로만 요리를 생성

```
<form id="recipeForm" class="mb-4">
  <div class="mb-3">
    <label for="ingredients" class="form-label">재료</label>
    <input type="text" class="form-control" id="ingredients" placeholder="예: 샐러드, 아보카도, 고수">
  </div>
  <div class="mb-3">
    <label for="cuisine" class="form-label">요리 유형</label>
    <input type="text" class="form-control" id="cuisine" placeholder="예: 멕시코 요리">
  </div>
  <div class="mb-3">
    <label for="dietaryRestrictions" class="form-label">식이 제한</label>
    <input type="text" class="form-control" id="dietaryRestrictions" placeholder="예: 패스코테리언, 유제품 없음">
  </div>
  <button type="button" class="btn btn-primary w-100" onclick="createRecipe()">레시피 생성</button>
</form>
```

# 레시피 생성 스크립트

```
{
  "ingredients": "파프리카, 버섯, 양파, 마늘, 올리브 오일",
  "cuisine": "지중해 요리",
  "dietaryRestrictions": "비건, 유제품 없음"
}

{
  "ingredients": "닭가슴살, 레몬, 타임, 로즈마리, 후추",
  "cuisine": "프랑스 요리",
  "dietaryRestrictions": "고단백, 저탄수화물"
}

{
  "ingredients": "새우, 아보카도, 고수, 라임, 토마토",
  "cuisine": "멕시코 요리",
  "dietaryRestrictions": "페스코테리언, 유제품 없음"
}
```

```
<script>
  async function createRecipe() {
    const ingredients = document.getElementById('ingredients').value;
    const cuisine = document.getElementById('cuisine').value;
    const dietaryRestrictions = document.getElementById('dietaryRestrictions').value;

    const recipeData = {
      ingredients: ingredients,
      cuisine: cuisine,
      dietaryRestrictions: dietaryRestrictions
    };
    // 로딩 메시지 표시
    document.getElementById('loadingMessage').style.display = 'block';
    document.getElementById('recipeOutput').style.display = 'none';
    document.getElementById('recipeUrl').style.display = 'none';
    try {
      const response = await fetch('/recipe', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(recipeData)
      });
      const data = await response.json();
      // URL 목록을 HTML 형식으로 생성
      const urlsHtml = data.urls && data.urls.length > 0
        ? data.urls.map(url => `<li><a href="${url}" target="_blank">${url}</a></li>`).join("")
        : '<li>참고할 수 있는 웹사이트가 없습니다.</li>';
      // 로딩 메시지 숨기기
      document.getElementById('loadingMessage').style.display = 'none';

      // 레시피와 URL을 출력
      document.getElementById('recipeOutput').innerText = `${data.recipe}`;
      document.getElementById('recipeOutput').style.display = 'block';

      document.getElementById('recipeUrl').innerHTML = `<h4>참고할 수 있는 웹사이트</h4><ul>${urlsHtml}</ul>`;
      document.getElementById('recipeUrl').style.display = 'block';
    } catch (error) {
      console.error('에러 발생:', error);
      document.getElementById('loadingMessage').style.display = 'none';
      document.getElementById('recipeOutput').innerHTML = `<p class="text-danger">레시피 생성 중 오류가 발생했습니다.</p>`;
      document.getElementById('recipeOutput').style.display = 'block';
    }
  }
</script>
```

# RecipeController

```
@Controller  
public class RouteController {  
  
    @GetMapping("/recipeview")  
    public String recipeview(){  
        return "recipe";  
    }  
}
```

```
@Data  
public class Recipe {  
    private String ingredients; // 음식재료  
    private String cuisine; // 요리유형  
    private String dietaryRestrictions; // 식단제한정보  
}
```

```
@RestController  
public class RecipeController {  
  
    private final RecipeService recipeService;  
  
    public RecipeController(RecipeService recipeService) {  
        this.recipeService = recipeService;  
    }  
  
    @PostMapping("/recipe")  
    public Map<String, Object> recipe(@RequestBody Recipe recipe) throws IOException {  
        return recipeService.createRecipeWithUrls(recipe);  
    }  
}
```

# RecipeService

## Google CSE(Custom Search Engine) 사용하여 검색결과 가져오기

```
@Service
public class RecipeService {

    private final ChatModel chatModel;

    private final String googleApiKey = "AIzaSyCUCsGUiRsRizNLKUWjNt-IWFjGLRDoNw48"; // Google API 키를 입력하세요
    private final String googleCx = "2261514c6d6474586"; // Google Custom Search Engine CX 키를 입력하세요

    private final RestTemplate restTemplate = new RestTemplate();

    public RecipeService(ChatModel chatModel) {
        this.chatModel = chatModel;
    }

    // 레시피 생성 메서드
    public String createRecipe(Recipe recipe) {
        String template = """
            제목: 요리 제목을 제공해 주세요.
            다음 재료를 사용하여 요리법을 만들고 싶습니다: {ingredients}.
            선호하는 요리 유형은 {cuisine}입니다.
            다음 식이 제한을 고려해 주세요: {dietaryRestrictions}.
            재료 목록과 조리법을 포함한 상세한 요리법을 제공해 주세요.
            """;
        PromptTemplate promptTemplate = new PromptTemplate(template);
        Map<String, Object> params = Map.of(
            "ingredients", recipe.getIngredients(),
            "cuisine", recipe.getCuisine(),
            "dietaryRestrictions", recipe.getDietaryRestrictions()
        );

        Prompt prompt = promptTemplate.create(params);
        return chatModel.call(prompt).getResult().getOutput().getContent();
    }
}
```

계속

# RecipeService

각 검색 요청에는 3개의 쿼리 [parameters] 가 필요합니다.

- API 키 - `key` 쿼리 매개변수를 사용하여 애플리케이션 식별.
- 프로그래밍 검색 엔진 ID - `cx` 를 사용하여 이 검색을 수행하는 데 사용할 프로그래밍 검색 엔진입니다.  
참고: 검색엔진 ID (cx)는 다른 형식 (예: 8ac1ab64606d234f1)일 수 있습니다.
- 검색어 - `q` 쿼리 매개변수를 사용하여 검색 표현식을 지정합니다.

다른 모든 쿼리 매개변수는 선택사항입니다.

```
// Google Custom Search API를 사용하여 관련 URL 검색
public List<String> searchRecipeUrls(String query) throws IOException {
    // API URI 생성
    Uri apiUrl = UriComponentsBuilder.fromHttpUrl("https://www.googleapis.com/customsearch/v1")
        .queryParam("key", googleApiKey)
        .queryParam("cx", googleCx)
        .queryParam("q", query)
        .build()
        .toUri();
    System.out.println(apiUrl.toString());
    // API 호출
    String response = restTemplate.getForObject(apiUrl, String.class);

    // JSON 응답 파싱
    JsonObject jsonResponse = JsonParser.parseString(response).getAsJsonObject();
    System.out.println(jsonResponse.toString());

    JSONArray itemsArray = jsonResponse.getAsJSONArray("items");

    // itemsArray가 null인지 확인
    List<String> urls = new ArrayList<>();
    if (itemsArray != null) {
        for (JsonElement item : itemsArray) {
            urls.add(item.getAsJsonObject().get("link").getAsString());
        }
    } else {
        System.out.println("No search results found for the query: " + query);
    }
    return urls;
}
// 레시피와 링크를 함께 제공하는 메서드
public Map<String, Object> createRecipeWithUrls(Recipe recipe) throws IOException {
    String recipeContent = createRecipe(recipe); // 레시피 설명 생성
    List<String> urls = searchRecipeUrls(recipe.getIngredients()); // Google Custom Search API로 URL 검색
    return Map.of("recipe", recipeContent, "urls", urls);
}
```

# Google Custom Search API

The screenshot shows the Google Cloud API library interface. On the left, there's a sidebar with navigation links: 'API 및 서비스' (selected), '라이브러리', '사용자 인증 정보', 'OAuth 등의 화면', and '페이지 사용 등의'. The main area has tabs for 'youtube' and 'Custom'. The 'Custom' tab is active, displaying a search result for 'Custom'. The result includes '검색 결과' (Search results) with items like 'Support', 'Custom Domains', and 'Custom Search API'. The 'Custom Search API' item is highlighted with a red dashed border. Below the search results, there are other services listed: 'Custom Voice 기본사항 - Cloud Text-to-Speech' and 'Marketplace'.

Google Cloud

youtube

API 및 서비스

API 및 서비스

UTC+9 오후 6:00 12월 23일

필터 필터

이름 요청

YouTube Data API v3 4

Analytics Hub API

BigQuery API

BigQuery Connection API

Custom

검색 결과

Support  
From basic free help to paid packages

Custom Domains  
App Engine

**Custom Search API**  
Google

Custom Voice 기본사항 - Cloud Text-to-Speech  
Custom Voice는 제공된 오디오 데이터와 최대한 유사하게 작동...

Marketplace

# Google Custom Search API

제품 세부정보

## Custom Search API

Google

Retrieve and display search results from Google Custom Search programmatically.

[관리](#) [API 사용해 보기](#) [✓ API 사용 설정됨](#)

You're all set!

You're ready to start developing with Custom Search API

YOUR API KEY

A1zaSyCUCsGUiTsRizNLKUjjNt-IIfjGLRDaNw48

To improve your app's security, restrict this key's usage in the [API Console](#).

DONE

### Custom Search JSON API: 소개

이 문서는 Custom Search JSON API 및 사용법을 익히는데 도움이 됩니다.

#### 시작하기 전에

##### 프로그래밍 검색 엔진 만들기

사용자는 API를 호출하여 프로그래밍 검색 엔진의 기존 인스턴스에 요청을 보냅니다. 따라서 API를 사용하기 전에 [제어판](#)에서 API를 만들어야 합니다. 다양한 구성 옵션에 대한 자세한 내용은 [튜토리얼](#)을 참조하세요. 검색엔진 ID를 만든 후 개요 페이지의 기본 섹션에서 확인할 수 있습니다. API에서 사용하는 `cx` 매개변수입니다.

#### API 키로 Google에 애플리케이션 식별

Custom Search JSON API를 사용하려면 API 키를 사용해야 합니다. API 키는 Google에서 클라이언트를 식별하는 방법입니다.

프로그래밍 검색 엔진 (무료 버전) 사용자: [키 가져오기](#)

API 키가 있으면 애플리케이션에서 쿼리 매개변수 `key=yourAPIKey`을 모든 요청 URL에 추가할 수 있습니다. API 키는 URL에 삽입해도 안전하며 인코딩이 필요하지 않습니다.

# Google Custom Search API

## 새 검색엔진 만들기

먼저 엔진에 관한 기본 정보를 제공해 주세요. 엔진 구성(언어, 지역 등)을 만든 후 추가로 맞춤설정할 수 있습니다. [자세히 알아보기](#)

검색엔진 이름 지정

검색할 내용  특성 사이트 또는 페이지 검색  
 전체 웹 검색

다음 항목을 추가할 수 있습니다.

개별 페이지: www.example.com/page.html  
전체 사이트: www.mysite.com/\*  
사이트의 일부: www.example.com/docs/\* 또는 www.example.com/docs/  
전체 도메인: \*.example.com

검색 설정  이미지 검색  
 세이프서치

로봇이 아닙니다.  reCAPTCHA  
개인정보 보호 - 악관

'만들기'를 클릭하면 다음에 동의하는 것으로 간주됩니다. [서비스 약관](#).

## 개요

### 기본

검색엔진 이름

설명

코드

검색엔진 ID

공개 URL

# Google Custom Search API

## 사용해 보기

코드를 작성하지 않고 API를 둘러보면서 기능을 살펴보려면 '[API 사용해 보기 도구](#)'를 방문하세요.

매개변수에 대한 자세한 설명은 [cse.list 참조](#)를 확인하세요.

HTTP 요청을 통해 API를 사용하는 방법을 알아보려면 [REST 사용](#)으로 이동하세요.

### 요청하기

REST([Representational State Transfer](#)) 맞춤 검색 JSON API의 API는 기존 REST와 다소 다릅니다. API는 리소스에 대한 액세스 권한을 제공하는 대신 서비스에 대한 액세스 권한을 제공합니다. 따라서 API는 서비스 엔드포인트 역할을 하는 단일 URI를 제공합니다.

HTTP `GET` 를 전송하여 특정 검색의 결과를 가져올 수 있습니다. 이 URI에 요청을 보냅니다. 검색 요청의 세부정보를 퀘리 매개변수를 추가할 수 있습니다. Custom Search JSON API URI의 형식은 다음과 같습니다.

`https://www.googleapis.com/customsearch/v1?parameters`

각 검색 요청에는 3개의 퀘리 `[parameters]` 가 필요합니다.

- **API 키** - `key` 퀘리 매개변수를 사용하여 [애플리케이션 식별](#).
- **프로그래밍 검색 엔진 ID** - `cx` 를 사용하여 이 검색을 수행하는 데 사용할 프로그래밍 검색 엔진입니다. 검색엔진은 [제어판](#)에서 만들어야 합니다.  
참고: 검색엔진 ID (`cx`)는 다른 형식 (예: 8ac1ab64606d234f1)일 수 있습니다.
- **검색어** - `q` 퀘리 매개변수를 사용하여 검색 표현식을 지정합니다.

다른 모든 퀘리 매개변수는 선택사항입니다.

이것은 강의용 프로그래밍 검색 엔진 테스트:

GET `https://www.googleapis.com/customsearch/v1?key=INSERT_YOUR_API_KEY&cx=017576662512468239146:omuauf_lfv&q=lectures`

# Google Custom Search API

www.googleapis.com/customsearch/v1?key=AIzaSyCUCsGUiIsRizNLKUWjNt-IWFjGLRDoNw4

retty print 적용

```

    "totalResults": "49800",
    "formattedTotalResults": "49,800"
}
{
  "items": [
    {
      "kind": "customsearch#result",
      "title": "펭이버섯볶음 블랙 올리브 요리! 짭조름한 간단한 와인안주 추천 ...",
      "htmlTitle": "펭이버섯볶음 블랙 올리브 요리! 짭조름한 간단한 와인안주 추천 ...",
      "link": "https://blog.naver.com/hsk0328/222791406035?viewType=pc",
      "displayLink": "blog.naver.com",
      "snippet": "Jun 27, 2022 ... ... 올리브오일 1스푼 반 둘러주시고, 편으로 쟈 깐 마늘을 넣고, 중불에서 2~3분 1스푼 반 둘러주시고,",
      "htmlSnippet": "Jun 27, 2022 #u003cb##u003E...#u003C/b##u003E ... #u003cb##u003E올리브오일#u003C/b##u003E 1스푼 반 둘러주시고,",
      "formattedUrl": "https://blog.naver.com/hsk0328/222791406035?viewType=pc",
      "htmlFormattedUrl": "https://blog.naver.com/hsk0328/222791406035?viewType=pc",
      "pagemap": {
        "metatags": [
          {
            "og:image": "https://blogthumb.pstatic.net/MjAyMjA2MjdfOCAg/MDAxNjU2MzM5Mzk0ODE1.XUpTx04PCzneprtW7LSIXQxDY5RWPfBF%CD%CO%CE%BE%C8%C1%D6-%C3%DF%C3%B5.jpg?type=w2",
            "og:type": "article",
            "og:site_name": "네이버 블로그 | 미소앨범 by 하준맘",
            "og:title": "펭이버섯볶음 블랙 올리브 요리! 짭조름한 간단한 와인안주 추천",
            "naverblog:profile_image": "https://blogthumb-phinf.pstatic.net/20160421_296/hsk0328_1461209520099c0A61_GIF/1",
            "napp-site-analysis": "gdid=90000003_0000000000000003DF6711D3,service=blog, collection=blog",
            "og:description": "펭이버섯볶음 블랙 올리브 요리! 짭조름한 간단한 와인안주 추천 스콘이나 피자 만들 때 사용하려면!",
            "referrer": "always",
            "naverblog:nickname": "미소순이",
            "me:feed:serviceid": "blog",
            "viewport": "width=device-width,initial-scale=1,minimum-scale=1,maximum-scale=1,user-scalable=no,viewport-fit=cover",
            "og:url": "https://blog.naver.com/hsk0328/222791406035",
            "og:article:author": "미소앨범 b.. | 블로그",
            "format-detection": "telephone=no"
          }
        ],
        "cse_image": [
          {
            "src": "https://blogthumb.pstatic.net/MjAyMjA2MjdfOCAg/MDAxNjU2MzM5Mzk0ODE1.XUpTx04PCzneprtW7LSIXQxDY5RWPfBChGuC3%DF%C3%B5.jpg?type=w2"
          }
        ]
      }
    }
  ]
}

```

---

**Github**

<https://github.com/parkmaeil/recipegen.git>

---

## 07\_음성을 텍스트로 변환 하기(STT)

**목표:** OpenAI API와 STT 기능을 사용하여 음성 데이터를 텍스트로 변환하는 기능을 구현한다.

---

## Speech to Text(STT)

음성을 텍스트로 변환하는 **STT**(Speech-to-Text) 기술을 활용하여, 대화 음성 파일을 텍스트로 저장할 수 있다

# Application UI

## Upload Audio/Video File for Transcription

Select audio/video file (MP4)

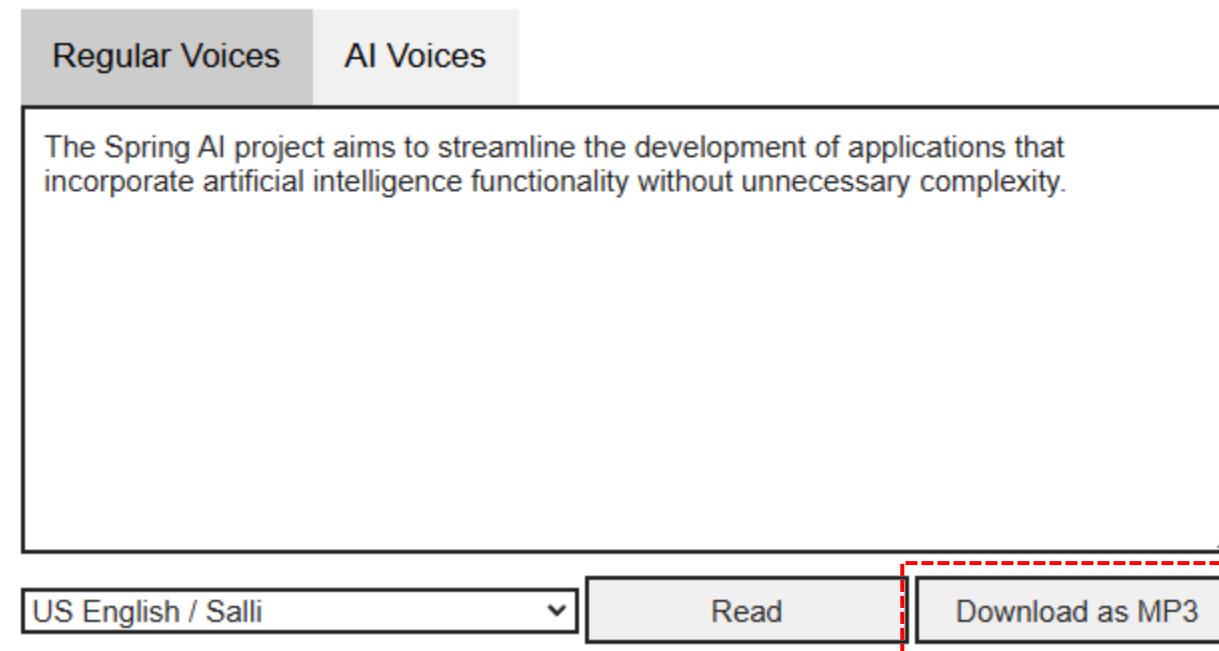
파일 선택

선택된 파일 없음

Upload and Transcribe

## 음성파일 만들기

<https://ttsmp3.com/>



## 음성을 텍스트로 변환

### Upload Audio/Video File for Transcription

Select audio/video file (MP4)

파일 선택 stt.mp3

Upload and Transcribe

Transcribed Text:

The Spring AI project aims to streamline the development of applications that incorporate artificial intelligence functionality without unnecessary complexity.

### Upload Audio/Video File for Transcription

Select audio/video file (MP4)

파일 선택 stt.mp3

Upload and Transcribe

Transcribed Text:

스프링 AI 프로젝트의 목표는 인공지능 기능을 필요한 복잡함 없이 개발한 기술 개발 프로젝트의 개발 작업을 스트리밍하는 것입니다.

# Transcription API

Spring AI는 OpenAI의 **음성 텍스트 변환(Transcription) API**를 지원합니다

The screenshot shows a navigation tree for 'Audio Models'. Under 'Transcription API', there are two items: 'Azure OpenAI' and 'OpenAI'. The 'OpenAI' link is highlighted with a red dashed rectangle. Below the tree, there is a link to the documentation: <https://docs.spring.io/spring-ai/reference/api/audio/transcriptions/openai-transcriptions.html>. There is also a link to 'Text-To-Speech (TTS) API'.

Next, create a **OpenAiAudioTranscriptionModel**

```
var openAiAudioApi = new OpenAiAudioApi(System.getenv("OPENAI_API_KEY"));

var openAiAudioTranscriptionModel = new OpenAiAudioTranscriptionModel(this.openAiAudioApi);

var transcriptionOptions = OpenAiAudioTranscriptionOptions.builder()
    .withResponseFormat(TranscriptResponseFormat.TEXT)
    .withTemperature(0f)
    .build();

var audioFile = new FileSystemResource("/path/to/your/resource/speech/jfk.flac");

AudioTranscriptionPrompt transcriptionRequest = new AudioTranscriptionPrompt(this.audioFile, this.transcriptionOptions);
AudioTranscriptionResponse response = openAiTranscriptionModel.call(this.transcriptionRequest);
```

# RestController

```
@RestController
public class AIController {

    private final OpenAiAudioTranscriptionModel openAiAudioTranscriptionModel;

    public AIController(OpenAiAudioTranscriptionModel openAiAudioTranscriptionModel){
        this.openAiAudioTranscriptionModel=openAiAudioTranscriptionModel;
    }

    // 오디오 파일 업로드 후 변환
    @PostMapping("/transcribe")
    public ResponseEntity<String> transcribe(@RequestParam("file") MultipartFile file) throws Exception {
        Resource resource = file.getResource();
        // Set transcription options
        OpenAiAudioTranscriptionOptions options = OpenAiAudioTranscriptionOptions.builder()
            .withResponseFormat(OpenAiAudioApi.TranscriptResponseFormat.TEXT)
            .withLanguage("ko")
            .withTemperature(0f)
            // .withResponseFormat(this.responseFormat)
            .build();

        // Create a transcription prompt
        AudioTranscriptionPrompt audioTranscriptionPrompt =new AudioTranscriptionPrompt(resource, options);

        // Call the transcription API
        AudioTranscriptionResponse audioTranscriptionResponse = openAiAudioTranscriptionModel.call(audioTranscriptionPrompt);

        // Return the transcribed text
        return new ResponseEntity<>(audioTranscriptionResponse.getResult().getOutput(), HttpStatus.OK);
    }
}
```

# View

```
<form id="uploadForm" enctype="multipart/form-data" method="post">
  <div class="mb-3">
    <label for="fileInput" class="form-label">Select audio/video file
    (MP4)</label>
    <input class="form-control" type="file" id="fileInput" name="file"
    accept=".mp4" required>
  </div>
  <button type="submit" class="btn btn-primary w-100">Upload and
  Transcribe</button>
</form>
```

```
<script>
  const uploadForm = document.getElementById('uploadForm');
  const loadingMessage = document.getElementById('loadingMessage');
  const transcriptionResult = document.getElementById('transcriptionResult');
  const transcriptionText = document.getElementById('transcriptionText');

  // Hide the result section initially
  transcriptionResult.style.display = 'none';

  // Handle form submission
  uploadForm.addEventListener('submit', function (event) {
    event.preventDefault();

    const formData = new FormData(uploadForm);
    loadingMessage.style.display = 'block';
    transcriptionResult.style.display = 'none';

    // Send file to server for transcription
    fetch('/transcribe', {
      method: 'POST',
      body: formData
    })
    .then(response => response.text())
    .then(data => {
      // Display the transcribed text
      transcriptionText.textContent = data;
      loadingMessage.style.display = 'none';
      transcriptionResult.style.display = 'block';
    })
    .catch(error => {
      console.error('Error:', error);
      loadingMessage.textContent = 'An error occurred during transcription.';
    });
  });
</script>
```

# Controller

```
@Controller  
public class RouteController {  
  
    @GetMapping("/stt")  
    public String stt(){  
        return "stt";  
    }  
}
```

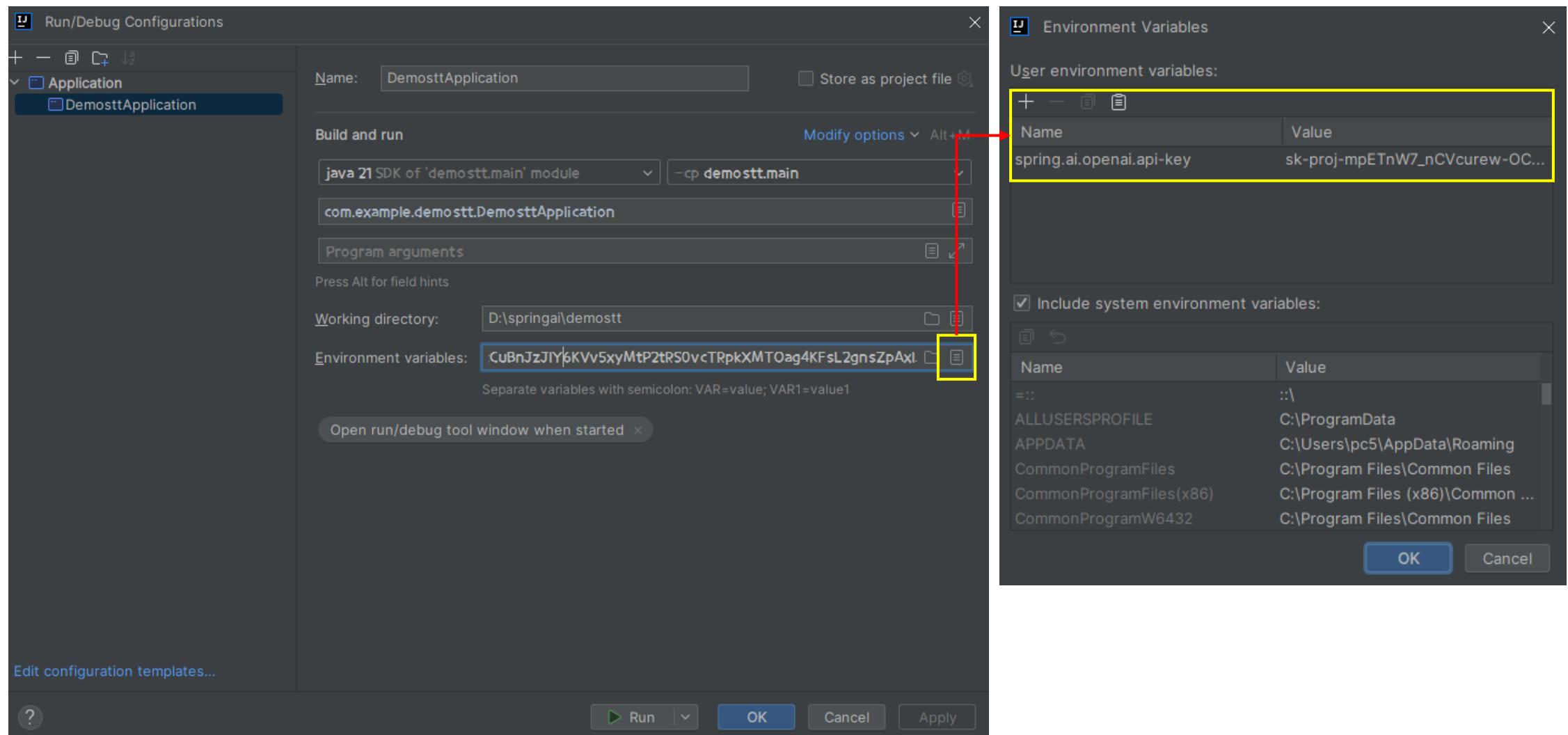
application.properties

```
spring.servlet.multipart.max-file-size=10MB  
spring.servlet.multipart.max-request-size=10MB
```

build.gradle

```
implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'  
implementation 'org.springframework.boot:spring-boot-starter-web'  
implementation 'org.springframework.ai:spring-ai-openai-spring-boot-starter'
```

# OpenAPI key Setting



---

**Github**

<https://github.com/parkmaeil/speechtotext.git>

---

## 08\_OpenAI로 만든 음성 서비스(TTS)

**목표:** OpenAI의 TTS(Text-to-Speech) 기능을 활용해 텍스트 데이터를 음성으로 출력하는 서비스를 구축한다.

---

## Text to Speech(TTS)

안녕하세요, 이 음성은 OpenAI로 만들어진 음성입니다.

# Application UI

## 텍스트를 음성으로 변환(TTS )

텍스트 파일 업로드

파일 선택

선택된 파일 없음

파일 업로드 및 변환

# Application UI

## 텍스트를 음성으로 변환(TTS )

텍스트 파일 업로드



파일 업로드 및 변환

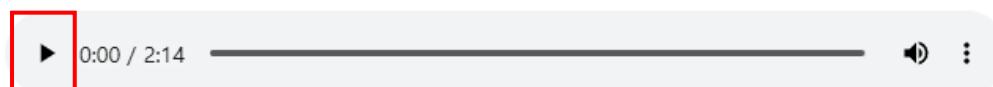
변환 중입니다. 잠시만 기다려주세요...

## 텍스트를 음성으로 변환(TTS )

텍스트 파일 업로드



파일 업로드 및 변환



tts.txt

한강은 1970년 대한민국 광주에서 태어나 아홉 살 때 가족과 함께 서울로 이주했습니다. 아버지가 저명한 소설가인 문학가 집안 출신입니다. 글쓰기와 함께 미술과 음악에도 심취했으며, 이는 그녀의 문학 작품 전반에 반영되어 있습니다.

한강은 1993년 잡지 『문학과사회』("Literature and Society")에 다수의 시를 발표하며 문단에 등단하기 시작했습니다. 1995년 단편소설집 『여수의 사랑』("Love of Yeosu")으로 산문 데뷔를 했고, 곧이어 소설과 단편소설 등 여러 산문 작품을 발표했습니다. 그중에서도 주목할 만한 작품은 소설 『그대의 차가운 손』(2002, "Your Cold Hands")으로, 예술에 대한 한강의 관심의 흔적이 뚜렷하게 남아 있습니다. 이 소설은 실종된 조각가가 여성의 신체 섹시 모형 제작에 집착하며 남긴 원고를 재현한 작품입니다. 인체 해부학에 대한 집착과 페르소나와 경험 사이의 유희, 조각가의 작업에서 신체를 드러내는 것과 감추는 것 사이의 갈등이 발생합니다. '삶은 심연 위에 아치형 시트를 엮은 것이고, 우리는 가면 쓴 곡예사처럼 그 위에서 살아간다'는 책의 마지막 문장이 이를 단적으로 말해줍니다.

한강은 소설 『채식주의자』(2007, "The Vegetarian", 2015)로 국제적으로 큰 반향을 일으켰습니다. 총 3부로 구성된 이 소설은 주인공 영혜가 음식 섭취의 규범에 복종하기를 거부했을 때 벌어지는 폭력적인 결과를 묘사합니다. 고기를 먹지 않기로 한 그녀의 결정은 전혀 다른 다양한 반응에 부딪힙니다. 남편과 권위주의적인 아버지는 그녀의 행동을 강제로 거부하고, 비디오 아티스트인 시동생은 그녀의 수동적인 몸에 집착하며 에로틱하고 미학적으로 그녀를 착취합니다. 결국 그녀는 정신병원에 입원하게 되고, 언니는 그녀를 구출해 '정상적인' 삶으로 돌려보내려고 노력합니다. 하지만 영혜는 위험하면서도 매혹적인 식물 왕국의 상징인 '불타는 나무'를 통해 정신병과 같은 상태에 점점 더 깊이 빠져들게 됩니다.

좀 더 서사에 기반한 책은 2010년 출간된 『바람이 분다, 가라』("The Wind Blows, Go")로, 우정과 예술에 관한 크고 복잡한 소설로 슬픔과 변화에 대한 갈망을 강렬하게 드러냅니다.

# TTSCController

```
@Controller
public class RouteController {

    @GetMapping("/audioPlay")
    public String audioPlay(){
        return "tts";
    }
}
```

**Flux는 Reactive Streams의 일종으로**  
 데이터를 비동기적으로 처리하며,  
 데이터가 준비되는 대로 전달할 수 있습니다.

```
@RestController
public class TTSCController {

    private final OpenAiAudioSpeechModel openAiAudioSpeechModel;

    public TTSCController(OpenAiAudioSpeechModel openAiAudioSpeechModel) {
        this.openAiAudioSpeechModel = openAiAudioSpeechModel;
    }

    @PostMapping("/upload")
    public ResponseEntity<StreamingResponseBody> uploadFile(@RequestParam("file") MultipartFile file) throws IOException {
        // 업로드된 파일의 텍스트 내용 읽기
        String content = new String(file.getBytes(), StandardCharsets.UTF_8);

        OpenAiAudioSpeechOptions options = OpenAiAudioSpeechOptions.builder()
            .withVoice(OpenAiAudioApi.SpeechRequest.Voice.ALLOY)
            .withSpeed(1.1f)
            .withResponseFormat(OpenAiAudioApi.SpeechRequest.AudioResponseFormat.MP3)
            .withModel(OpenAiAudioApi.TtsModel.TTS_1.value)
            .build();

        SpeechPrompt speechPrompt = new SpeechPrompt(content, options);

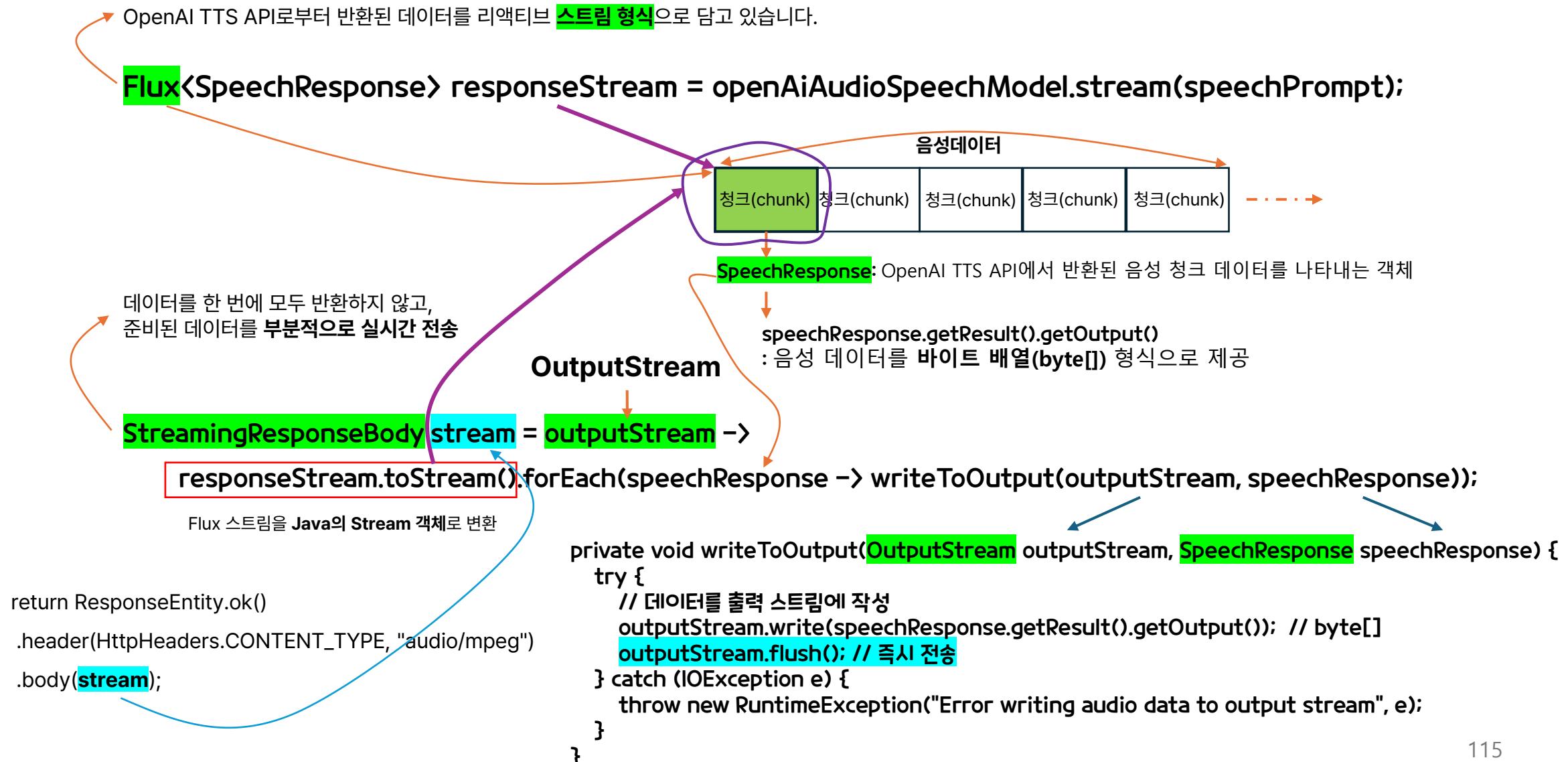
        // 리액티브 스트림 생성(실시간 오디오 스트리밍)
        Flux<SpeechResponse> responseStream = openAiAudioSpeechModel.stream(speechPrompt);

        // 청크 데이터를 클라이언트로 전달
        StreamingResponseBody stream = outputStream ->
            responseStream.toStream().forEach(speechResponse -> writeToOutput(outputStream, speechResponse));

        return ResponseEntity
            .OK()
            .header(HttpHeaders.CONTENT_TYPE, "audio/mpeg") // MP3 파일로 설정
            .body(stream);
    }

    private void writeToOutput(OutputStream outputStream, SpeechResponse speechResponse) {
        try {
            // 데이터를 출력 스트림에 작성
            outputStream.write(speechResponse.getResult().getOutput());
            outputStream.flush(); // 즉시 전송
        } catch (IOException e) {
            throw new RuntimeException("Error writing audio data to output stream", e);
        }
    }
}
```

# TTSCController



```

return ResponseEntity.ok()
.header(HttpHeaders.CONTENT_TYPE, "audio/mpeg")
.body(stream);
  
```

## 텍스트파일 업로드 화면

### 텍스트를 음성으로 변환(TTS )

텍스트 파일 업로드

파일 선택      선택된 파일 없음

파일 업로드 및 변환

```
<form id="uploadForm" class="mb-3">
    <div class="mb-3">
        <label for="fileInput" class="form-label">텍스트 파일 업로드</label>
        <input class="form-control" type="file" id="fileInput" accept=".txt" required>
    </div>
    <button type="submit" class="btn btn-primary w-100">파일 업로드 및 변환</button>
</form>
```

## 텍스트파일 업로드 스크립트

```
<script>
  const uploadForm = document.getElementById('uploadForm');
  const fileInput = document.getElementById('fileInput');
  const audioElement = document.getElementById('audioPlayer');
  const loadingMessage = document.getElementById('loadingMessage');

  // 파일 업로드 및 서버 요청
  uploadForm.addEventListener('submit', async function (event) {
    event.preventDefault();

    const file = fileInput.files[0];
    if (!file) {
      alert('텍스트 파일을 선택해주세요.');
      return;
    }

    // FormData에 파일 추가
    const formData = new FormData();
    formData.append('file', file);

    // 로딩 메시지 표시
    loadingMessage.style.display = 'block';
    audioElement.style.display = 'none';
  });
</script>
```

```
try {
  // 서버에 요청
  const response = await fetch('/upload', { method: 'POST', body: formData });
  if (!response.ok) throw new Error('네트워크 오류');

  // 응답 스트림 처리
  const reader = response.body.getReader();
  const chunks = [];
  const contentType = response.headers.get('Content-Type');

  while (true) {
    const { done, value } = await reader.read(); // 4,8,16KB
    if (done) break;
    chunks.push(value);
  }

  // Blob으로 변환
  const blob = new Blob(chunks, { type: contentType || 'audio/mpeg' });
  const url = URL.createObjectURL(blob);

  // 오디오 설정 및 재생
  audioElement.src = url;
  loadingMessage.style.display = 'none';
  audioElement.style.display = 'block';
} catch (error) {
  console.error('Error:', error);
  loadingMessage.textContent = '변환에 실패했습니다. 다시 시도해주세요.';
}
});
```

## 텍스트파일 업로드 스크립트

```

const response = await fetch('/upload', {
  method: 'POST',
  body: formData
});

// 스트림을 읽을 수 있는 Reader 객체 반환
const reader = response.body.getReader();

const chunks = []; chunks 배열에는 서버에서 받은 모든 청크 데이터가 저장

while (true) {
  // 서버로부터 데이터 **청크(chunk)**를 비동기로 읽습니다.
  const { done, value } = await reader.read(); // 4,8,16KB , byte[]
  if (done) break;
  chunks.push(value);
}

```

Blob 객체는 이러한 청크 데이터를 **하나의 파일(객체)**로 결합(MP3)

```

const blob = new Blob(chunks, { type: contentType || 'audio/mpeg' });
const url = URL.createObjectURL(blob); Blob 객체에서 파일 데이터를 URL 형식으로 변환합니다.

```

변환된 URL은 브라우저에서 **해당 파일을 참조하는 경로**로 사용할 수 있습니다.

URL을 <audio> 태그의 src 속성에 설정하면 브라우저에서 바로 재생이 가능합니다.



- **done:**  
true이면 모든 데이터를 다 읽은 상태입니다.  
false이면 아직 읽을 데이터가 남아 있습니다.
- **value:**  
현재 읽은 데이터 \*\*청크(byte array)\*\*입니다.

---

## application.properties

application.properties

```
spring.servlet.multipart.max-file-size=10MB  
spring.servlet.multipart.max-request-size=10MB
```

```
spring.ai.openai.audio.speech.options.model=tts-1  
spring.ai.openai.audio.speech.options.voice=alloy  
spring.ai.openai.audio.speech.options.response-format=mp3  
spring.ai.openai.audio.speech.options.speed=1.0
```

build.gradle

```
implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'  
implementation 'org.springframework.boot:spring-boot-starter-web'  
implementation 'org.springframework.ai:spring-ai-openai-spring-boot-starter'
```

---

**Github**

<https://github.com/parkmaeil/texttospeech.git>

---

## **09\_사용자 친화형 검색 서비스(GPT+SQL)**

**목표:** GPT와 SQL을 결합하여 데이터베이스 질의와 검색 결과를 자연어로 표현하는 사용자 친화형 검색 서비스를 구축한다.

---

## Table Schema

### movies.sql

-- 감독의 기본 정보

```
CREATE TABLE Directors (
    id /V/T NOT NULL AUTO_INCREMENT COMMENT '감독의 고유 ID',
    name VARCHAR(255) NOT NULL COMMENT '감독 이름',
    birth_year /V/T COMMENT '감독의 출생 연도',
    PRIMARY KEY (id)
);
```

-- 영화의 기본 정보

```
CREATE TABLE Movies (
    id /V/T NOT NULL AUTO_INCREMENT COMMENT '영화의 고유 ID',
    title VARCHAR(255) NOT NULL COMMENT '영화 제목',
    release_year /V/T NOT NULL COMMENT '영화 개봉 연도',
    director_ref /V/T NOT NULL COMMENT '영화 감독 ID (Directors 테이블 참조)',
    PRIMARY KEY (id),
    FOREIGN KEY (director_ref) REFERENCES Directors(id)
);
```

-- 배우의 기본 정보

```
CREATE TABLE Actors (
    id /V/T NOT NULL AUTO_INCREMENT COMMENT '배우의 고유 ID',
    name VARCHAR(255) NOT NULL COMMENT '배우 이름',
    birth_year /V/T COMMENT '배우의 출생 연도',
    PRIMARY KEY (id)
);
```

-- 영화 장르

```
CREATE TABLE MovieGenres (
    id /V/T NOT NULL AUTO_INCREMENT COMMENT '영화 장르의 고유 ID',
    genre_name VARCHAR(100) NOT NULL COMMENT '영화 장르 이름',
    PRIMARY KEY (id)
);
```

-- 영화에 출연한 배우를 연결

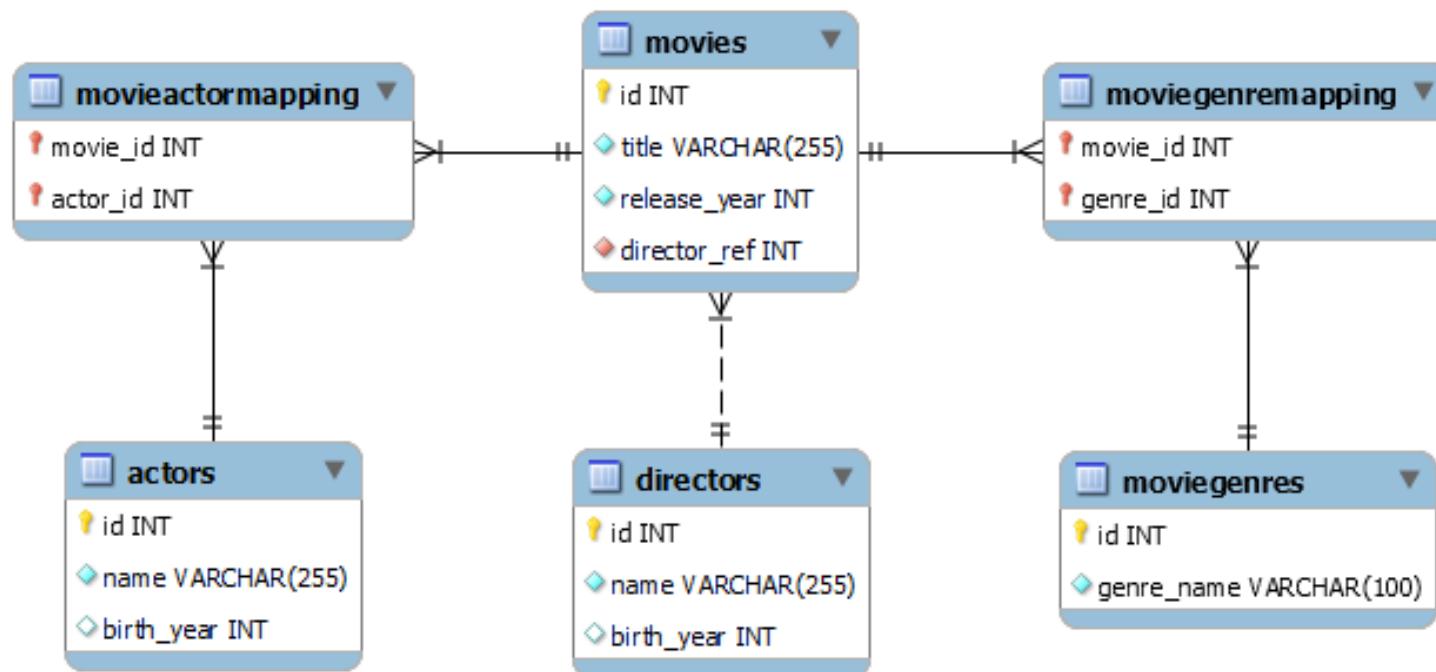
```
CREATE TABLE MovieActorMapping (
    movie_id /V/T NOT NULL COMMENT '영화 ID (Movies 테이블 참조)',
    actor_id /V/T NOT NULL COMMENT '배우 ID (Actors 테이블 참조)',
    PRIMARY KEY (movie_id, actor_id),
    FOREIGN KEY (movie_id) REFERENCES Movies(id),
    FOREIGN KEY (actor_id) REFERENCES Actors(id)
);
```

-- 영화의 장르를 연결

```
CREATE TABLE MovieGenreMapping (
    movie_id /V/T NOT NULL COMMENT '영화 ID (Movies 테이블 참조)',
    genre_id /V/T NOT NULL COMMENT '장르 ID (MovieGenres 테이블 참조)',
    PRIMARY KEY (movie_id, genre_id),
    FOREIGN KEY (movie_id) REFERENCES Movies(id),
    FOREIGN KEY (genre_id) REFERENCES MovieGenres(id)
);
```

---

# ERD



## Data INSERT SQL

### -- Directors (감독 데이터)

```
INSERT INTO Directors (id, name, birth_year) VALUES
(1, '봉준호', 1969),
(2, '박찬욱', 1963),
(3, '김한민', 1969),
(4, '연상호', 1978),
(5, '이창동', 1954);
```

### -- Actors (배우 데이터)

```
INSERT INTO Actors (id, name, birth_year) VALUES
(1, '송강호', 1967),
(2, '이병헌', 1970),
(3, '전도연', 1973),
(4, '마동석', 1971),
(5, '김혜수', 1970),
(6, '유아인', 1986),
(7, '정우성', 1973),
(8, '김윤석', 1968);
```

### -- MovieGenres (장르 데이터)

```
INSERT INTO MovieGenres (id, genre_name) VALUES
(1, '드라마'),
(2, '스릴러'),
(3, '액션'),
(4, '코미디'),
(5, 'SF');
```

### -- Movies (영화 데이터)

```
INSERT INTO Movies (id, title, release_year, director_ref) VALUES
(1, '기생충', 2019, 1),
(2, '괴물', 2006, 1),
(3, '올드보이', 2003, 2),
(4, '아가씨', 2016, 2),
(5, '명량', 2014, 3),
(6, '한산: 용의 출현', 2022, 3),
(7, '부산행', 2016, 4),
(8, '비상선언', 2022, 4),
(9, '버닝', 2018, 5),
(10, '밀양', 2007, 5);
```

### -- MovieActorMapping (영화와 배우 매팅)

```
INSERT INTO MovieActorMapping (movie_id, actor_id) VALUES
(1, 1), (1, 6),
(2, 1), (2, 5),
(3, 6), (3, 7),
(4, 6), (4, 3),
(5, 2), (5, 8),
(6, 2), (6, 8),
(7, 4), (7, 6),
(8, 1), (8, 4),
(9, 6), (9, 3),
(10, 3), (10, 5);
```

### -- MovieGenreMapping (영화와 장르 매팅)

```
INSERT INTO MovieGenreMapping (movie_id, genre_id) VALUES
(1, 1),(1, 2),
(2, 2),(2, 3),
(3, 2),(4, 1),
(5, 3),(6, 3),
(7, 5),(7, 2),
(8, 3),(8, 1),
(9, 1),(10, 1);
```

## SQL SELECT

Q) 송강호 배우가 출연한 영화와 해당 영화의 감독 정보를 알려줘

```

SELECT
    m.title AS movie_title,
    d.name AS director_name
FROM
    Movies m
JOIN
    Directors d ON m.director_ref = d.id
JOIN
    MovieActorMapping ma ON m.id = ma.movie_id
JOIN
    Actors a ON ma.actor_id = a.id
WHERE
    a.name = '송강호';
  
```

movie_title	director_name
기생충	봉준호
괴물	봉준호
비상선언	연상호

- 특정 배우가 출연한 모든 영화를 검색하는 SQL.
- 특정 감독이 제작한 모든 영화를 검색하는 SQL.
- 특정 장르에 속한 모든 영화를 검색하는 SQL.
- 여러 배우가 함께 출연한 영화를 검색하는 SQL.

## SQL SELECT

Q) 특정 배우가 출연한 모든 영화를 검색하는 SQL.

- 특정 배우가 출연한 모든 영화를 검색하는 SQL.
- 특정 감독이 제작한 모든 영화를 검색하는 SQL.
- 특정 장르에 속한 모든 영화를 검색하는 SQL.

마동석 배우가 출연한 모든 영화를 알려줘

마동석 배우가 출연한 모든 영화를 알려줘

검색

Generated SQL Query

```
SELECT m.title
FROM Movies m
JOIN MovieActorMapping mam ON m.id = mam.movie_id
JOIN Actors a ON mam.actor_id = a.id
WHERE a.name = '마동석';
```

title
부산행
비상선언

## Movie Search UI

### 사용자 친화형 검색(GPT+SQL)

송강호 배우가 출연한 영화와 해당 영화의 감독 정보를 알려줘

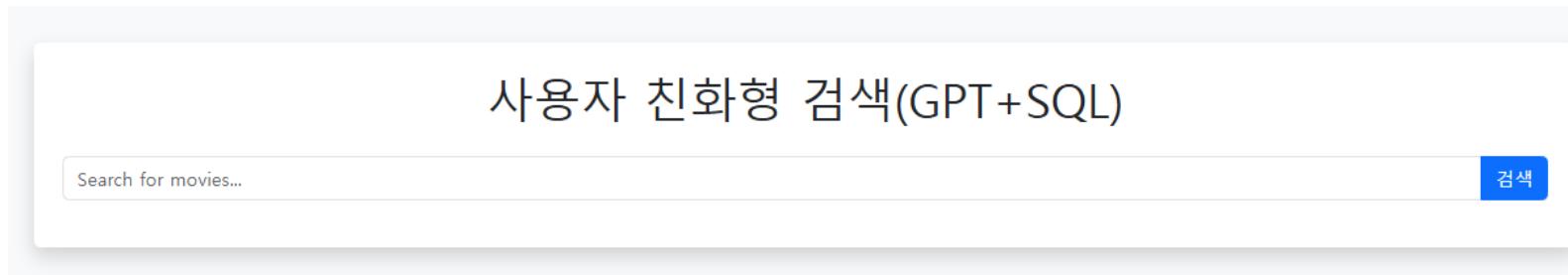
검색

#### Generated SQL Query

```
SELECT m.title, d.name AS director_name
FROM Movies m
JOIN MovieActorMapping mam ON m.id = mam.movie_id
JOIN Actors a ON mam.actor_id = a.id
JOIN Directors d ON m.director_ref = d.id
WHERE a.name = '송강호';
```

title	director_name
기생충	봉준호
괴물	봉준호
비상선언	연상호

## 검색화면



```
<div class="container bg-white rounded shadow p-4">
    <h1 class="text-center mb-4">사용자 친화형 검색(GPT+SQL)</h1>
    <div class="input-group mb-3">
        <input id="newsQuery" type="text" class="form-control" placeholder="Search for movies...">
        <button id="searchBtn" class="btn btn-primary">검색</button>
    </div>
    <div id="loading" class="d-none text-center my-3">
        <div class="spinner-border text-primary" role="status">
            <span class="visually-hidden">Loading...</span>
        </div>
    </div>
    <div id="newsResult" class="mt-3">
        <!-- 검색 결과가 표시됩니다. -->
    </div>
</div>
```

## Controller

```
@Controller  
public class IndexController {  
  
    @GetMapping("/")  
    public String index(){  
        return "sql";  
    }  
}
```

```
@RestController  
@RequestMapping("/sql")  
public class SqlController {  
  
    @Value("classpath:/movies.sql")  
    private Resource ddlResource;  
  
    @Value("classpath:/sql-prompt-template.st")  
    private Resource sqlPromptTemplateResource;  
  
    private final ChatClient aiClient;  
    private final JdbcTemplate jdbcTemplate;  
  
    public SqlController(ChatClient.Builder aiClient, JdbcTemplate jdbcTemplate) {  
        this.aiClient = aiClient.build();  
        this.jdbcTemplate = jdbcTemplate;  
    }  
  
    @PostMapping  
    public SqlResponse sql(@RequestParam(name = "question") String question) throws IOException {  
        String schema = ddlResource.getContentAsString(Charset.defaultCharset());  
  
        String query = aiClient.prompt()  
            .user(userSpec -> userSpec  
                  .text(sqlPromptTemplateResource)  
                  .param("question", question)  
                  .param("ddl", schema)  
            )  
            .call()  
            .content();  
  
        if (query.toLowerCase().startsWith("select")) {  
            return new SqlResponse(query, jdbcTemplate.queryForList(query));  
        }  
        return new SqlResponse(query, List.of());  
    }  
}
```

## View

SQL

```

    {
      sqlQuery: "SELECT m.title, d.name FROM Movies m JOIN Directors d ON m.director_ref = d.id WHERE d.name = '송강호';",
      results: Array(3)
    }
  
```

results[0] 0: {title: '기생충', name: '봉준호'}  
 results[1] 1: {title: '괴물', name: '봉준호'}  
 results[2] 2: {title: '비상선언', name: '연상호'}  
 length: 3  
 [[Prototype]]: Array(0)  
 sqlQuery: "SELECT m.title, d.name FROM Movies m JOIN Directors d ON m.director\_ref = d.id WHERE d.name = '송강호';"  
 [[Prototype]]: Object

**title**

기생충	봉준호
괴물	봉준호
비상선언	연상호

**name**

List<Map<String, Object>>

```

// 결과 테이블 생성
const resultsHtml = `
  <div class="table-responsive">
    <table class="table table-bordered table-striped">
      <thead class="table-light">
        <tr>
          ${Object.keys(data.results[0]).map(
            key => '<th>'${key}</th>')
          .join("")}
        </tr>
      </thead>
      <tbody>
        ${data.results.map(row =>
          <tr>
            ${Object.values(row).map(
              value => '<td>'${value}</td>')
              .join("")}
            </tr>
          ).join("")}
      </tbody>
    </table>
  </div>
`;
  
```

title

name

기생충

봉준호

괴물

봉준호

비상선언

연상호

기생충

봉준호

봉준호

연상호

기생충

봉준호

**Prompt****sql-prompt-template.st**

DDL 섹션의 DDL이 주어지면 질문 섹션에서 질문에 답하기 위한 SQL 쿼리를 작성합니다.  
SQL 쿼리 이외에는 아무것도 작성하지 마십시오. 쿼리는 마크다운 포맷이 아닌 일반 텍스트로 작성합니다.  
선택 쿼리만 생성합니다. 질문으로 인해 삽입, 업데이트 또는 삭제가 발생하거나 쿼리가  
어떤 식으로든 DDL을 변경하는 경우 작업이 지원되지 않는다고 말합니다.  
질문에 대한 답을 얻을 수 없는 경우에는 DDL이 해당 질문에 대한 답을 지원하지 않는다고 말합니다.

QUESTION  
{question}

DDL  
{ddl}

**결과생성 레코드**

```
public record SqlResponse(String sqlQuery, List<Map<String, Object>> results) { }
```

## application.yaml

```
# OpenAI 설정
spring.ai.openai.api-key=${OPENAI_API_KEY}
spring.ai.openai.chat.options.model=gpt-4o

# 파일 업로드 설정
spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=10MB

# 데이터베이스 설정
spring.datasource.url=jdbc:mysql://localhost:3306/movies
spring.datasource.username=root
spring.datasource.password=12345
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# JPA 설정
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect
```

---

**Github**

<https://github.com/parkmaeil/gptsql.git>

---

## 10\_RAG기반 PDF내 문서 유사도 검색

**목표:** RAG (Retrieval-Augmented Generation) 기법을 적용해 데이터베이스에서 유의미한 정보를 검색하고  
신뢰도 높은 응답을 생성한다.



# RAG Tutorial

## Spring AI PDF 문서 유사도 검색

---

# RAG

RAG (Retrieval-Augmented Generation)는 **검색 기반 생성 모델**로, **정보 검색(retrieval)**과 **언어 생성(generation)**을 결합한 기술입니다. 이 기술은 주로 대규모 언어 모델(LLM)이 특정 질문에 대해 더 정확하고 정보에 근거한 답변을 생성할 수 있도록 도와줍니다.

RAG는 크게 두 가지 주요 단계로 나눌 수 있습니다:

## 1. 정보 검색 (Retrieval):

1. 사용자가 입력한 질문에 대해 관련된 정보를 외부 데이터베이스나 문서에서 검색합니다.
2. 이 검색은 특정 질의와 관련된 **문서나 텍스트 조각**을 찾기 위해 **벡터 검색(Vector Search)** 기술을 사용할 수 있습니다. 예를 들어, PGvector나 FAISS와 같은 검색 엔진을 사용하여, 데이터베이스에 저장된 문서들 중 관련도가 높은 정보를 검색합니다.

## 2. 언어 생성 (Generation):

1. 검색된 정보를 바탕으로, **대형 언어 모델(LLM)**이 사용자에게 더 자연스러운 형태의 답변을 생성합니다.
2. 이 과정에서 모델은 단순히 검색된 정보를 나열하는 것이 아니라, 검색된 정보를 **통합**하여 **적절한 텍스트**로 생성합니다. 이를 통해 보다 정확하고 구체적인 답변을 제공할 수 있습니다.

## RAG(Retrieval-Augmented Generation)

질문을 입력하세요:

CB인사이트 분석 결과, 2024년 3분기 벤처 투자 몇 %가 AI 스타트업에 집중하는 거야

질문 제출

답변:

2024년 3분기 벤처 투자 중 31%가 AI 스타트업에 집중되었습니다.

## RAG(Retrieval-Augmented Generation)

질문을 입력하세요:

다이스 조사 중에서 AI 전문가의 몇 %가 2025년 중 이직을 고려한다고 했어?

질문 제출

답변:

AI 전문가의 73%가 2025년 중 이직을 고려하고 있다고 했습니다.

# Spring Boot Project



<b>Project</b>	<b>Language</b>		
<input checked="" type="radio"/> Gradle - Groovy	<input type="radio"/> Gradle - Kotlin	<input checked="" type="radio"/> Java	<input type="radio"/> Kotlin
<input type="radio"/> Maven	<input type="radio"/> Groovy		
<b>Spring Boot</b>			
<input type="radio"/> 3.4.0 (SNAPSHOT)	<input type="radio"/> 3.4.0 (RC1)	<input type="radio"/> 3.3.6 (SNAPSHOT)	<input checked="" type="radio"/> 3.3.5
<input type="radio"/> 3.2.12 (SNAPSHOT)	<input type="radio"/> 3.2.11	 	
<b>Project Metadata</b>			
Group	com.example		
Artifact	pdfrag		
Name	pdfrag		
Description	Demo project for Spring Boot		
Package name	com.example.pdfrag		
Packaging	<input checked="" type="radio"/> Jar	<input type="radio"/> War	
Java	<input type="radio"/> 23	<input checked="" type="radio"/> 21	<input type="radio"/> 17

## Dependencies

[ADD DEPENDENCIES... CTRL + B](#)

### Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

### OpenAI AI

Spring AI support for ChatGPT, the AI language model and DALL-E, the Image generation model from OpenAI.

### Docker Compose Support DEVELOPER TOOLS

Provides docker compose support for enhanced development experience.

### Thymeleaf TEMPLATE ENGINES

A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

### Lombok DEVELOPER TOOLS

Java annotation library which helps to reduce boilerplate code.

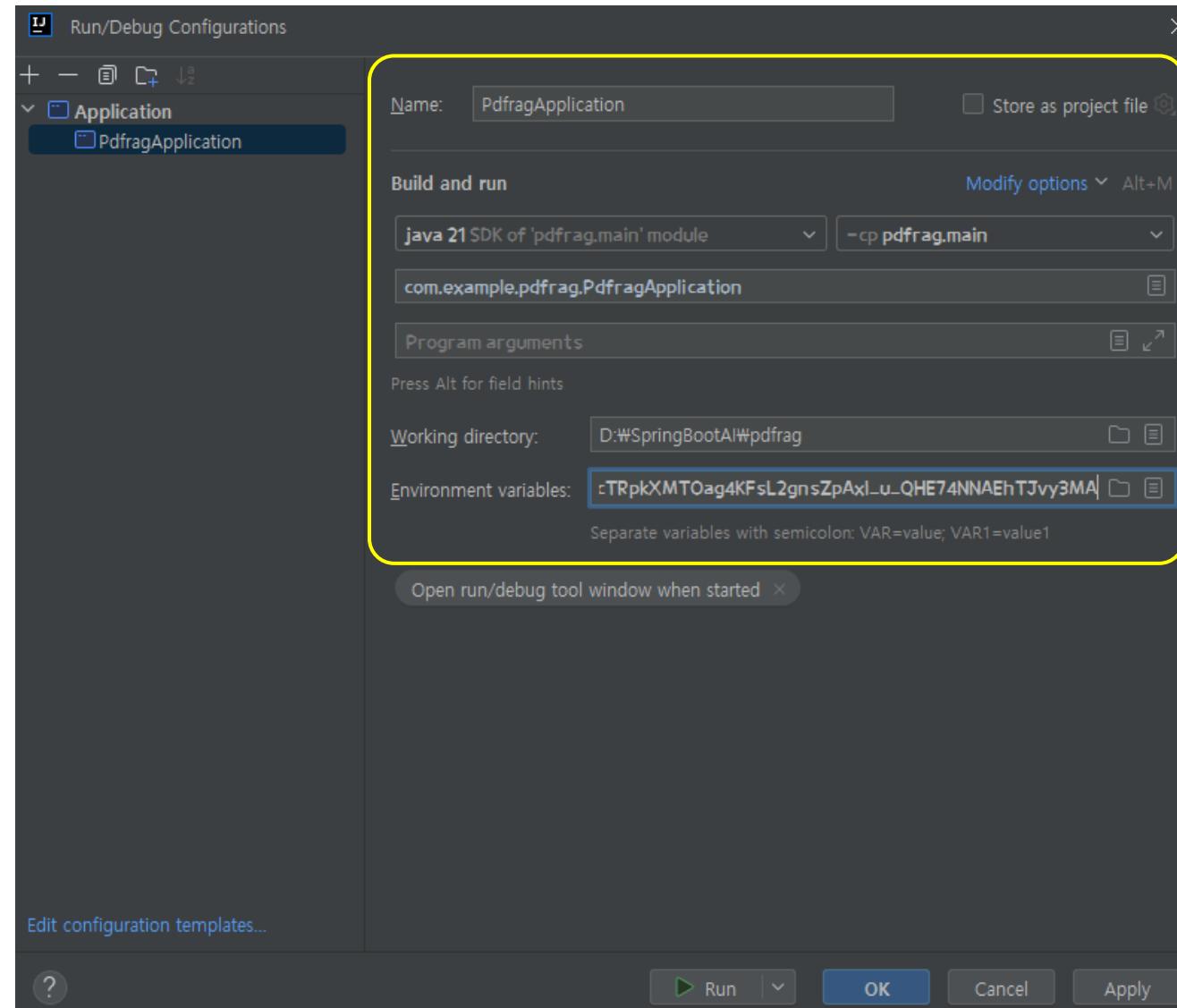
### PDF Document Reader AI

Spring AI PDF document reader. It uses Apache PdfBox to extract text from PDF documents and converting them into a list of Spring AI Document objects.

### PGvector Vector Database AI

Spring AI vector database support for PGvector. It is an open-source extension for PostgreSQL that enables storing and searching over machine learning-generated embeddings.

# OpenAI API Key Setting



# Docker desktop Running

The screenshot shows the Docker Desktop application window. The title bar includes the Docker logo, the text "docker desktop PERSONAL", and a search bar with the placeholder "Search for images, containers, volume...". Below the search bar are several icons: a shield (Security), a gear (Settings), a three-dot menu, a blue circle with a white letter "B" (Background), a minus sign (Close Tab), a square (New Window), and an "X" (Close). The main area is titled "Containers" with a "Give feedback" link. On the left, a sidebar lists "Containers" (selected), "Images", "Volumes", "Builds", "Docker Scout", and "Extensions". The central content area displays "Container CPU usage" and "Container memory usage", both stating "No containers are running." There is a "Show charts" link. Below these are a search bar containing "sha256:19e5069a7c47a6", a filter icon, and a toggle switch labeled "Only show running containers". A table header row includes columns for "Name", "Container ID", "Image", "Port(s)", "CPU (%)", and "Actions". At the bottom, the message "No containers found" is displayed.

Name	Container ID	Image	Port(s)	CPU (%)	Actions
------	--------------	-------	---------	---------	---------

No containers found

## Docker Compose & application.yml setting

### - compose.yaml

**Docker Compose**를 사용하여 **PostgreSQL** 데이터베이스에 pgvector 확장을 설치하고 구성하는 설정입니다.

이 설정은 **Spring Boot** 애플리케이션과 연동되어 사용할 수 있도록 작성되었습니다.

Spring Boot 개발 환경에서 PGvector를 사용할 때 **Docker Compose**나 Testcontainers를 활용하여 PostgreSQL 및 pgvector 확장을 자동으로 설정하는 방법에 대한 설명입니다.  
이를 통해 개발자가 별도로 **데이터베이스 설정을 구성할 필요가 없게 됩니다.**

```
services:
  pgvector:
    image: 'pgvector/pgvector:pg16'
    environment:
      - 'POSTGRES_DB=markets'
      - 'POSTGRES_PASSWORD=password'
      - 'POSTGRES_USER=user'
    labels:
      - "org.springframework.boot.service-connection=postgres"
    ports:
      - '5432:5432'
```

호스트의 5432 포트를 컨테이너의 5432 포트에 매핑

Docker Compose에서 컨테이너를 정의  
**pgvector**라는 단일 서비스가 정의

데이터베이스 초기 설정

Spring Boot는 이 레이블을 사용하여 pgvector 서비스와 자동으로 연결합니다.  
spring.datasource.url,  
spring.datasource.username,  
spring.datasource.password 설정을 별도로 지정하지 않아도 됩니다.

### application.yml

```
spring:
  ai:
    vectorstore:
      pgvector:
        index-type: HNSW
        distance-type: COSINE_DISTANCE
        dimensions: 1536
        initialize-schema: true
```

Default value

## - application.yml

**Spring Boot** 애플리케이션에서 **PGvector** 벡터 스토어를 구성하기 위한 application.yml 파일의 예시입니다.

이 설정은 PostgreSQL 데이터베이스를 사용하여 PGvector를 활용하기 위한 설정입니다.

**Spring Boot Dev Services 또는 Docker Compose/Testcontainers를 사용하지 않고,  
로컬 PostgreSQL 데이터베이스를 직접 연결하여 사용하는 경우**

```
spring:
  datasource:
    url: jdbc:postgresql://localhost:5432/postgres
    username: postgres
    password: postgres
  ai:
    vectorstore:
      pgvector:
        index-type: HNSW
        distance-type: COSINE_DISTANCE
        dimensions: 1536
        initialize-schema: true
```

PostgreSQL 데이터베이스에 연결하기 위한 JDBC URL

- localhost: 데이터베이스 서버 주소 (로컬 서버를 의미).
- 5432: PostgreSQL의 기본 포트 번호.
- postgres: 데이터베이스 이름.

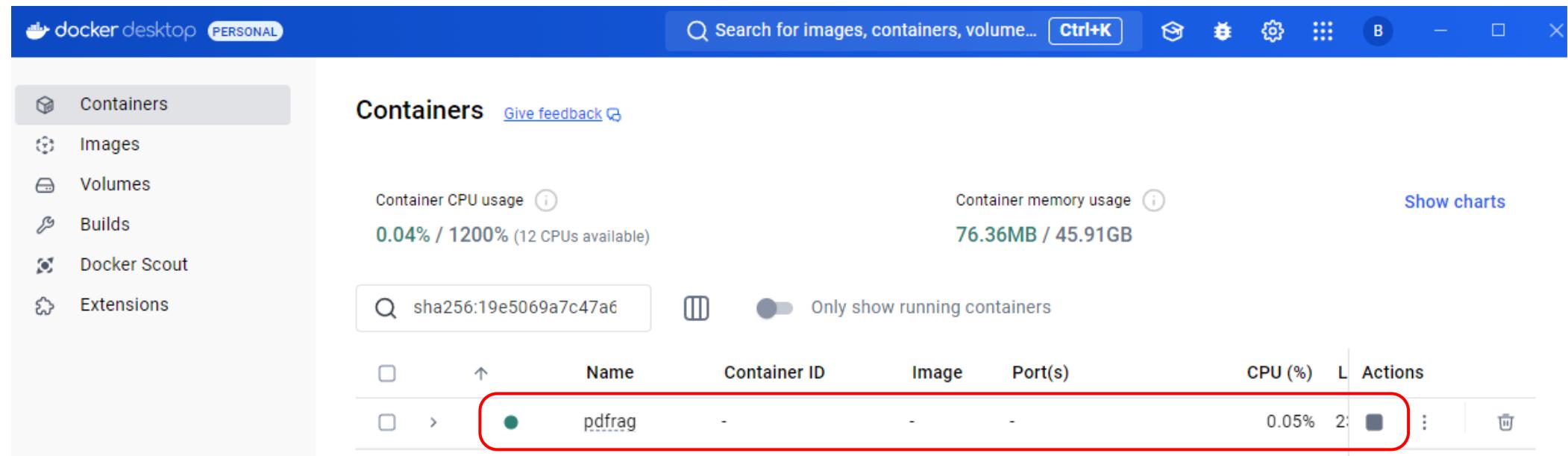
### Spring AI 벡터 스토어 설정

- **index-type**: 벡터 검색에 사용할 인덱스 유형.
  - HNSW (Hierarchical Navigable Small World): 고성능 근사 근접 이웃 검색을 위한 인덱스 방식입니다.
- **distance-type**: 벡터 간의 유사도를 측정할 때 사용할 거리 계산 방식.
  - COSINE\_DISTANCE: 코사인 거리. 벡터의 방향을 기준으로 유사도를 측정합니다. (값이 작을수록 유사함)
- **dimensions**: 벡터의 차원 수.
  - 1536: OpenAI의 GPT 모델에서 생성된 임베딩의 기본 벡터 차원입니다.

```
docker run -it --rm --name postgres -p 5432:5432 -e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=postgres pgvector/pgvector
```

# Project Running

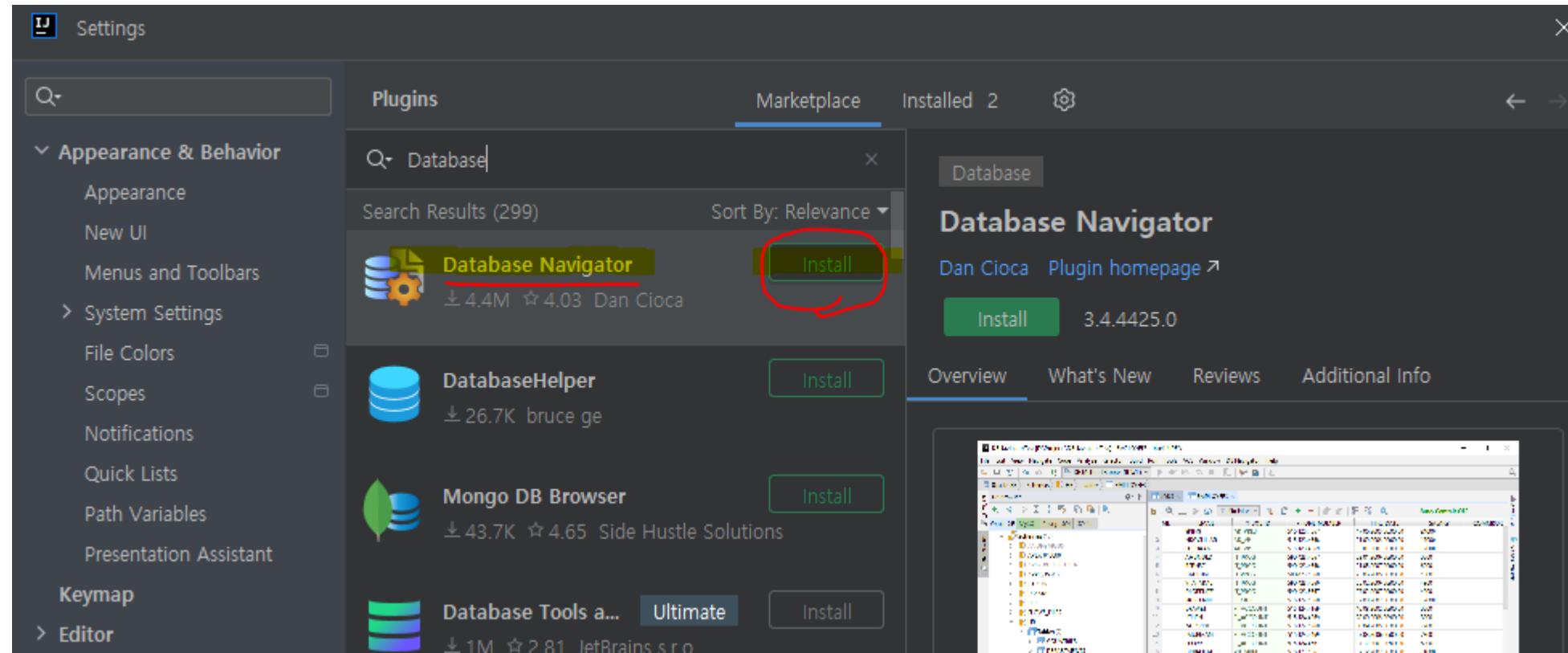
- Pgvector Database 실행



The screenshot shows the Docker Desktop interface. The left sidebar has 'Containers' selected. The main area displays container statistics: Container CPU usage (0.04% / 1200%) and Container memory usage (76.36MB / 45.91GB). A search bar at the top contains the SHA256 hash 'sha256:19e5069a7c47a6'. Below the search bar is a toggle switch for 'Only show running containers'. The table lists one container:

	Name	Container ID	Image	Port(s)	CPU (%)	L	Actions
	pdffrag	-	-	-	0.05%	2:	 

## Database Navigator Plugins 설치



## Database Connection 확인

The screenshot shows the DB Browser interface for MySQL. On the left, a sidebar displays a tree structure with a 'Connection' node selected. A yellow box highlights the '+' button next to the search icon. On the right, a detailed configuration dialog for a PostgreSQL connection is open. The 'Database' tab is selected. The configuration fields are as follows:

- Name:** Connection
- Description:** (empty)
- Connection**
  - Config type:** Database (selected from dropdown)
  - Host:** localhost
  - Database:** mydatabase
  - URL:** jdbc:postgresql://localhost:5432/mydatabase
  - Authentication**
    - Authentication:** User / Password (selected from dropdown)
    - User:** myuser
    - Password:** (redacted)
- Drivers:** Driver source: Bundled library

A yellow arrow points from the '+' button in the sidebar to the 'Name' field in the configuration dialog. A yellow box encloses the 'Connection' section of the configuration dialog. To the right of the configuration dialog, a terminal window displays a Docker service configuration:

```
services:  
pgvector:  
  image: 'pgvector/pgvector:pg16'  
environment:  
  - 'POSTGRES_DB=mydatabase'  
  - 'POSTGRES_PASSWORD=secret'  
  - 'POSTGRES_USER=myuser'  
labels:  
  - "org.springframework.boot.service-connection=postgres"  
ports:  
  - '5432:5432'
```

## vector\_store table 생성 확인

The screenshot shows the DB Browser interface with the following details:

- Left Panel (Tree View):** Shows the database structure:
  - Connection
  - Schemas (4):
    - information\_schema
    - pg\_catalog
    - pg\_toast
    - public
  - Tables (1):
    - vector\_store
- Center Panel (Table View):** Displays the columns of the vector\_store table:

	id	content	metadata	embedding
--	----	---------	----------	-----------
- Right Panel (Text View):** Shows the SQL code used to create the table:

```
CREATE EXTENSION IF NOT EXISTS vector;
CREATE EXTENSION IF NOT EXISTS hstore;
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

CREATE TABLE IF NOT EXISTS vector_store (
    id uuid DEFAULT uuid_generate_v4() PRIMARY KEY,
    content text,
    metadata json,
    embedding vector(1536) // 1536 is the default embedding dimension
);

CREATE INDEX ON vector_store USING HNSW (embedding vector_cosine_ops);
```

## PDF 분석 자료 다운로드

<https://spri.kr/posts?code=AI-Brief>



### AI 브리프

The screenshot displays four versions of the 'SPRI AI Brief' report, each with a different release date and a corresponding PDF file link:

- 2024년 11월호**: A thumbnail of the report cover with a red border around the title area.
- 2024년 10월호**: A thumbnail showing the file structure under 'resources' and a blue highlighted link to the PDF file: [SPRI AI Brief\\_11월호\\_산업동향\\_F.pdf](#).
- 2024년 9월호**: A thumbnail of the report cover.
- 2024년 8월호**: A thumbnail of the report cover.

A green arrow points upwards from the 2024년 10월호 thumbnail towards the 2024년 11월호 thumbnail, indicating the progression of time.

# 응답의 결과를 더 신뢰하는 RAG



Dice survey, what percentage of AI experts said they would consider changing jobs in 2025?

## 질문 내용 ▾

다이스 조사, AI 전문가의 몇 %가 2025년 중 이직 고려한다고 했어?



현재까지 공개된 자료에 따르면, AI 전문가들이 2025년에 이직을 고려하는 비율에 대한 다이스 (Dice)의 설문조사 결과는 확인되지 않습니다. 다이스는 2024년 기술 전문가들을 대상으로 한 설문 조사에서, 지난해 대비 해고된 기술 전문가의 비율이 60% 증가했으며, 실직을 경험한 기술 전문가 중 82%가 즉시 새로운 일자리를 찾기 시작했다고 보고했습니다. 그러나 이 조사에서는 AI 전문가들의 2025년 이직 의향에 대한 구체적인 수치는 제공되지 않았습니다.

**RAG(Retrieval-Augmented Generation)** 기술을 사용하면 AI의 헛소리(hallucination)를 줄이는 데 도움이 됩니다.

RAG는 외부 데이터베이스나 문서 저장소에서 신뢰할 수 있는 정보를 검색하고, 이를 기반으로 답변을 생성하는 방식입니다. 이렇게 하면 모델이 맥락에 맞는 정보를 활용해 응답을 제공할 수 있어 헛소리를 줄일 수 있습니다.

SPRI AI Brief |  
2024-11월호

다이스 조사, AI 전문가의 73%는 2025년 중 이직 고려

## KEY Contents

- 다이스에 따르면, AI 전문가의 73%는 2025년 이직을 계획 중이며, 58%는 2024년 중 현재보다 더 나은 일자리를 찾을 자신이 있다고 응답해 여타 기술 전문가 대비 직업 전망을 낙관
- AI 전문가들은 여타 기술 전문가 대비 AI 도구 사용에도 적극적이며, 업무에 생성AI가 상당한 영향을 미친다는 응답도 36%로 여타 기술 전문가(22%) 대비 높은 수치를 기록

OpenAI가 잘못된 답변을 제공할 때 이를

\*\*"hallucination"\*\*이라고 부릅니다.

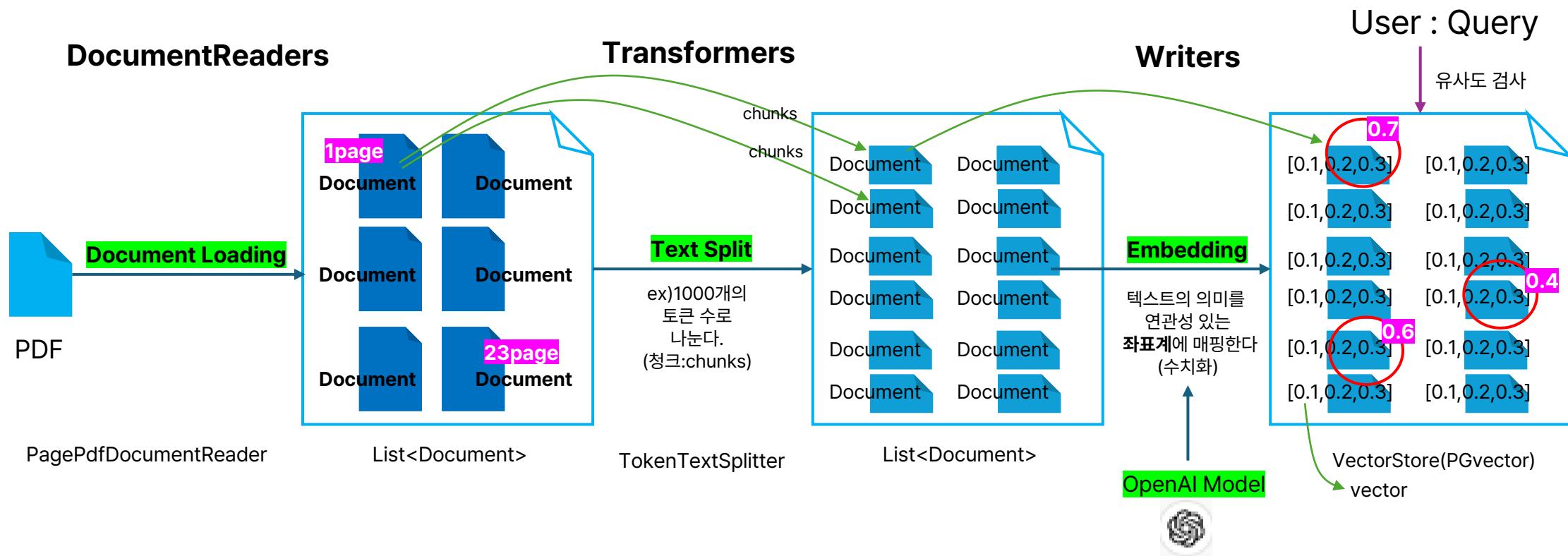
AI가 실제로 존재하지 않는 정보나 잘못된 사실을 제시하는 오류를 의미합니다. 이 현상은 특히 대형 언어 모델(LLM)에서 흔하게 발생하며, 모델이 자신감 있게 잘못된 답변을 생성할 때 이를 "hallucination"이라고 표현합니다.

Hallucination : 헛소리, 환각

PDF

# PDF Embedding

implementation 'org.springframework.ai:spring-ai-pdf-document-reader'



## 텍스트 임베딩(PDF)

```
public TokenTextSplitter(
    int defaultChunkSize,      // 기본 청크 크기
    int minChunkSizeChars,     // 청크의 최소 문자 크기
    int minChunkLengthToEmbed, // 임베딩에 포함되기 위한 청크의 최소 길이
    int maxNumChunks,         // 생성할 수 있는 청크의 최대 개수
    boolean keepSeparator     // 분리 기호를 청크에 포함할지 여부
)
```

### 1. defaultChunkSize (기본 청크 크기)

- 텍스트를 나눌 때, 한 청크의 기본 크기(토큰 개수)입니다.
- 예를 들어, 1000으로 설정하면 각 청크가 **1000개의 토큰**을 포함하도록 기본적으로 분할됩니다.

### 2. minChunkSizeChars (청크의 최소 문자 크기)

- 청크로 나누어질 때, 각 청크의 최소 문자 수를 지정합니다.
- 예를 들어, 400으로 설정하면 각 청크는 **최소 400자 이상**이 됩니다.

### 3. minChunkLengthToEmbed (임베딩에 포함되기 위한 청크의 최소 길이)

- 청크가 임베딩 작업에 포함되기 위해 필요한 최소 길이(문자 수)입니다.
- 예를 들어, 10으로 설정하면 **10자 미만인 청크는 임베딩에 포함되지 않습니다.**

### 4. maxNumChunks (생성할 수 있는 청크의 최대 개수)

- 텍스트를 분할한 후 생성할 수 있는 청크의 최대 개수입니다.
- 예를 들어, 5000으로 설정하면 **최대 5000개의 청크**까지만 생성됩니다.

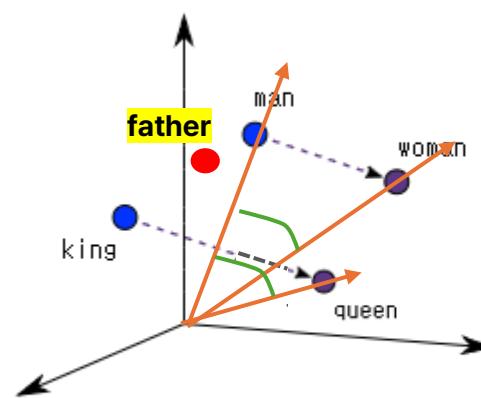
### 5. keepSeparator (분리 기호 유지 여부)

- true: 분리 기호(예: 공백, 줄바꿈 문자 등)가 각 청크에 포함됩니다.
- false: 분리 기호를 제외하고 청크를 생성합니다.

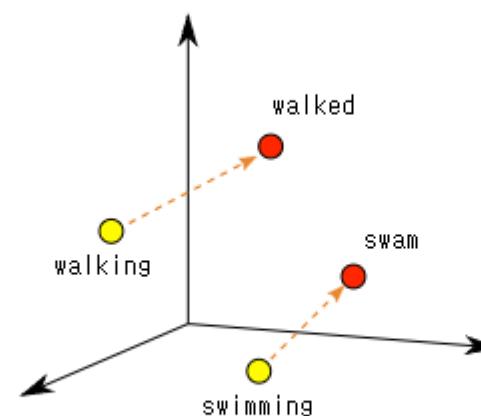
# 임베딩(Embedding)

단어를 3차원 공간에 수치로 표시하는 방법

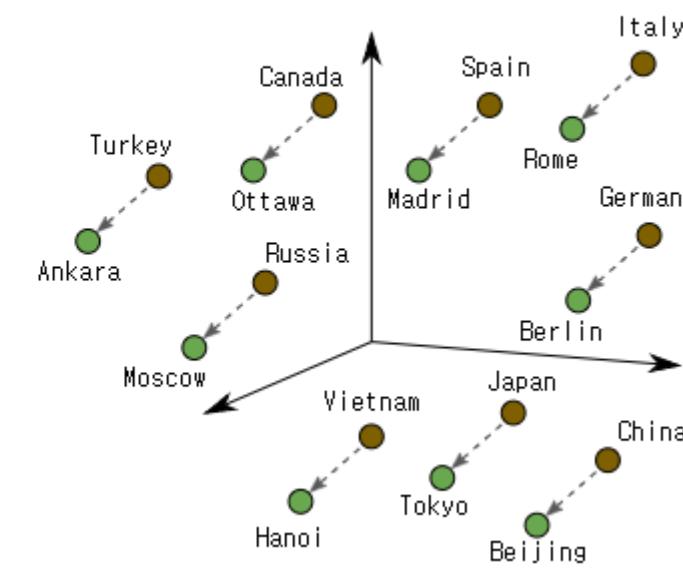
$1 \times 1536$  차원에 표현



단어 간의 의미적 관계

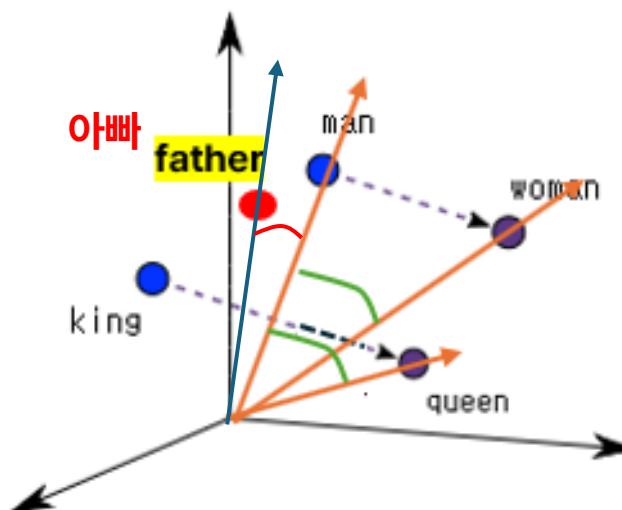


동사의 시제 변화에 따른 관계



나라와 수도 관계

# 코사인 유사도



코사인 유사도로 실제 확인해야 함:

예를 들어, 임베딩 모델에서 각 단어의 벡터를 다음과 같이 정의했다고 가정합시다:

- man : [0.5, 0.6, 0.7]
- queen : [0.4, 0.7, 0.6]
- woman : [0.5, 0.8, 0.7]

이 경우:

1. man ↔ woman 의 코사인 유사도:

$$\cos \theta = \frac{\vec{m}an \cdot \vec{w}oman}{\|\vec{m}an\| \|\vec{w}oman\|}$$

계산 결과: 0.99 (높은 유사도)

2. man ↔ queen 의 코사인 유사도:

$$\cos \theta = \frac{\vec{m}an \cdot \vec{q}ueen}{\|\vec{m}an\| \|\vec{q}ueen\|}$$

계산 결과: 0.95 (조금 낮은 유사도)

이처럼 계산 결과를 통해 실제로 man ↔ woman 이 더 가까운 관계라는 것을 알 수 있습니다.

# 코사인 유사도 계산 예시

## 코사인 유사도 공식

$$\text{Cosine Similarity} = \cos \theta = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| \|\vec{B}\|}$$

- $\vec{A} \cdot \vec{B}$ : 두 벡터의 내적.
- $\|\vec{A}\|, \|\vec{B}\|$ : 각각의 벡터 크기(노름).

### 1. man ↔ queen 코사인 유사도 계산

벡터:

$$\vec{m}an = [0.5, 0.6, 0.7], \quad \vec{que}en = [0.4, 0.7, 0.6]$$

#### 1.1 내적 계산 ( $\vec{m}an \cdot \vec{que}en$ )

$$\begin{aligned} \vec{m}an \cdot \vec{que}en &= (0.5 \cdot 0.4) + (0.6 \cdot 0.7) + (0.7 \cdot 0.6) \\ &= 0.2 + 0.42 + 0.42 = 1.04 \end{aligned}$$

#### 1.2 벡터 크기 계산 ( $\|\vec{m}an\|, \|\vec{que}en\|$ )

$$\|\vec{m}an\| = \sqrt{(0.5)^2 + (0.6)^2 + (0.7)^2} = \sqrt{0.25 + 0.36 + 0.49} = \sqrt{1.1} \approx 1.0488$$

$$\|\vec{que}en\| = \sqrt{(0.4)^2 + (0.7)^2 + (0.6)^2} = \sqrt{0.16 + 0.49 + 0.36} = \sqrt{1.01} \approx 1.0049$$

#### 1.3 코사인 유사도 계산

$$\cos \theta = \frac{\vec{m}an \cdot \vec{que}en}{\|\vec{m}an\| \|\vec{que}en\|} = \frac{1.04}{1.0488 \cdot 1.0049} \approx \frac{1.04}{1.0536} \approx 0.9867$$

### 2. man ↔ woman 코사인 유사도 계산

벡터:

$$\vec{m}an = [0.5, 0.6, 0.7], \quad \vec{wo}man = [0.5, 0.8, 0.7]$$

#### 2.1 내적 계산 ( $\vec{m}an \cdot \vec{wo}man$ )

$$\begin{aligned} \vec{m}an \cdot \vec{wo}man &= (0.5 \cdot 0.5) + (0.6 \cdot 0.8) + (0.7 \cdot 0.7) \\ &= 0.25 + 0.48 + 0.49 = 1.22 \end{aligned}$$

#### 2.2 벡터 크기 계산 ( $\|\vec{m}an\|, \|\vec{wo}man\|$ )

$$\|\vec{wo}man\| = \sqrt{(0.5)^2 + (0.8)^2 + (0.7)^2} = \sqrt{0.25 + 0.64 + 0.49} = \sqrt{1.38} \approx 1.1749$$

(이미 계산된  $\|\vec{m}an\| = 1.0488$  사용)

#### 2.3 코사인 유사도 계산

$$\cos \theta = \frac{\vec{m}an \cdot \vec{wo}man}{\|\vec{m}an\| \|\vec{wo}man\|} = \frac{1.22}{1.0488 \cdot 1.1749} \approx \frac{1.22}{1.2316} \approx 0.9902$$

## 텍스트 임베딩(PDF) 소스 코드

### 1. PdfDocumentReaderConfig.builder()

PdfDocumentReaderConfig의 빌더를 생성합니다.

### 2. .withPageTopMargin(0)

PDF 페이지의 상단 여백을 설정합니다.

0으로 설정했으므로 페이지 상단의 여백을 제거하지 않습니다.

### 3. .withPageExtractedTextFormatter(ExtractedTextFormatter.builder()...)

페이지에서 추출된 텍스트를 처리하기 위한 포맷터를 설정합니다.

내부에서 ExtractedTextFormatter.builder()를 사용해 텍스트 포맷팅을 세부적으로 지정합니다.

### 4. .withNumberOfTopTextLinesToDelete(0)

페이지의 상단에서 삭제할 텍스트 줄 수를 설정합니다.

0으로 설정했으므로 페이지 상단에서 삭제되는 텍스트 줄이 없습니다.

PDF 페이지에서 헤더와 같은 불필요한 정보를 제거할 때 사용됩니다.

### 5. .withPagesPerDocument(1)

PDF 문서를 읽을 때 한 번에 처리할 페이지 수를 설정합니다.

1로 설정했으므로 PDF를 한 페이지씩 분리하여 처리합니다.

이 설정은 PDF를 청크 단위로 나누어 처리하는 경우에 유용합니다.

### 6. .build()

빌더 패턴의 최종 단계로, 설정이 완료된 PdfDocumentReaderConfig 객체를 생성합니다.

이 객체는 PDF를 처리하는 다른 클래스에서 사용됩니다.

```
@Configuration
public class DataLoader {
```

```
    private final VectorStore vectorStore;
    private final JdbcClient jdbcClient;
```

```
    // # 0. PDF 경로(resources 아래)
```

```
    @Value("classpath:/SPRI AI Brief_11월호_산업동향_F.pdf")
```

```
    private Resource pdfResource;
```

```
    public DataLoader(VectorStore vectorStore, JdbcClient jdbcClient) {
```

```
        this.vectorStore = vectorStore;
```

```
        this.jdbcClient = jdbcClient;
```

```
}
```

```
@PostConstruct
```

```
public void init(){
```

```
    Integer count=jdbcClient.sql("select count(*) from vector_store")
```

```
        .query(Integer.class)
```

```
        .single();
```

```
    System.out.println("No of Records in the PG Vector Store="+count);
```

```
    if(count==0){
```

```
        System.out.println("Loading.....");
```

```
        // PDF Reader
```

```
        PdfDocumentReaderConfig config=PdfDocumentReaderConfig.builder()
```

```
            .withPageTopMargin(0)
```

```
            .withPageExtractedTextFormatter(ExtractedTextFormatter.builder()
```

```
                .withNumberOfTopTextLinesToDelete(0)
```

```
                .build()
```

```
            .withPagesPerDocument(1)
```

```
            .build();
```

```
// # 1.단계 : 문서로드(Load Documents)
```

```
PagePdfDocumentReader pdfReader =new PagePdfDocumentReader(pdfResource,config);
```

```
List<Document> documents=pdfReader.get();
```

```
// System.out.println(documents.toString()); // 25
```

```
// 1000글자 단위로 자른다.
```

```
// # 2.단계 : 문서분할(Split Documents)
```

```
TokenTextSplitter splitter = new TokenTextSplitter(1000, 400, 10, 5000, true);
```

```
List<Document> splitDocuments=splitter.apply(documents);
```

```
System.out.println(splitDocuments.size()); // 45
```

```
System.out.println(splitDocuments.get(0)); // 25
```

```
// # 3.단계 : 임베딩(Embedding) -> 4.단계 : DB에 저장(벡터스토어 생성)
```

```
vectorStore.accept(splitDocuments); // OpenAI 임베딩을 거친다.
```

```
System.out.println("Application is ready to Serve the Requests");
```

```
}
```

```
}
```

## 텍스트 임베딩(PDF) 소스 코드

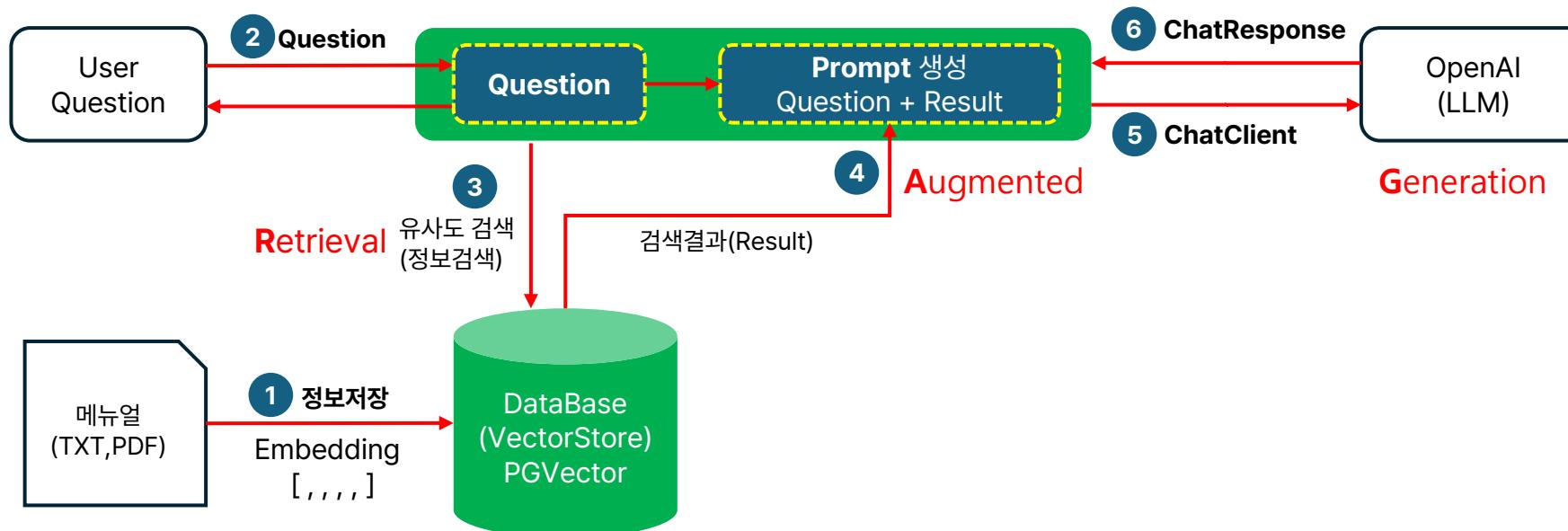
DB Browser for PostgreSQL

Connection: Auto-Commit OFF - connected

	id	content	metadata	embedding
1	5aad4fe3-cbb0-47	2024년 11월호	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 1}	[ -0.012479925, -0.017742543, 0.00839404, -0.000000000, 0.000000000, 0.000000000 ]
2	404bdb83-b429-4	2024년 11월호	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 2}	[ 0.004036141, -0.039374433, 0.006219252, -0.000000000, 0.000000000, 0.000000000 ]
3	eba57a89-c889-47	기술/연구	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 2}	[ -0.013097304, -0.02673106, 0.0105000865, -0.000000000, 0.000000000, 0.000000000 ]
4	bde8e629-0332-4c	I . 인공지능	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 3}	[ -0.0049545593, -0.025357759, -0.01529359, -0.000000000, 0.000000000, 0.000000000 ]
5	4bdb7540-0465-4	1. 정책/법제	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 4}	[ 0.0019456303, -0.012915532, 0.025499046, -0.000000000, 0.000000000, 0.000000000 ]
6	943fe426-140d-4b	n 보고서에 따르면	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 4}	[ -0.01883755, -0.014580106, 0.03377135, -0.000000000, 0.000000000, 0.000000000 ]
7	56279a62-8ae5-4c	인한 불평등을	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 4}	[ -0.012288317, -0.005216305, 0.018478377, -0.000000000, 0.000000000, 0.000000000 ]
8	3a2d341d-1d3a-4c	SPRI AI Brief I	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 5}	[ -0.0066778883, -0.045688, -0.005864899, -0.000000000, 0.000000000, 0.000000000 ]
9	8cc80891-e6c3-48	☞ 출처 : CB Ins	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 9}	[ 0.007699629, -0.023417726, 0.002404862, -0.000000000, 0.000000000, 0.000000000 ]
10	19901bc6-bc68-49	· 정부 기관의 개인	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 5}	[ 0.0104505895, -0.029442016, 0.007970564, -0.000000000, 0.000000000, 0.000000000 ]
11	1731d014-f159-49	· 각 정부 기관	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 5}	[ -0.0090328595, -0.03433692, -0.020690538, -0.000000000, 0.000000000, 0.000000000 ]
12	c0eeae80-450b-49	1. 정책/법제	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 6}	[ 0.008866218, -0.027195036, 0.03451066, -0.000000000, 0.000000000, 0.000000000 ]
13	90778167-6971-4	· 기계번역과 같은	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 6}	[ -0.0030954126, -0.037073154, 0.018889034, -0.000000000, 0.000000000, 0.000000000 ]
14	4564dd55-313d-4	SPRI AI Brief I	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 7}	[ 0.013994264, -0.036765046, 0.005321433, -0.000000000, 0.000000000, 0.000000000 ]
15	ac84273a-9b14-47	1. 정책/법제	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 10}	[ -0.018115526, -0.02821557, 0.00942088, -0.000000000, 0.000000000, 0.000000000 ]
16	df7982e7-2f87-4a	부문 디지털화를 우	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 7}	[ 0.002597297, -0.037299212, 0.008280007, -0.000000000, 0.000000000, 0.000000000 ]
17	9d83e231-81f5-4c	10.15.	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 7}	[ 0.002071164, -0.014654876, 0.0112169795, -0.000000000, 0.000000000, 0.000000000 ]
18	598044a5-096f-41	1. 정책/법제	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 8}	[ 0.008349125, -0.02365924, 0.01942035, -0.010000000, 0.000000000, 0.000000000 ]
19	e3047053-34bf-44	· 기존 규제 당국이	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 8}	[ 0.008073683, -0.02373649, 0.020829145, -0.000000000, 0.000000000, 0.000000000 ]
20	d726879c-c1d3-49	광범위한 도입	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 8}	[ 0.009461189, -0.041154478, 0.01460889, -0.000000000, 0.000000000, 0.000000000 ]
21	f731357e-3597-44	SPRI AI Brief I	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 9}	[ -0.0131700905, -0.043792572, 0.016146153, -0.000000000, 0.000000000, 0.000000000 ]
22	3ec3c3f5-2a0b-48	기업이며, 다음	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 9}	[ -0.0035356213, -0.06084834, 0.017787047, -0.000000000, 0.000000000, 0.000000000 ]
57	8deb563a-de24-44	SPRI AI Brief I	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 23}	[ -0.0115309395, -0.027584614, 0.0077755745, -0.000000000, 0.000000000, 0.000000000 ]
58	6df6221b-e81f-47	평가됐으며,	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 23}	[ -0.0024448135, -0.017353974, 0.016305476, -0.000000000, 0.000000000, 0.000000000 ]
59	dc9ae187-6458-4a	· 생성AI는 직	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 23}	[ -0.027716558, -0.01518642, -0.0024584024, -0.000000000, 0.000000000, 0.000000000 ]
60	f3e12e43-ddae-4d	II . 주요	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 24}	[ -0.016644033, -0.026326379, 0.031394273, -0.000000000, 0.000000000, 0.000000000 ]
61	b96f5c8d-5787-41	홈페이지 : https://	{"file name": "SPRI AI Brief 11월호 산업동향 F.pdf", "page number": 25}	[ -0.0068561058, -0.013304189, -0.0025749131, -0.000000000, 0.000000000, 0.000000000 ]

# RAG?

RAG는 **Retrieval-Augmented Generation**의 약자로, 정보 검색과 생성 AI를 결합한 기술입니다. RAG는 대규모 언어 모델(LLM)과 데이터베이스를 조합하여, 질문에 대해 더 정확하고 신뢰할 수 있는 답변을 생성하는 데 사용됩니다.

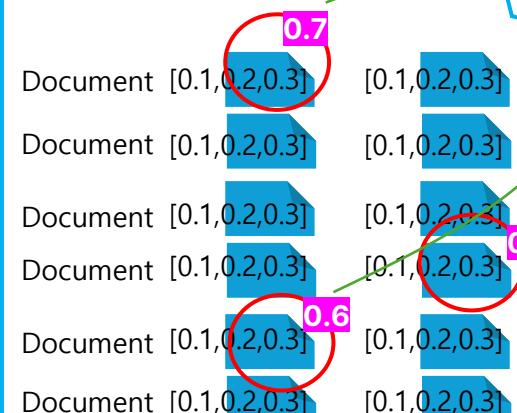


# RAG 유사도 검사

User : Query  
: 다이스 조사, AI 전문가의 몇%가 2025년 중 이직을 고려 있다고 했어?

Embedding 0.7  
List<Document> documents =  
vectorStore.similaritySearch(SearchRequest  
.query(question)  
.withTopK(2));

VectorStore(PGvector)

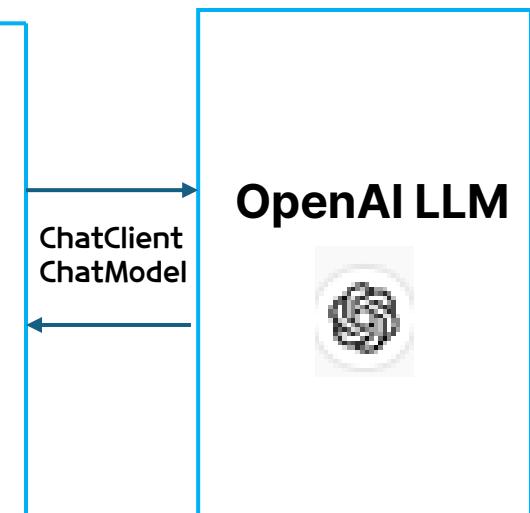


Prompt 만들기

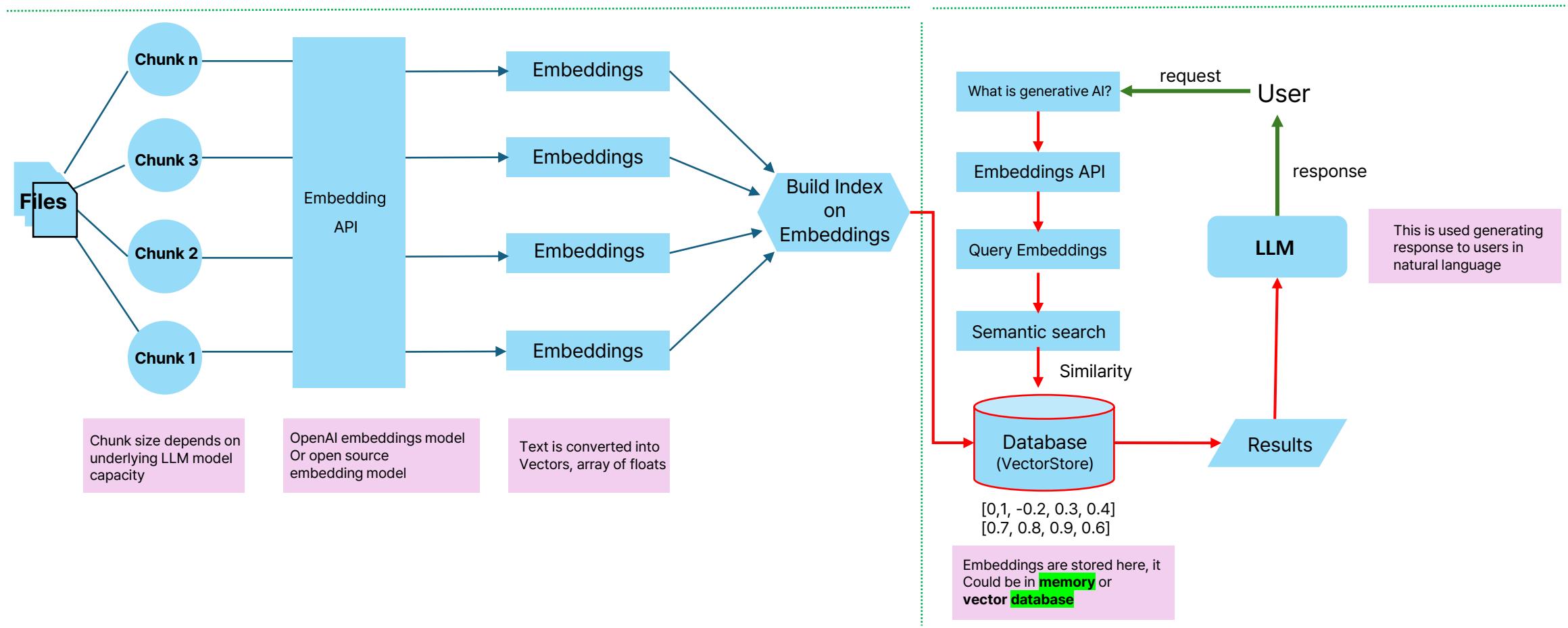
```
private String prompt = """""
이 질문에 답변하는 것입니다.
질문에 정확하게 답변하기 위해 문서에 있는 정보를 사용해야 합니다.
만약 정보가 부족하거나 문서에서 답을 찾을 수 없다면,
알지 못한다고 간단히 답변하세요.

질문:
{input}

문서 :
{documents}
....;
```



# RAG Process



# 유사도 검사

@RestController

```
public class PDFController {

    private final ChatModel chatModel; // OpenAI
    private final VectorStore vectorStore;

    private String prompt = """"
```

질문에 답변하는 것입니다. 질문에 정확하게 답변하기 위해 문서에 있는 정보를 사용해야 합니다.  
만약 정보가 부족하거나 문서에서 답을 찾을 수 없다면, 알지 못한다고 간단히 답변하세요.

질문:

{input}

문서 :

{documents}

**Prompt 만들기**  
**<프롬프트 엔지니어링>**

""";

```
public PDFController(ChatModel chatModel, VectorStore vectorStore) {
    this.chatModel = chatModel;
    this.vectorStore = vectorStore;
}

@GetMapping("/api/answer")
public String simplify(String question) {

    PromptTemplate template
        = new PromptTemplate(prompt);
    Map<String, Object> promptsParameters = new HashMap<>();
    PromptsParameters.put("input", question);
    PromptsParameters.put("documents", findSimilarData(question));

    return chatModel
        .call(template.create(promptsParameters))
        .getResult()
        .getOutput()
        .getContent();
}

private String findSimilarData(String question) {
    List<Document> documents =
        vectorStore.similaritySearch(SearchRequest
            .query(question)
            .withTopK(5));

    return documents
        .stream()
        .map(document -> document.getContent().toString())
        .collect(Collectors.joining());
}
```

# View Page

<http://localhost:8080/pdfRag>

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>RAG 질문 클라이언트</title>
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
        rel="stylesheet">
</head>
<body>
<div class="container mt-5">
    <h2 class="text-center">RAG(Retrieval-Augmented Generation) </h2>
    <div class="form-group">
        <label for="question">질문을 입력하세요:</label>
        <input type="text" id="question" class="form-control" placeholder="질문을 입력하세요">
    </div>
    <button class="btn btn-primary" onclick="submitQuestion()">질문 제출</button>
    <div class="mt-4">
        <h4>답변:</h4>
        <div id="loading" class="text-info" style="display: none;">데이터가 로딩 중입니다...</div>
        <div id="response" class="border p-3 rounded" style="min-height: 100px;">
            <em>답변이 여기에 표시됩니다.</em>
        </div>
    </div>
</div>
</body>
```

```
<script>
    async function submitQuestion() {
        const question = document.getElementById("question").value;
        const responseDiv = document.getElementById("response");
        const loadingDiv = document.getElementById("loading");

        if (!question) {
            responseDiv.innerHTML = "<span class='text-danger'>질문을 입력해 주세요.</span>";
            return;
        }

        // 로딩 메시지 표시
        loadingDiv.style.display = "block";
        responseDiv.innerHTML = ""; // 기존 응답 지우기

        // Fetch API를 이용하여 서버에 질문 전송
        try {
            const response = await fetch('/api/answer?question=${encodeURIComponent(question)}');
            const result = await response.text();

            // 서버의 응답을 받아와서 표시
            responseDiv.innerText = result ? result : "적절한 답변이 없습니다.";
        } catch (error) {
            console.error("Error fetching answer:", error);
            responseDiv.innerHTML = "<span class='text-danger'>오류가 발생했습니다. 다시 시도해 주세요.</span>";
        } finally {
            // 로딩 메시지 숨기기
            loadingDiv.style.display = "none";
        }
    }
</script>
</body>
</html>
```

# RAG 기술 적용 결과

다이스 조사 중에서 AI 전문가의 몇 %가 2025년 중 이직을 고려한다고 했어?

Dice survey, what percentage of AI experts said they would consider changing jobs in 2025?

질문 내용 ▾

다이스 조사, AI 전문가의 몇 %가 2025년 중 이직 고려한다고 했어?

현재까지 공개된 자료에 따르면, AI 전문가들이 2025년에 이직을 고려하는 비율에 대한 다이스 (Dice)의 설문조사 결과는 확인되지 않습니다. 다이스는 2024년 기술 전문가 대상으로 한 설문 조사에서, 지난해 대비 해고된 기술 전문가의 비율이 60% 증가했으며, 실직률은 2023년 기술 전문가 중 82%가 즉시 새로운 일자리를 찾기 시작했다고 보고했습니다. 그러나 이 조사에서는 AI 전문가들의 2025년 이직 의향에 대한 구체적인 수치는 제공되지 않았습니다.

다이스 조사 중에서 AI 전문가의 몇 %가 2025년 중 이직을 고려한다고 했어?

사이트 4개 검색함 ▾

다이스의 설문 조사에 따르면, AI 전문가의 약 50%가 2025년 안에 이직을 고려하고 있다고 밝혔습니다. 이 수치는 현재의 기술 시장 내 불안정성, 직업 안정성 감소, 그리고 AI 기술 혁신의 빠른 변화로 인한 영향을 반영합니다. 특히 젊은 기술 전문가들 사이에서 이러한 경향이 두드러지며, 많은 이들이 현재 AI 프로젝트의 전략적 중요성에 대한 의구심을 갖고 있거나 더 나은 기회를 찾고자 합니다.

추가적인 정보를 원하시면, [다이스 보고서](#)의 관련 자료를 확인하실 수 있습니다.

SPRI AI Brief |  
2024-11월호

다이스 조사, AI 전문가의 73%는 2025년 중 이직 고려

KEY Contents

- 다이스에 따르면, AI 전문가의 73%는 2025년 이직을 계획 중이며, 58%는 2024년 중 현재보다 더 나은 일자리를 찾을 자신이 있다고 응답해 여타 기술 전문가 대비 직업 전망을 낙관
- AI 전문가들은 여타 기술 전문가 대비 AI 도구 사용에도 적극적이며, 업무에 생성AI가 상당한 영향을 미친다는 응답도 36%로 여타 기술 전문가(22%) 대비 높은 수치를 기록

RAG(Retrieval-Augmented Generation)

질문을 입력하세요:

다이스 조사 중에서 AI 전문가의 몇 %가 2025년 중 이직을 고려한다고 했어?

질문 제출

답변:

다이스 조사에 따르면, AI 전문가의 73%가 2025년 중 이직을 고려하고 있습니다.



<https://github.com/parkmaeil/ragpdf.git>

---

## 11\_ 플롯 유사성 기반 영화 추천 시스템

**목표:** 영화 플롯 데이터를 임베딩하고 벡터 유사도 검색을 통해 사용자의 선호에 맞는 영화를 추천하는 시스템을 구현한다.

---

# RAG(Retrieval-Augmented Generation)

: PGVector 활용

## 1. 플롯 유사성을 기반으로 한 영화 추천 시스템 :

플롯(영어: plot)은 소설, 영화 등에서 이야기를 구성하는 일련의 사건(series of events) 또는 사건의 논리적인 패턴과 배치를 의미한다. 플롯은 소설에서 인물, 주제, 배경, 문체 등과 함께 허구의 이야기를 구성하는 기본 요소이다.

1. 데이터베이스에 영화 줄거리 요약을 저장합니다.
2. OpenAI의 'text-embedding-ada-002'와 같은 모델을 사용하여 **플롯을 삽입(임베딩)**합니다.
3. 사용자가 영화 선호도를 입력할 수 있는 **검색 시스템을 구현**하고 시스템은 줄거리 요약을 기반으로 **유사한 영화를 검색**합니다.

---

## - 20개 영화 줄거리 데이터(예시) : movie\_plots\_korean.txt -

- 1: 한 명의 숙련된 도둑이 누군가의 마음에 생각을 심는 데 성공하면 구원을 받을 기회를 얻게 됩니다. 영화는 꿈과 현실의 경계선을 탐구합니다. (**인셉션**)
- 2: 살인 혐의로 억울하게 투옥된 한 남자가 수년간 신중하게 탈출 계획을 세우며 교도소에서 깊은 우정을 쌓아갑니다. (**쇼생크 탈출**)
- 3: 해커가 자신이 살고 있는 세계가 사실 기계들에 의해 통제되는 시뮬레이션이라는 사실을 발견하고 반란군에 합류합니다. (**매트릭스**)
- 4: 고등학생들이 감옥 같은 학교에서 살아남기 위해 싸우는 이야기를 그린 영화입니다. (**배틀로얄**)
- 5: 한 명의 사이코패스 연쇄 살인범이 고통스럽고 교묘하게 사람들을 죽이는 과정을 그린 영화입니다. (**세븐**)
- 6: 세계 각지에서 사람들이 알 수 없는 이유로 사라지며 발생하는 미스터리를 풀어가는 이야기입니다. (**로스트**)
- 7: 인류 멸망 이후, 살아남은 사람들이 자원을 찾기 위해 황무지에서 생존하는 이야기를 그린 영화입니다. (**매드 맥스: 분노의 도로**)
- 8: 미래의 디스토피아에서 감정을 억제하기 위해 의약품을 복용하는 사람들이 감정을 되찾기 위한 투쟁을 그리는 영화입니다. (**이퀼리브리엄**)
- 9: 세계적인 음악가가 되기 위해 노력하는 한 젊은 피아니스트의 이야기를 다룬 영화입니다. (**위플래쉬**)
- 10: 한 명의 무술 고수가 복수를 다짐하며 적을 하나씩 쓰러뜨리는 과정을 그린 영화입니다. (**킬 빌**)
- 11: 어린 아이가 우연히 초능력을 얻게 되어 벌어지는 기이한 사건을 그린 영화입니다. (**크로니클**)
- 12: 전 세계적인 전염병으로 인류가 멸종 위기에 처한 상황에서 살아남기 위한 생존자들의 투쟁을 그린 영화입니다. (**월드워Z**)
- 13: 외계인이 침공하여 지구가 파괴될 위기에 처한 가운데, 인류가 생존을 위해 싸우는 이야기를 다룬 영화입니다. (**인디펜던스 데이**)
- 14: 시간 여행을 통해 과거와 미래를 오가며 인류를 구하기 위한 과학자의 이야기를 그린 영화입니다. (**인터스텔라**)
- 15: 한 남자가 죽은 아내를 잊지 못해 그녀와의 추억을 계속해서 되새기며 살아가는 이야기를 그린 영화입니다. (**이터널 선샤인**)
- 16: 전쟁에서 살아남은 병사가 PTSD로 고통받으며 새로운 삶을 살아가는 과정을 그린 영화입니다. (**아메리칸 스나이퍼**)
- 17: 젊은 해커들이 각종 사이버 범죄를 저지르며 법의 경계를 넘나드는 이야기를 그린 영화입니다. (**스노든**)
- 18: 신비로운 무기가 발견되며 이를 차지하기 위한 전쟁이 벌어지는 이야기를 다룬 영화입니다. (**반지의 제왕**)
- 19: 거대한 우주선이 지구로 오며 벌어지는 미지의 존재와의 첫 접촉을 다룬 영화입니다. (**컨택트**)
- 20: 마법의 세계에서 전설적인 마법사가 되어 악과 싸우는 이야기를 그린 영화입니다. (**해리 포터**)

# 영화 추천 시스템

## 플롯 유사성을 기반으로 한 영화 추천 시스템

Enter a movie plot or theme to find similar movies:

Search

## 플롯 유사성을 기반으로 한 영화 추천 시스템

Enter a movie plot or theme to find similar movies:

Search

Recommended Movies for: "어떤 남자가 억울하게 감옥에 갇혔고, 그는 탈출을 계획합니다"

### 쇼생크 탈출

2: 살인 혐의로 억울하게 투옥된 한 남자가 수년간 신중하게 탈출 계획을 세우며 교도소에서 깊은 우정을 쌓아갑니다. (쇼생크 탈출)



# 필요한 테이블

SQL 스크립트는 PostgreSQL 데이터베이스에서 고차원 벡터 데이터를 저장하고 유사도 검색을 효율적으로 수행하기 위한 환경을 설정하고 테이블을 생성하는 코드

```
CREATE EXTENSION IF NOT EXISTS vector;
CREATE EXTENSION IF NOT EXISTS hstore;
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
```

수동으로 만들 때

```
CREATE TABLE IF NOT EXISTS movie_vector(
    id uuid DEFAULT uuid_generate_v4() PRIMARY KEY,
    content text,
    metadata json,
    embedding vector(1536)
);
CREATE INDEX ON movie_vector USING HNSW (embedding vector_cosine_ops);
```

**embedding 필드**

- 타입: vector(1536)
- 역할: 고차원 벡터 데이터를 저장.
- 크기: 벡터의 차원은 1536으로 제한.
- 예시: AI 모델(예: OpenAI text-embedding-ada-002)이 생성한 임베딩 데이터.

```
spring:
  datasource:
    url: jdbc:postgresql://localhost:5432/mydatabase
    username: myuser
    password: secret
  ai:
    vectorstore:
      pgvector:
        index-type: HNSW
        distance-type: COSINE_DISTANCE
        dimensions: 1536
        table-name : movie_vector
        initialize-schema: true
```

자동으로 만들 때

(1) **vector 확장**

- 설명: PostgreSQL에서 벡터 데이터를 저장하고 연산할 수 있도록 지원하는 확장입니다.
- 용도:
  - vector(1536)와 같은 필드를 정의할 수 있음.
  - 벡터 간 거리 계산(코사인, 유clidean 등)을 효율적으로 수행

(2) **hstore 확장**

- 설명: 간단한 키-값 구조 데이터를 저장할 수 있도록 지원하는 PostgreSQL 확장입니다.
- 용도:
  - 데이터 구조가 JSON보다 간단한 경우 적합.
  - 현재 스크립트에서는 직접 사용되지 않으나, 메타데이터 처리를 위해 설치된 것으로 보임.

(3) **uuid-ossp 확장**

- 설명: UUID를 생성하기 위한 PostgreSQL 확장입니다.
- 용도:
  - uuid\_generate\_v4() 함수를 제공.
  - 기본 키를 UUID로 자동 생성하도록 지원.

HNSW 인덱스:

- HNSW(다중 단계 근사 검색) 알고리즘을 사용해 벡터 검색을 효율화.
- PostgreSQL 확장 vector에서 지원.

vector\_cosine\_ops:

- 코사인 유사도를 기반으로 벡터 간의 거리를 계산하는 연산자.
- 유사도 검색에서 사용.

```
drop table movie_vector;
commit;
```

# 데이터 로딩

t	id	content	metadata	embedding
3749dc1f-8de6-4079-83ad-0d4ef12b5062	1: 한 명의 숙련된 드둑이 누군가의 마음에 생각을 심는 데 성 {}			[ -0.014536328, -0.026084634, 0.02398494, -0.0
e65256a9-d00b-492c-b1d6-421cd8d6153d4	2: 살인 혐의로 악을 향해 투옥된 한 남자가 수년간 신중하게 틀 {}			[ 0.0006885263, -0.03508169, 0.018615616, -0.0
56fbeeae1-32bd-404b-a80c-62183c1fcddf	3: 해커가 자신이 살고 있는 세계가 사실 기계들에 의해 통제 {}			[ -0.0007236007, -0.021433596, 0.0024673324,
0cfa4ec2-9187-4434-98ad-3abdb1bb45e02	4: 고등학생들이 감옥 같은 학교에서 살아남기 위해 싸우는 이 {}			[ 0.012539152, -0.033802893, -0.009376652, -0.
b26a1db3-0721-485a-a4c3-bb044165b647	5: 한 명의 사이코패스 연예 살인범이 고를 풍자하고 교묘하게 사 {}			[ -0.017864913, -0.0066502444, 0.02607275, -0.
02266c6d-ed85-4e75-bb66-abebbe94e20f9	6: 세계 각지에서 사람들이 알 수 없는 이유로 사라지며 발생하는 {}			[ 0.010809804, -0.022918083, 0.015724532, -0.
d84e21ce-f891-48bb-a5f1-91a8a949f979	7: 인류 몰락 이후, 살아남은 사람들이 자원을 찾기 위해 활동하는 {}			[ 0.0074350713, -0.044323355, -0.012996478, -1.
505060cc-092a-4dfe-9c56-73a766a6aa0	8: 미래의 디스토피아에서 강정을 억제하기 위해 의약품을 복용하는 {}			[ -0.035505407, -0.015794978, 0.031589955, -0.
0b008597-f297-4a2c-b2f8-446ad390345a	9: 세계적인 음악가가 되기 위해 노력하는 한 젊은 피아니스트 {}			[ 0.0010506647, -0.036570504, 0.034159828, -0.
36934e0e-00a2-4bc6-8041-2b7f67b492f1	10: 한 명의 무술 고수가 복수를 다짐하며 적을 하나씩 쓰러뜨리는 {}			[ -0.016647898, -0.025032118, 0.023331165, -0.
fc88b9a2-6082-4993-8999-a0f384b531d4	11: 어린 아이가 우연히 초능력을 얻게 되어 벌어지는 기이한 사건 {}			[ -0.0046362784, -0.035500646, 0.011769524, -1.
df521958-3d56-4e58-b737-c2d1ce7a0fd7	12: 전 세계적인 전염병으로 인류가 멸종 위기에 처한 상황에서의 생존 {}			[ -0.005801096, -0.031000426, 0.013439643, -0.
41617e3e-f861-416f-bd35-30b509da8242	13: 외계인이 제공하여 지구가 파괴될 위기에 처한 가운데, 인류가 {}			[ 0.0066437293, -0.037344266, 0.009395239, -0.
a0a398f7-82f8-4088-9440-fe448a0d7440	14: 시간 여행을 통해 과거와 미래를 오가며 인류를 구하기 위한 {}			[ 0.016079223, -0.02465394, 0.009351269, -0.
c450b446-ae67-4ebb-b087-d9465c1b4edb	15: 한 남자가 죽은 아내를 잊지 못해 그녀와의 추억을 계속해 {}			[ -0.008496891, -0.03144235, 0.015296974, -0.
e12de8a0-bf3-4c27-b415-57e7d2d023e	16: 전쟁에서 살아남은 병사가 PTSD로 고통받으며 새로운 삶을 살피는 {}			[ -0.012307323, -0.024784582, 0.02554276, -0.
9c765ade-7635-4ecc-afc9-82b06f1dd09	17: 젊은 해커들이 각종 사이버 범죄를 저지르며 범의 경계를 {}			[ 0.022878762, -0.03500678, 0.023480482, -0.
ace2d46b-9a16-4644-c914-aecb6970d358	18: 신비로운 무기가 발견되어 이를 차지하기 위한 전쟁이 벌어지는 {}			[ -0.014887009, -0.04728354, -0.0018657688, -0.
384c0ae2-5532-4d4c-806e-9fa68e78a204	19: 거대한 우주선이 지구로 오며 벌어지는 미지의 존재와의 만남 {}			[ 0.009907656, -0.03375647, 0.009316254, -0.
bd7248eb-cd02-4172-9bbd-f43d527853e7	20: 마법의 세계에서 전설적인 마법사가 되어 악과 싸우는 이 {}			[ 0.00032628395, -0.034624483, -0.002768615, -0.

## @Configuration

```
public class MovieLoader {

    private final VectorStore vectorStore;
    private final JdbcClient jdbcClient;

    @Value("classpath:movie_plots_korean.txt")
    Resource resource;

    public MovieLoader(VectorStore vectorStore, JdbcClient jdbcClient) {
        this.vectorStore = vectorStore;
        this.jdbcClient = jdbcClient;
    }

    @PostConstruct
    public void init() throws Exception {
        Integer count=jdbcClient.sql("select count(*) from movie_vector")
            .query(Integer.class)
            .single();
        System.out.println("No of Records in the PG Vector Store=" + count);
        if(count==0){
            List<Document> documents = Files.lines(resource.getFile().toPath())
                .map(Document::new)
                .collect(Collectors.toList());
            TextSplitter textSplitter = new TokenTextSplitter();
            for(Document document : documents) {
                List<Document> splitteddocs = textSplitter.split(document);
                System.out.println("before adding document: " +
                    document.getContent());
                vectorStore.add(splitteddocs);
                System.out.println("Added document: " + document.getContent());
                Thread.sleep(1000);
            }
            System.out.println("Application is ready to Serve the Requests");
        }
    }
}
```

# 데이터 로딩

```
@Controller
public class MovieController {

    private final VectorStore vectorStore;
    private final ChatClient chatClient;

    public MovieController(VectorStore vectorStore, ChatClient.Builder chatClient) {
        this.vectorStore = vectorStore;
        this.chatClient = chatClient.build();
    }

    @GetMapping("/movie")
    public String getRecommendationForm() {
        return "movieRAG"; // returns the HTML file 'recommend.html'
    }

    @PostMapping("/recommend")
    public String recommendMovies1(@RequestParam("query") String query, Model model) throws Exception {
        // Fetch similar movies using vector store
        List<Document> results = vectorStore.similaritySearch(SearchRequest.query(query).withSimilarityThreshold(0.85).withTopK(1));

        if (!results.isEmpty()) {
            Document topResult = results.get(0);
            String movieContent = topResult.getContent();
            String title=movieContent.substring(movieContent.indexOf("(")+1, movieContent.lastIndexOf("")));
            // Use Jsoup to fetch the YouTube URL
            List<String> url = URLTest.searchYouTube(title);
            model.addAttribute("title", title);
            // Add the movie details and YouTube URL to the model
            model.addAttribute("results", movieContent);
            model.addAttribute("youtubeUrls", url);
        }else{
            model.addAttribute("message", "No closely related movies found.");
        }
        model.addAttribute("query", query);

        return "movieRAG"; // Renders the 'movieRAG.html' view
    }
}
```

## 유튜브 영상 검색

```
public class URLTest {  
    // Helper method to fetch YouTube link using ChatClient  
    public static List<String> searchYouTube(String movieTitle) throws Exception {  
  
        String url = "https://www.googleapis.com/youtube/v3/search?part=snippet&type=video&q="  
            + movieTitle  
            + "&order=relevance"  
            + "&key=[REDACTED]";  
  
        RestTemplate restTemplate = new RestTemplate();  
        ResponseEntity<String> response = restTemplate.getForEntity(url, String.class);  
  
        List<String> videoUrls = new ArrayList<>();  
        JSONObject jsonResponse = new JSONObject(response.getBody());  
        JSONArray items = jsonResponse.getJSONArray("items");  
  
        for (int i = 0; i < items.length(); i++) {  
            JSONObject item = items.getJSONObject(i);  
            String videoid = item.getJSONObject("id").getString("videoid");  
            videoUrls.add(videoid);  
        }  
        return videoUrls;  
    }  
}
```

# View

```
<div class="container mt-5">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <h1 class="text-center mb-4">플롯 유사성을 기반으로 한 영화 추천 시스템</h1>
            <p class="text-center">Enter a movie plot or theme to find similar movies:</p>

            <!-- Form to input query -->
            <form action="#" th:action="@{/recommend}" method="post" class="mb-4">
                <div class="input-group mb-3">
                    <input type="text" name="query" class="form-control" placeholder="Enter a movie plot or theme..." th:value="${query}" required>
                    <button class="btn btn-primary" type="submit">Search</button>
                </div>
            </form>

            <!-- Show results after user submits a query -->
            <div th:if="${results} != null">
                <h4 class="text-center">Recommended Movies for: <span th:text="${query}"></span></h4>
                <ul class="list-group mt-4">
                    <!-- Display the movie title and plot -->
                    <li class="list-group-item mb-3">
                        <h5 th:text="${title}"></h5>
                        <p th:text="${results}"></p>

                        <!-- Loop over YouTube URLs and embed each video -->
                        <div th:each="youtubeUrl : ${youtubeUrls}">
                            <div class="embed-responsive embed-responsive-16by9 mb-3">
                                <iframe th:src="https://www.youtube.com/embed/" + ${youtubeUrl}>
                                    frameborder="0" allowfullscreen class="embed-responsive-item"></iframe>
                            </div>
                        </div>
                    </li>
                </ul>
            </div>

            <div th:if="${message} != null">
                <p class="alert alert-warning text-center" th:text="${message}"></p>
            </div>
        </div>
    </div>
</div>
```

---

## Github

<https://github.com/parkmaeil/movierag.git>

---

## **12\_ RAG기반 호텔 AI 챗봇 서비스**

**목표:** GPT 모델을 활용하여 사용자의 질문을 이해하고 관련 정보를 스트리밍 방식으로 실시간 제공하는 챗봇을 구현한다.

# Application UI

## RAG기반 호텔 AI 챗봇 서비스

호텔 챗봇은 고객의 질문에 답변하거나 필요한 정보를 제공하는 AI 기반 서비스입니다.

**User:**

취소정책을 알려주세요

**Bot:**

호텔의 취소 정책은 다음과 같습니다:

- 전날 취소 시 숙박 요금의 30%가 부과됩니다.
- 당일 취소 시 숙박 요금의 50%가 부과됩니다.
- 연락 없이 취소할 경우 숙박 요금의 100%가 부과됩니다.

이 정책은 모든 예약에 적용되며, 예약 시 고객에게 명확히 고지됩니다. 만약 특별한 사정이 있는 경우, 관리자의 승인을 받아 유연하게 대응할 수 있습니다. 추가적인 문의사항이 있으시면 언제든지 말씀해 주세요.

Type your message...

Send

## Docker 컨테이너 실행

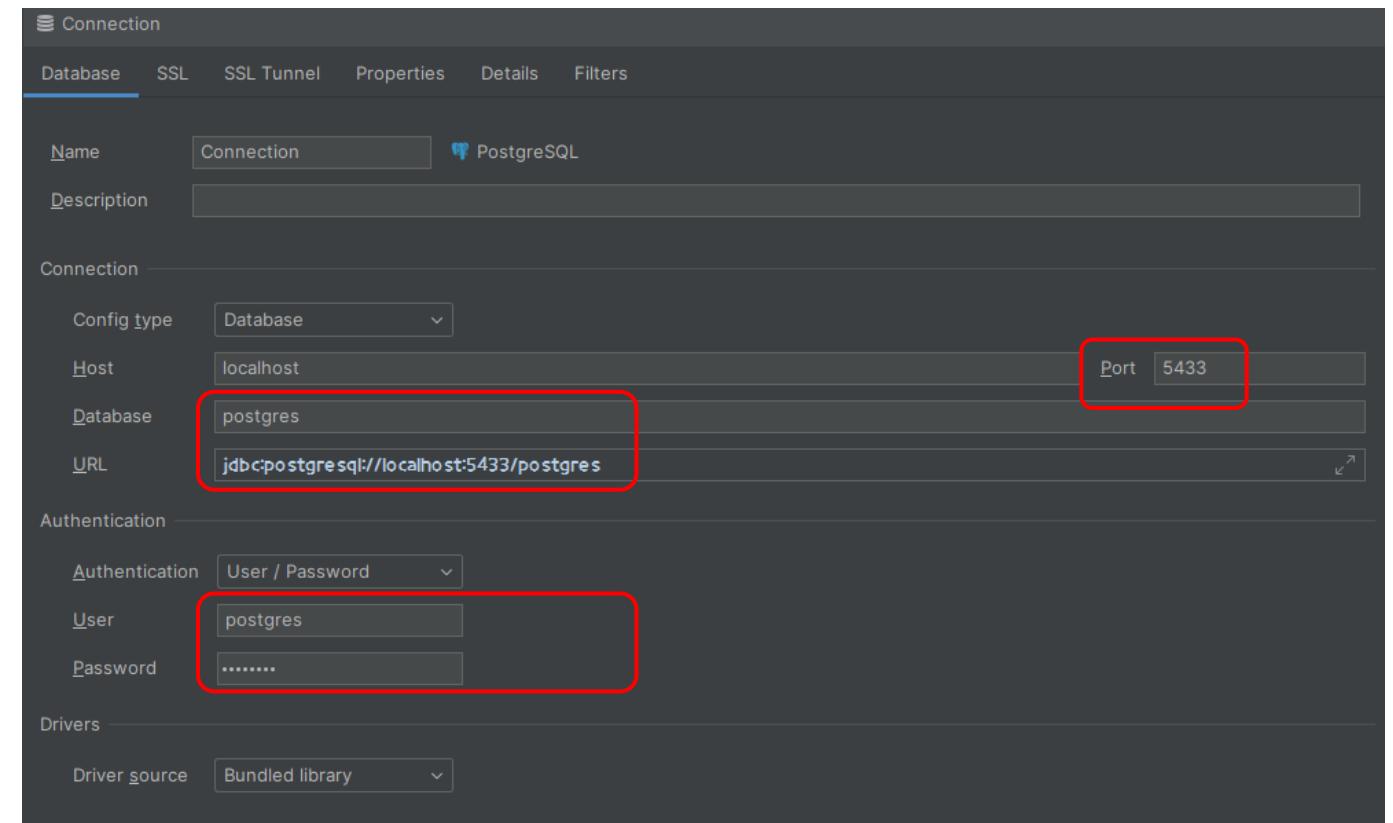
```
docker run -it --rm --name postgres2 -p 5433:5432 -e POSTGRES_USER=postgres -e  
POSTGRES_PASSWORD=postgres pgvector/pgvector:pg16
```

The screenshot shows the Docker Container Overview page. At the top, there are two summary metrics: 'Container CPU usage' (0.01% / 1200%) and 'Container memory usage' (123.25MB / 45.91GB). Below these are search and filter controls. A red box highlights the 'Name' column header in the table, and a red arrow points from the 'Name' in the command above to this header. The table lists one container:

Name	Container ID	Image	Port(s)	CPU (%)	Actions
postgres2	ac39e995ddcd	pgvector/p... 5433:5432 ↗	0%		

# application.yml

```
spring:  
  datasource:  
    url: jdbc:postgresql://localhost:5433/postgres  
    username: postgres  
    password: postgres  
  ai:  
    vectorstore:  
      pgvector:  
        index-type: HNSW  
        distance-type: COSINE_DISTANCE  
        dimensions: 1536  
        table-name : hotel_vector  
        initialize-schema: true
```



# Data Embedding

data.txt

- 손님 맞이:손님이 호텔에 도착하면 친절한 미소와 함께 예의 바르고 활기찬 인사말을 건네야 합니다. 예를 들어, 'LuxeStay Hotel에 오신 것을 환영합니다!'라고 인사하며 고객의 이름을 알고 있을 경우 '어서 오세요, [고객 이름]님!'이라는 개인화된 인사말을 사용합니다. 고객이 처음 방문인지 재방문인지 확인하고, 재방문 고객에게는 특별한 감사 인사를 전달하여 만족도를 높입니다.
- 체크인과 체크아웃:체크인 시간은 오후 3시, 체크아웃 시간은 오전 11시입니다. 고객이 이른 체크인이나 늦은 체크아웃을 요청할 경우, 객실 공실 상태를 확인한 후 가능 여부를 안내합니다. 불가능한 경우, 짐을 무료로 보관하는 서비스를 제공합니다. 고객에게 체크아웃 연장 시 발생할 수 있는 추가 요금에 대해 사전 안내하는 것이 중요합니다.
- Wi-Fi 및 주차장 안내:LuxeStay Hotel의 모든 객실에는 무료 Wi-Fi가 제공됩니다. 고객에게 Wi-Fi 이름과 비밀 번호를 명확히 안내하고, 연결에 문제가 발생할 경우 간단한 문제 해결 방법을 제공해야 합니다. 또한, 호텔은 200대 차량을 수용할 수 있는 무료 주차장을 제공합니다. 주차장의 위치와 개방 시간(24시간 운영)을 정확히 안내합니다.
- 배리어 프리 대응:유니버설 룸은 훨체어를 사용하는 고객을 위해 설계되었습니다. 객실 내 샤워 체어, 높이 조절 침대 등의 시설을 확인하여 필요에 따라 설명합니다. 또한, 엘리베이터 위치, 장애인 전용 화장실, 경사로 등 호텔 내 편의시설을 숙지하고 고객에게 필요한 경우 도움을 제공합니다. 훨체어 대여 서비스도 준비되어 있습니다.

The screenshot shows a DB Browser interface connected to a PostgreSQL database named 'Connection'. The left sidebar shows the database structure with 'Consoles', 'Schemas' (including 'information\_schema', 'pg\_catalog', 'pg\_toast', and 'public'), and a single table 'hotel\_vector' under the 'public' schema. The main window displays the 'hotel\_vector' table with 12 rows of data. The table structure is as follows:

	id	content	metadata	embedding
1	81392417-b3f2-43ee-a3bc-e6817859ee28	1. 손님 맞이:손님	{}	[-0.0041910037, 0.0]
2	3f9bf4c4-b97f-4824-a20c-6d7a364c7c6c	2. 체크인과 체크	{}	[0.014657134, 0.000]
3	0c15350b-1e83-44eb-b6e9-f4e0c9b8b0c4	3. Wi-Fi 및 주차	{}	[0.01015419, 0.0068]
4	13c9f50a-3a81-47b1-9198-d9028865ba0b	4. 배리어 프리 다	{}	[0.01153626, -0.007]
5	ca1a76de-bcf4-4f72-8260-21c2ad3df312	5. 반려동물 대응	{}	[-0.01596592, 0.011]
6	04577192-f0a5-4278-b697-36d9d3a395c9	6. 룸 서비스:룸 서	{}	[0.014770305, -0.00]
7	ea1338d8-ac25-4556-a87e-aefc56b25ac5	7. 금연 정책 및 흡	{}	[-0.0024168643, 0.0]
8	d8a5ebe9-86fa-492d-bb7e-cc4ef1aaf500	8. 취소 정책:취소	{}	[0.0090279365, 0.0]
9	d153a640-b97e-43d8-a94a-201dcc26558b	9. 결제 방법:체크	{}	[-0.0032679674, 0.0]
10	abd971fa-ce62-4c52-9989-228e082b5827	10. 고객 불만 처	{}	[-0.021703867, 0.00]
11	74918188-98e5-4e5f-a131-484dd825f123	11. 긴급 상황 대응	{}	[0.0015004743, 0.00]
12	6c440b98-c93b-407a-9ff2-5c0eddf87fb4	12. 관광 정보 및	{}	[-0.0030166898, 0.0]

# Data Loading

## data.txt

- 손님 맞이:손님이 호텔에 도착하면 친절한 미소와 함께 예의 바르고 활기찬 인사말을 건네야 합니다. 예를 들어, 'LuxeStay Hotel에 오신 것을 환영합니다!'라고 인사하며 고객의 이름을 알고 있을 경우 '어서 오세요, [고객 이름]님!'이라는 개인화된 인사말을 사용합니다. 고객이 처음 방문인지 재방문인지 확인하고, 재방문 고객에게는 특별한 감사 인사를 전달하여 만족도를 높입니다.
- 체크인과 체크아웃:체크인 시간은 오후 3시, 체크아웃 시간은 오전 11시입니다. 고객이 이른 체크인이나 늦은 체크아웃을 요청할 경우, 객실 공실 상태를 확인한 후 가능 여부를 안내합니다. 불가능한 경우, 짐을 무료로 보관하는 서비스를 제공합니다. 고객에게 체크아웃 연장 시 발생할 수 있는 추가 요금에 대해 사전 안내하는 것이 중요합니다.
- Wi-Fi 및 주차장 안내:LuxeStay Hotel의 모든 객실에는 무료 Wi-Fi가 제공됩니다. 고객에게 Wi-Fi 이름과 비밀 번호를 명확히 안내하고, 연결에 문제가 발생할 경우 간단한 문제 해결 방법을 제공해야 합니다. 또한, 호텔은 200대 차량을 수용할 수 있는 무료 주차장을 제공합니다. 주차장의 위치와 개방 시간(24시간 운영)을 정확히 안내합니다.
- 배리어 프리 대응:유니버설 룸은 휠체어를 사용하는 고객을 위해 설계되었습니다. 객실 내 샤워 체어, 높이 조절 침대 등의 시설을 확인하여 필요에 따라 설명합니다. 또한, 엘리베이터 위치, 장애인 전용 화장실, 경사로 등 호텔 내 편의시설을 숙지하고 고객에게 필요한 경우 도움을 제공합니다. 휠체어 대여 서비스도 준비되어 있습니다.

```

@Configuration
public class HotelLoader {

    private final VectorStore vectorStore;
    private final JdbcClient jdbcClient;

    @Value("classpath:data.txt")
    Resource resource;

    public HotelLoader(VectorStore vectorStore, JdbcClient jdbcClient) {
        this.vectorStore = vectorStore;
        this.jdbcClient = jdbcClient;
    }

    @PostConstruct
    public void init() throws Exception {
        Integer count=jdbcClient.sql("select count(*) from hotel_vector")
            .query(Integer.class)
            .single();
        System.out.println("No of Records in the PG Vector Store="+count);
        if(count==0){
            List<Document> documents = Files.lines(resource.getFile().toPath())
                .map(Document::new)
                .collect(Collectors.toList());
            TextSplitter textSplitter = new TokenTextSplitter();
            for(Document document : documents) {
                List<Document> splitteddocs = textSplitter.split(document);
                System.out.println("before adding document: " + document.getContent());
                vectorStore.add(splitteddocs); //임베딩
                System.out.println("Added document: " + document.getContent());
                Thread.sleep(1000); // 1초
            }
            System.out.println("Application is ready to Serve the Requests");
        }
    }
}

```

# HotelController

```
@RestController
public class HotelController {

    private final VectorStore vectorStore;
    private final ChatClient chatClient;

    public HotelController(VectorStore vectorStore, ChatClient.Builder chatClient) {
        this.vectorStore = vectorStore;
        this.chatClient = chatClient.build();
    }

    @GetMapping("/question")
    public Flux<String> question(@RequestParam("question") String question, Model model) throws Exception {
        System.out.println(question);
        // Fetch similar question using vector store
        List<Document> results = vectorStore.similaritySearch(SearchRequest.query(question).withSimilarityThreshold(0.5).withTopK(1));
        System.out.println(results.size());

        String template = """
            당신은 어느 호텔 직원입니다. 문맥에 따라 고객의 질문에 정중하게 대답해 주십시오.
            컨텍스트가 질문에 대답할 수 없는 경우 '모르겠습니다'라고 대답하세요.

            컨텍스트:
            {context}
            질문:
            {question}

            답변:
            """;
    }

    return chatClient.prompt()
        .user(promptUserSpec -> promptUserSpec.text(template)
            .param("context", results)
            .param("question", question))
        .stream()
        .content();
    }
}
```

# View

```

<script>
  const chatBox = document.getElementById('chatBox');
  const chatForm = document.getElementById('chatForm');
  const messageInput = document.getElementById('messageInput');

  chatForm.addEventListener('submit', async function(e) {
    e.preventDefault();
    const userMessage = messageInput.value.trim();
    if (!userMessage) return;

    // Add user message to chat
    appendMessage('User', userMessage);

    // Clear input field
    messageInput.value = '';

    try {
      // Fetch streaming response
      const response = await fetch('/question?question=${encodeURIComponent(userMessage)}');
      const reader = response.body.getReader();
      let botMessageElement = appendMessage('Bot', ''); // Add empty bot message
      let contentElement = botMessageElement.querySelector('.content');
      await processStream(reader, contentElement);
    } catch (error) {
      console.error('Error:', error);
      appendMessage('System', 'An error occurred. Please try again.');
    }
  });
}

function appendMessage(sender, content) {
  const messageElement = document.createElement('div');
  messageElement.className = 'message';
  messageElement.innerHTML = `
    <div class="sender ${sender === 'User' ? 'text-primary' : 'text-success'}">${sender}</div>
    <div class="content">${content}</div>
  `;
  chatBox.appendChild(messageElement);
  chatBox.scrollTop = chatBox.scrollHeight;
  return messageElement;
}

async function processStream(reader, contentElement) {
  const decoder = new TextDecoder('utf-8');
  try {
    while (true) {
      const { done, value } = await reader.read();
      if (done) break;
      contentElement.innerHTML += decoder.decode(value, { stream: true });
      chatBox.scrollTop = chatBox.scrollHeight;
    }
  } catch (error) {
    console.error('Error processing stream:', error);
    contentElement.innerHTML += '<br><span class="text-danger">[Stream interrupted]</span>';
  }
}
</script>

```

---

## Github

<https://github.com/parkmaeil/hotelrag.git>

## 04\_DALL-E 3를 이용한 이미지 생성하기 이후 학습

---

### 13\_Web Speech와 OpenAI로 음성 기반 이미지 생성

**목표:** Web Speech API와 OpenAI를 이용한 사용자의 실시간 라이브 음성을 텍스트로 변환하여 LLM을 통한 이미지 생성을 학습 합니다.

# Application UI

The screenshot shows a web browser window with the URL `localhost:8080/imageview`. A red dashed circle highlights the refresh button in the top-left corner of the browser bar. Below the browser bar, the main content area has a title "Web Speech와 OpenAI로 음성 기반 이미지 생성". A text input field contains the placeholder "음성 메시지가 표시됩니다.(최소 10자 이상이 되어야 합니다.)" and a red dashed circle highlights the microphone icon on its right. A large blue outlined box covers the bottom-left portion of the form, containing the text "음성 인식 중...". To the right of this box are two input fields: "모델 선택" with "DALL-E 2" and a dropdown arrow, and "이미지 개수 (1-3)" with the number "1". On the far right, there are two bullet points with microphone icons: "성경 창세기 1장 1절을 이미지로 만들어 줘" and "성경 노아의 방주를 이미지로 만들어 줘".

Web Speech와 OpenAI로 음성 기반 이미지 생성

음성 메시지가 표시됩니다.(최소 10자 이상이 되어야 합니다.)

음성 인식 중...

모델 선택

DALL-E 2

이미지 개수 (1-3)

1

성경 창세기 1장 1절을 이미지로 만들어 줘

성경 노아의 방주를 이미지로 만들어 줘

# View

## Web Speech와 OpenAI로 음성 기반 이미지 생성

성경 창세기 1장 1절을 이미지로 만들어 줘



음성 인식이 종료되었습니다.

모델 선택

DALL-E 3

이미지 개수 (1-3)

1



## Web Speech와 OpenAI로 음성 기반 이미지 생성

성경 노아의 방주를 이미지로 만들어 줘



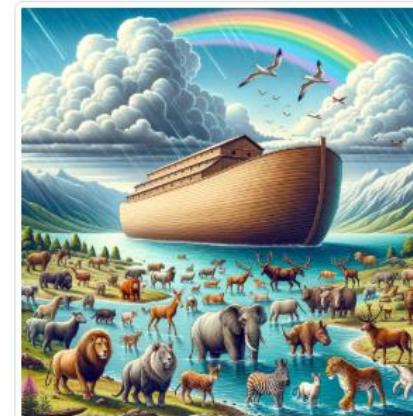
음성 인식이 종료되었습니다.

모델 선택

DALL-E 3

이미지 개수 (1-3)

1



# View

Web Speech API의 음성 인식 객체를 생성하는 부분

```
recognition = new (window.SpeechRecognition || window.webkitSpeechRecognition)();
```

- **window.SpeechRecognition**: 표준 음성 인식 API입니다.
- 최신 브라우저(예: Chrome, Edge)는 SpeechRecognition 객체를 기본으로 제공합니다.
- **window.webkitSpeechRecognition**: 이전 버전의 Web Speech API 구현으로, 일부 브라우저(특히 구버전 Chrome)에서 사용됩니다.
- SpeechRecognition 객체가 없는 경우, webkitSpeechRecognition 객체를 사용합니다.
- **new** 키워드를 통해 음성 인식 객체를 생성

음성 인식 결과를 처리하여 텍스트 데이터를 추출하는 부분

```
const transcript = event.results[0][0].transcript; // 첫 번째 결과
```

**event.results:**

- 음성 인식 이벤트에서 반환되는 결과입니다.
- **event.results**는 음성 인식 결과 목록으로, 사용자가 말한 내용을 여러 세그먼트로 나눠 제공합니다.
- 이 배열의 각 항목은 인식된 세그먼트(문장 또는 단어)를 나타냅니다.

**event.results[0]:**

- 첫 번째 결과 세그먼트를 가져옵니다.
- 이 세그먼트는 사용자가 말한 문장의 일부 또는 전체를 나타낼 수 있습니다.

**event.results[0][0]:**

- 첫 번째 결과 세그먼트의 첫 번째 대안을 가져옵니다.
- 음성 인식 API는 종종 여러 대안을 제공합니다(예: "안녕하세요" vs. "아녕하세요"). 여기서는 가장 신뢰도가 높은 첫 번째 대안을 선택합니다.

**event.results[0][0].transcript:**

- 대안에서 인식된 텍스트를 추출합니다.
- transcript 속성에는 사용자가 말한 내용을 텍스트 형태로 반환합니다.

**작동과정**

- 사용자가 말을 시작하면, **recognition** 객체가 음성을 분석하여 **onresult** 이벤트를 트리거합니다.
- 이벤트 핸들러에서 **event.results**를 통해 인식된 결과를 가져옵니다.
- 첫 번째 세그먼트와 첫 번째 대안을 선택하여 **transcript** 변수에 텍스트를 저장합니다.
- 이 텍스트는 HTML 요소(예: <input> 텍스트 박스)에 표시되거나 서버로 전송될 수 있습니다.

```
// 음성 인식 시작
function startSpeechRecognition() {
    recognition = new (window.SpeechRecognition || window.webkitSpeechRecognition)();
    recognition.lang = 'ko-KR'; // 한국어 설정
    recognition.continuous = false; // 한 번만 인식
    recognition.interimResults = false; // 중간 결과 비활성화

    recognition.onstart = () => {
        console.log("음성 인식 시작");
        document.getElementById("status").innerText = "음성 인식 중...";
    };

    recognition.onresult = (event) => {
        const transcript = event.results[0][0].transcript; // 첫 번째 결과
        console.log("음성 인식 결과:", transcript);

        // 텍스트 박스에 음성 인식 결과 입력
        document.getElementById("message").value = transcript;
        document.getElementById("status").innerText = "음성 인식 완료";

        // 텍스트 길이가 10자 이상일 경우 generateImage() 호출
        if (transcript.length >= 10) {
            generateImage();
        } else {
            document.getElementById("status").innerText += " - 텍스트 길이가 너무 짧습니다.";
        }
    };

    recognition.onerror = (event) => {
        console.error("오류 발생:", event.error);
        document.getElementById("status").innerText = "오류 발생: " + event.error;
    };

    recognition.onend = () => {
        console.log("음성 인식 종료");
        document.getElementById("status").innerText = "음성 인식이 종료되었습니다.";
    };
}

recognition.start(); // 음성 인식 시작
}
```

---

## Github

<https://github.com/parkmaeil/voiceimage.git>

- 콘텐츠 제공자 : 이츠코딩 대표 : 박매일(bitcocom@empas.com)

**비대면(온라인) 수업에 따른 강의 영상 및 자료 무단 복제, 배포 금지 안내**

- 비대면(온라인) 수업에서 제공하는 강의 영상(콘텐츠), 강의 자료는 학습 목적으로 수강생만 이용할 수 있습니다.
- 저작자의 허락 없이 무단 복제, 전송, 배포, 도용 등은 저작권법 위반 행위이 오니 수강 학생은 학습자료 관리에 유의하시기 바랍니다