생성형 인공지능(OpenAI) API 활용 –
Youtube 영상 요약하기

OpenAI OpenAPI 활용

ChatGPT, Whisper

https://platform.openai.com/

Youtube 영상 요약하기

생성형 인공지능 API 활용

Youtube 영상 요약하기

- 라이브러리 및 ffmpeg 설치
- 유튜브 영상에서 오디오 추출하기
- 파일 자르기
- 텍스트 추출 및 요약하기

이 예제는 OpenAI의 API를 사용합니다. 그러므로 OpenAI 플랫폼에서 구매한 Credit이 있어야 하며, API 키가 생성되어 있어야 합니다.

- 결제: https://platform.openai.com/settings/organization/billing/overview
- API 키 발급: https://platform.openai.com/settings/organization/api-keys

준비하기

YouTube의 영상 내용 요약하기

📌 전체 작업 흐름

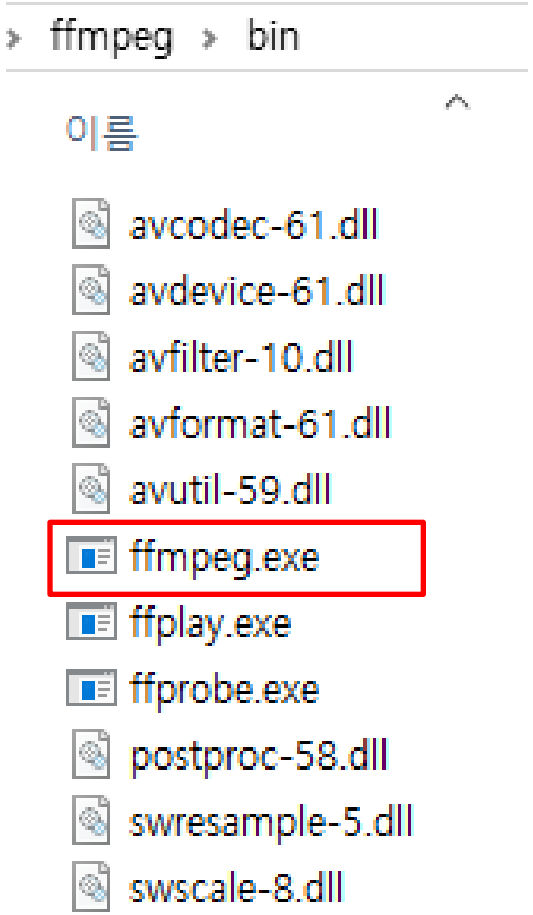
1.YouTube 오디오 추출 (MP3)
2.Whisper API로 음성 → 텍스트 변환
3.GPT(Chat Completions)으로 요약

필요 라이브러리

```
pip install yt-dlp ffmpeg-python
```

❗FFmpeg 설치 필요

- Mac: brew install ffmpeg
- Ubuntu: sudo apt install ffmpeg
- Windows: FFmpeg 다운로드 후 환경 변수 설정
  - https://ffmpeg.org/download.html
  - https://github.com/BtbN/FFmpeg-Builds/releases
  - 윈도우용 zip 파일을 내려받아 압축을 풀어주세요.
  - Path 환경 변수에 ffmpeg.exe가 있는 폴더를 추가하세요.



유튜브 영상에서 오디오 추출하기

YouTube의 영상 내용 요약하기

```python
import yt_dlp
import os
import shutil

def download_youtube_mp3(video_id: str = 'gqXEEydkbh4'):
    # FFmpeg 설치 여부 확인
    if shutil.which("ffmpeg") is None:
        print("오류: FFmpeg이 설치되어 있지 않습니다. 설치 후 다시 시도하세요.")
        return None

    url = f"https://www.youtube.com/watch?v={video_id}"
    output_dir = "downloads"
    os.makedirs(output_dir, exist_ok=True)  # 다운로드 폴더 생성

    ydl_opts = {
        'format': 'bestaudio/best',
        'outtmpl': f"{output_dir}/%(id)s.%(ext)s",
        'postprocessors': [{
            'key': 'FFmpegExtractAudio',
            'preferredcodec': 'mp3',
            'preferredquality': '192',
        }],
    }

    try:
        with yt_dlp.YoutubeDL(ydl_opts) as ydl:
            info = ydl.extract_info(url, download=True)
            file_name = f"{info['id']}.mp3"
            file_path = os.path.join(output_dir, file_name)
        print(f"다운로드 완료! 파일 경로: {file_path}")
        return file_path
    except Exception as e:
        print(f"오류 발생: {e}")
        return None

audio_path = download_youtube_mp3('gqXEEydkbh4') # 테스트 실행
```

- yt_dlp: YouTube 및 여러 사이트에서 동영상을 다운로드할 수 있는 라이브러리.
- os: 파일 및 디렉터리 관련 작업 수행.
- shutil: 시스템 명령어 실행을 위한 라이브러리 (FFmpeg 설치 확인용).

- 'format': 'bestaudio/best' → 오디오 중 가장 좋은 품질로 다운로드.
- 'outtmpl': f"{output_dir}/%(title)s.%(ext)s" → 다운로드된 파일의 저장 경로 지정.
  - %(title)s: 동영상 제목
  - %(id)s: 동영상 아이디
  - %(ext)s: 원본 오디오 확장자
- 'postprocessors': FFmpeg을 사용하여 오디오를 MP3로 변환.
  - 'preferredcodec': 'mp3': 변환 코덱을 MP3로 설정.
  - 'preferredquality': '192': 오디오 비트레이트 192kbps 설정.

파일 자르기

YouTube의 영상 내용 요약하기

```python
import ffmpeg
import os

def trim_audio_file(file_path:str, max_size:int=24*1024*1024):
    file_size = os.path.getsize(file_path)  # 바이트 단위
    output_path = file_path
    if file_size > max_size:
        # ffmpeg로 오디오 비트레이트 가져오기
        probe = ffmpeg.probe(file_path)
        bitrate = int(probe['format']['bit_rate'])  # 비트레이트 (bps)

        # 최대 24MB로 자를 수 있는 재생 시간 계산
        max_size = 24 * 1024 * 1024  # 24MB
        max_duration = max_size * 8 / bitrate  # 초 단위로 변환

        # 출력 파일 경로 설정
        output_path = file_path.replace(".mp3", "_trimmed.mp3")

        # ffmpeg로 파일 자르기
        ffmpeg.input(file_path, ss=0, t=max_duration).output(output_path, codec="copy").run(overwrite_output=True)
    return output_path

trimmed_audio_path = trim_audio_file(audio_path)
```

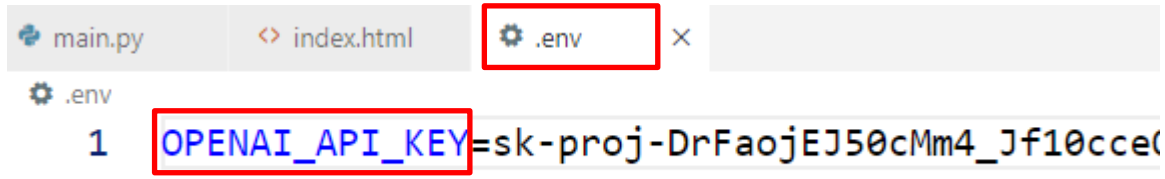
- ffmpeg.probe()를 사용하여 원본 오디오 파일의 비트레이트를 가져옵니다.
- 24MB(= 24 * 1024 * 1024 바이트)에 해당하는 재생 시간을 계산합니다.
- ffmpeg를 사용하여 해당 길이까지만 잘라서 저장합니다.

오디오 파일에서 텍스트 추출하기

YouTube의 영상 내용 요약하기

```python
from openai import OpenAI
import os
from dotenv import load_dotenv

load_dotenv()
api_key = os.getenv("OPENAI_API_KEY")
client = OpenAI(
    api_key = api_key
)

def transcribe_audio(audio_path):
    audio_file= open(audio_path, "rb")
    response = client.audio.transcriptions.create(
        model="whisper-1",
        file=audio_file,
        response_format="text"
    )
    return response

transcribed_text = transcribe_audio(trimmed_audio_path)
print("변환된 텍스트:\n", transcribed_text)
```

.env 파일 또는 시스템 환경변수에 OPENAI_API_KEY가 설정되어 있어야 합니다.



텍스트 내용 요약하기

YouTube의 영상 내용 요약하기

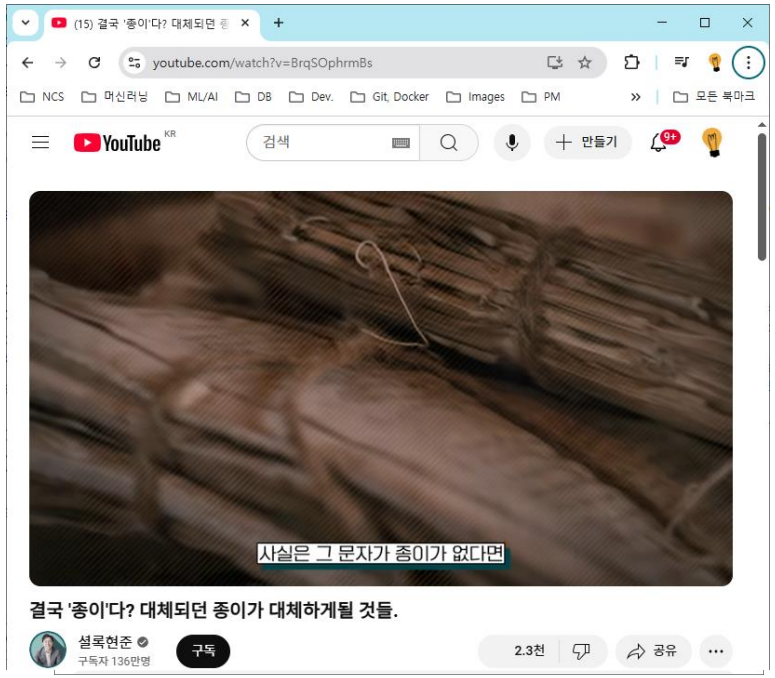
```
def summarize_text(text):
     response = client.chat.completions.create(
          model="gpt-4o-mini",
          temperature=0,
          messages=[
               { "role": "system", "content": "주어진 텍스트를 간결하고 명확하게 요약해 주세요." },
               { "role": "user", "content": text }
          ],
     )
     return response.choices[0].message.content

# 요약 실행
summary = summarize_text(transcribed_text)
print("요약된 내용:\n", summary)
```

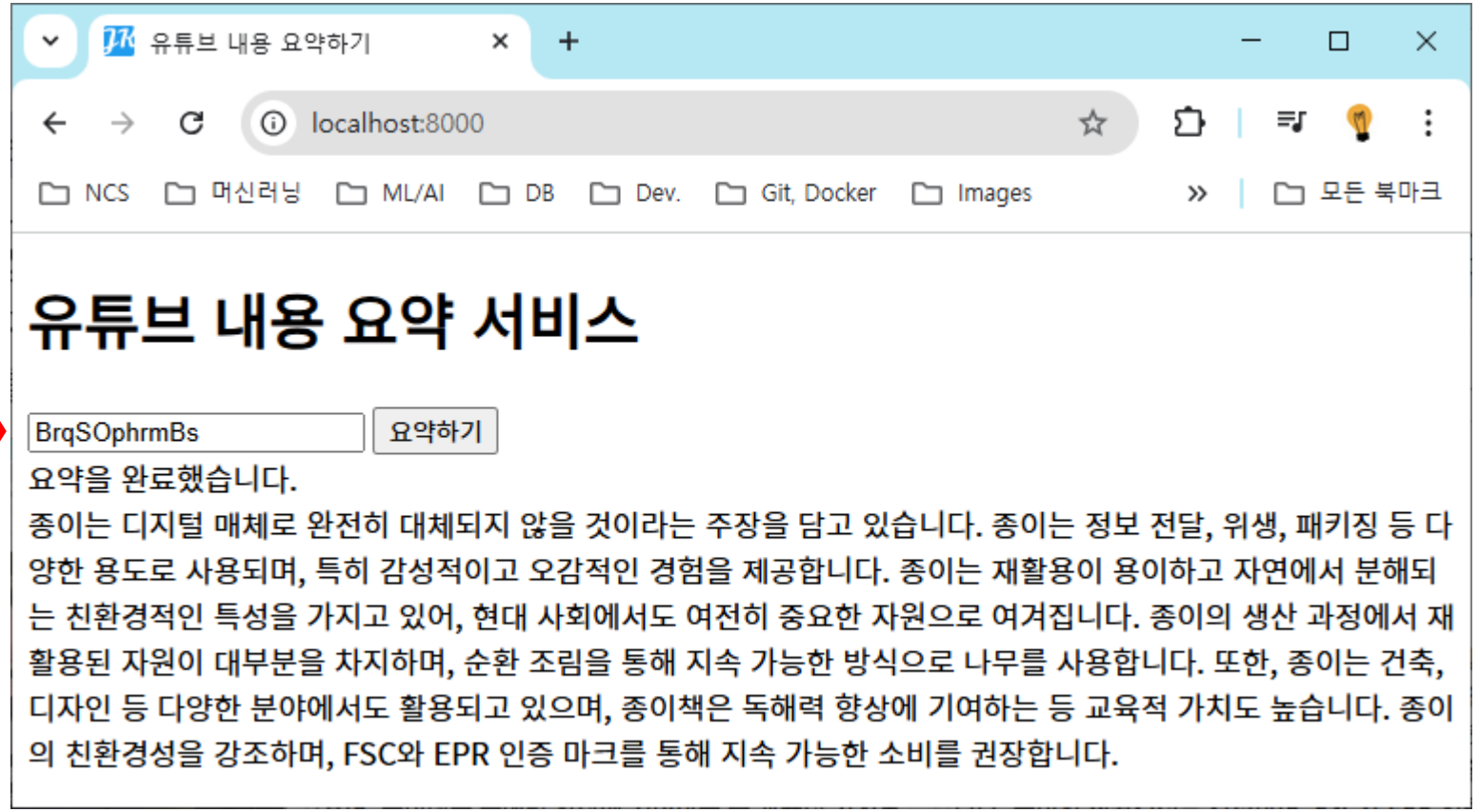
Youtube 영상 내용 요약 웹으로 서비스하기

생성형 인공지능 API 활용

Youtube 영상 내용 요약 웹으로 서비스하기

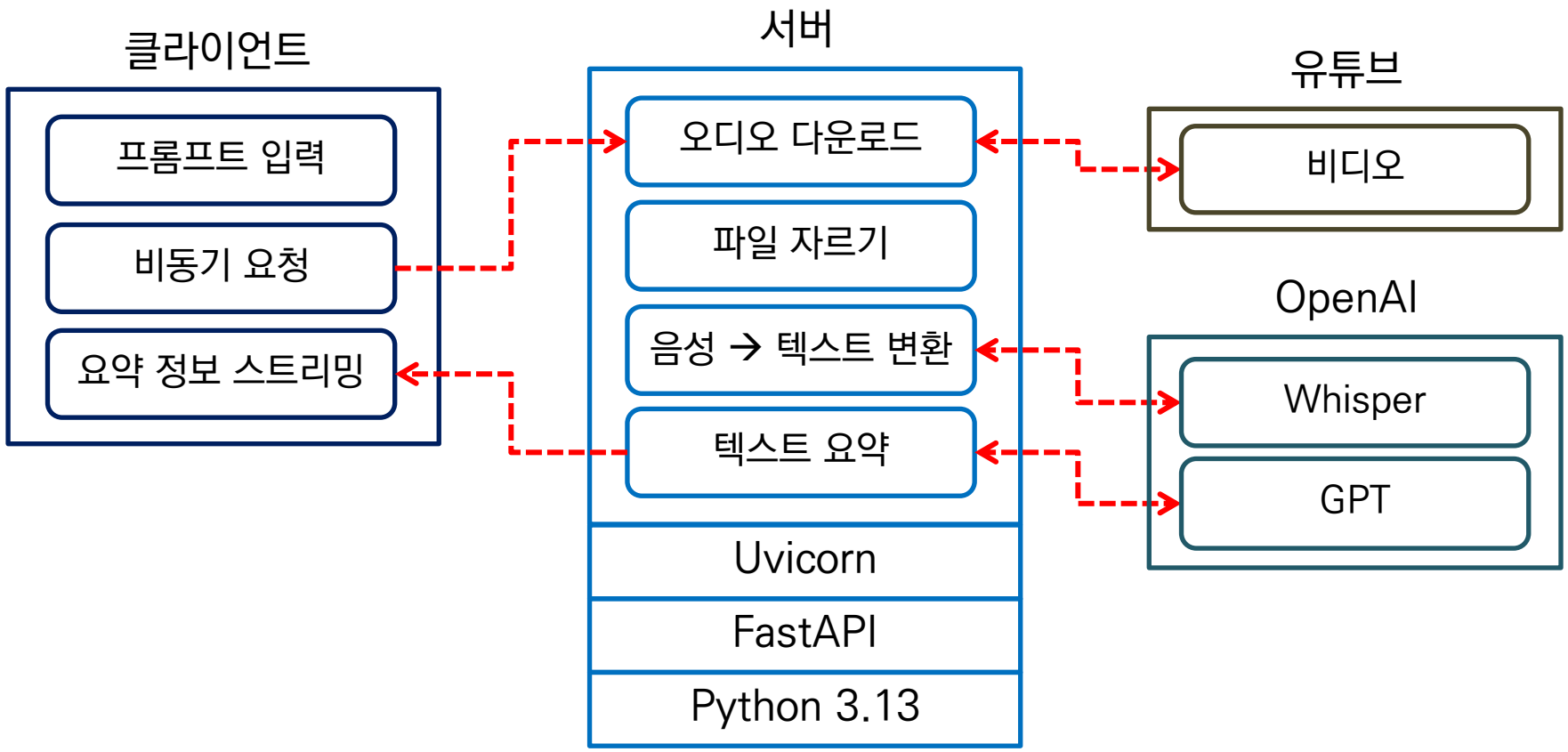
- yt_lib.py 모듈
- 유튜브 영상에서 오디오 추출하기
- 파일 자르기
- 텍스트 추출 및 요약하기





시스템 아키텍처

Youtube 영상 내용 요약 웹으로 서비스하기



yt_lib.py 모듈 (1/2)

Youtube 영상 내용 요약 웹으로 서비스하기

```python
import yt_dlp
import ffmpeg
import os
import shutil


def download_youtube_mp3(video_id: str = 'gqXEEydkbh4', output_dir: str = "downloads"):
    # FFmpeg 설치 여부 확인
    if shutil.which("ffmpeg") is None:
        print("오류: FFmpeg이 설치되어 있지 않습니다. 설치 후 다시 시도하세요.")
        return None

    url = f"https://www.youtube.com/watch?v={video_id}"

    os.makedirs(output_dir, exist_ok=True)  # 다운로드 폴더 생성

    ydl_opts = {
        'format': 'bestaudio/best',
        'outtmpl': f"{output_dir}/%(id)s.%(ext)s",
        'postprocessors': [{
            'key': 'FFmpegExtractAudio',
            'preferredcodec': 'mp3',
            'preferredquality': '192',
        }],
    }

    try:
        with yt_dlp.YoutubeDL(ydl_opts) as ydl:
            info = ydl.extract_info(url, download=True)
            file_name = f"{info['id']}.mp3"
            file_path = os.path.join(output_dir, file_name)

        print(f"다운로드 완료! 파일 경로: {file_path}")
        return file_path
    except Exception as e:
        print(f"오류 발생: {e}")
        return None
```

- yt_dlp: YouTube 및 여러 사이트에서 동영상을 다운로드할 수 있는 라이브러리.
- os: 파일 및 디렉터리 관련 작업 수행.
- shutil: 시스템 명령어 실행을 위한 라이브러리 (FFmpeg 설치 확인용).

- 'format': 'bestaudio/best' → 오디오 중 가장 좋은 품질로 다운로드.
- 'outtmpl': f"{output_dir}/%(id)s.%(ext)s" → 다운로드된 파일의 저장 경로 지정.
  - %(title)s: 동영상 제목
  - %(id)s: 동영상 아이디
  - %(ext)s: 원본 오디오 확장자
- 'postprocessors': FFmpeg을 사용하여 오디오를 MP3로 변환.
  - 'preferredcodec': 'mp3': 변환 코덱을 MP3로 설정.
  - 'preferredquality': '192': 오디오 비트레이트 192kbps 설정.

yt_lib.py 모듈 (1/2)

Youtube 영상 내용 요약 웹으로 서비스하기

```python
def trim_audio_file(file_path: str, max_size: int = 24*1024*1024):
    '''오디오 파일을 24메가 크기로 자릅니다.'''
    file_size = os.path.getsize(file_path)  # 바이트 단위
    output_path = file_path
    if file_size >= max_size:
        # ffmpeg로 오디오 비트레이트 가져오기
        probe = ffmpeg.probe(file_path)
        bitrate = int(probe['format']['bit_rate'])  # 비트레이트 (bps)

        # 최대 24MB로 자를 수 있는 재생 시간 계산
        max_duration = max_size * 8 / bitrate  # 초 단위로 변환

        # 출력 파일 경로 설정
        output_path = file_path.replace(".mp3", "_trimmed.mp3")

        # ffmpeg로 파일 자르기
        ffmpeg.input(file_path, ss=0, t=max_duration).output(output_path, codec="copy").run(overwrite_output=True)
    return output_path
```

- ffmpeg.probe()를 사용하여 원본 오디오 파일의 비트레이트를 가져옵니다.
- 24MB(= 24 * 1024 * 1024 바이트)에 해당하는 재생 시간을 계산합니다.
- ffmpeg를 사용하여 해당 길이까지만 잘라서 저장합니다.

main.py – 라이브러리 임포트 및 OpenAI 클라이언트 객체 생성

Youtube 영상 내용 요약 웹으로 서비스하기

```python
from fastapi import FastAPI, Request
from fastapi.responses import StreamingResponse, HTMLResponse
from fastapi.templating import Jinja2Templates
from fastapi.staticfiles import StaticFiles
from openai import OpenAI
from yt_lib import download_youtube_mp3, trim_audio_file
from dotenv import load_dotenv
import json
import os
import asyncio

# API 키 불러오기
load_dotenv()
api_key = os.getenv("OPENAI_API_KEY")
if not api_key:
    raise ValueError("환경 변수 OPENAI_API_KEY가 설정되지 않았습니다.")

client = OpenAI() # 최근 API는 자동으로 환경변수를 불러와서 client를 초기화 함

```

앞에서 만든 라이브러리의 함수를 import 하세요.

asyncio는 sleep() 함수를 사용하기 위해서 import 합니다.

- asyncio.sleep(seconds): 지정한 seconds만큼 비동기적으로 대기합니다.
- await 키워드와 함께 사용해야 다른 작업을 동시에 수행할 수 있습니다.
- 일반 time.sleep()과 달리 이벤트 루프를 블로킹하지 않습니다. 즉, 기다리는 동안 다른 코루틴이 실행될 수 있습니다.

main.py – app 초기화 및 기본 설정

Youtube 영상 내용 요약 웹으로 서비스하기

```
20  app = FastAPI()
21
22  # 템플릿 디렉토리 설정
23  templates = Jinja2Templates(directory="templates")
24
25  # static 폴더에 있는 파일 서빙 설정
26  app.mount("/static", StaticFiles(directory="static"), name="static")
27
28
29  @app.get("/", response_class=HTMLResponse)
30  async def read_root(request: Request):
31      return templates.TemplateResponse("index.html", {"request": request})
32
33
```

main.py – 오디오 다운로드 및 자르기, 텍스트 –〉 음성 변환

Youtube 영상 내용 요약 웹으로 서비스하기

```python
@app.get("/youtube/summary/{video_id}")
async def chat_stream(video_id: str = 'gqXEEydkbh4'):
    """YouTube 오디오 다운로드 → Whisper 변환 → GPT 요약 (스트리밍 응답)"""

    async def event_stream():
        # Step 1: 유튜브 오디오 다운로드 중
        yield f"data: {json.dumps({'status': 'downloading_audio', 'message': '유튜브 오디오를 내려받고 있습니다.'}, ensure_ascii=False)}\n\n"

        # 오디오 다운로드
        audio_path = download_youtube_mp3(video_id)
        yield f"data: {json.dumps({'status': 'download_complete', 'message': '오디오 다운로드가 완료되었습니다.'}, ensure_ascii=False)}\n\n"
        await asyncio.sleep(1) # 너무 빠르게 메시지가 전송되는 것을 방지하고 자연스럽게 출력되도록 조절.

        # 오디오 자르기
        trimmed_audio_path = trim_audio_file(audio_path)
        yield f"data: {json.dumps({'status': 'trim_audio', 'message': '오디오 자르기가 완료되었습니다.'}, ensure_ascii=False)}\n\n"
        await asyncio.sleep(1) # 너무 빠르게 메시지가 전송되는 것을 방지하고 자연스럽게 출력되도록 조절.

        # Step 2: 음성 → 텍스트 변환 중
        yield f"data: {json.dumps({'status': 'transcribing_audio', 'message': '음성을 텍스트로 변환 중입니다.'}, ensure_ascii=False)}\n\n"

        # Whisper를 이용한 음성 인식
        with open(trimmed_audio_path, "rb") as audio_file:
            text = client.audio.transcriptions.create(
                model="whisper-1",
                file=audio_file,
                response_format="text"
            )
        yield f"data: {json.dumps({'status': 'transcription_complete', 'message': '음성 변환을 완료했습니다.'}, ensure_ascii=False)}\n\n"
```

응답 데이터 형식에 있어서 유효한 필드 이름이 정해져 있음. data 필드는 서버가 보낼 메시지이며, 값이 없으면 수신측에서 message 이벤트가 발생하지 않음.

yt_lib.py 모듈의 함수를 호출해서 오디오를 내려받은 후 오디오를 24M 만큼만 자릅니다.

휘스퍼를 이용해서 음성을 텍스트로 변환합니다.

개행 문자 두 개로 메시지의 끝을 표시해야 함.

main.py – 내용 요약 및 스트리밍

Youtube 영상 내용 요약 웹으로 서비스하기

```python
        # Step 3: GPT 요약 시작
        yield f"data: {json.dumps({'status': 'summarizing_text', 'message': '내용을 요약하고 있습니다.'}, ensure_ascii=False)}\n\n"

        # GPT-4o-mini 요약 (스트리밍)
        stream = client.chat.completions.create(
            model="gpt-4o-mini",
            temperature=0,
            messages=[
                {"role": "system", "content": "주어진 텍스트를 간결하고 명확하게 요약해 주세요."},
                {"role": "user", "content": text}
            ],
            stream=True
        )

        for chunk in stream:
            content = chunk.choices[0].delta.content if chunk.choices and chunk.choices[0].delta else None
            if content:
                yield f"data: {json.dumps({'status': 'processing', 'token': content}, ensure_ascii=False)}\n\n"
                await asyncio.sleep(0)  # 다른 작업을 수행할 수 있도록 컨텍스트 스위칭

        # Step 4: 모든 과정 완료
        yield f"data: {json.dumps({'status': 'complete', 'message': '요약이 완료되었습니다!'}, ensure_ascii=False)}\n\n"

    return StreamingResponse(event_stream(), media_type="text/event-stream")
```

GPT를 이용해서 텍스트를 요약합니다.

- asyncio.sleep(0)은 **"나는 바로 끝날 테니, 다른 코루틴이 실행될 기회를 줘!"** 라는 의미입니다.
- CPU를 점유하지 않고 다른 코루틴이 실행될 수 있도록 **컨텍스트 스위칭 (Task switching)**을 유도합니다.
- 일반적인 함수에서는 동시성 효과가 없지만, 여러 코루틴이 있을 때는 다른 코루틴이 실행될 틈을 만들어줍니다.

main.py – main 함수

Youtube 영상 내용 요약 웹으로 서비스하기

```
90  if __name__ == "__main__":
91      import uvicorn
92      import asyncio
93
94      async def main():
95          config = uvicorn.Config(app, host="0.0.0.0", port=8000)
96          server = uvicorn.Server(config)
97          await server.serve()
98
99      asyncio.run(main())
100
```

templates/index.html – 입력 양식

Youtube 영상 내용 요약 웹으로 서비스하기

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>유튜브 내용 요약하기</title>
    <link rel="icon" href="/static/favicon.ico" type="image/x-icon">
</head>
<body>
<h1>유튜브 내용 요약 서비스</h1>
<input type="text" id="videoId" value="KPFVuGJ4WII" placeholder="유튜브 비디오 아이디를 입력하세요." size="20">
<button id="submit">요약하기</button>
<div id="loader" style="display: none;">요약하는 중...</div>
<div id="status" style="display: block;"></div>
<div id="response"></div>

```

templates/index.html – 이벤트 처리

Youtube 영상 내용 요약 웹으로 서비스하기

```
17  <script>
18  document.getElementById('submit').addEventListener('click', function() {
19      const videoId = document.getElementById('videoId').value;
20      const loader = document.getElementById('loader');
21      const status = document.getElementById('status');
22      const response = document.getElementById('response');
23
24      if (!videoId) {
25          alert('유튜브 비디오 아이디를 입력하세요.');
26          return;
27      }
28
29      // 초기 UI 설정
30      response.innerHTML = '';
31      status.innerHTML = '';
32      loader.style.display = 'block';
33
34      // 기존 EventSource가 있다면 닫기
35      if (window.eventSource) {
36          window.eventSource.close();
37      }
38
```

templates/index.html – 서버의 응답 메시지 처리

Youtube 영상 내용 요약 웹으로 서비스하기

```
42      window.eventSource.onmessage = function(event) {
43          const data = JSON.parse(event.data);
44
45          // 상태 메시지 업데이트
46          if (data.status !== 'processing' && data.status !== 'complete') {
47              status.innerHTML += `${data.message}<br>`; // 서버에서 받은 메시지를 그대로 출력
48          }
49
50          // 요약 진행 중인 경우
51          if (data.status === 'processing') {
52              response.innerText += data.token;
53          }
54
55          // 요약 완료
56          if (data.status === 'complete') {
57              loader.style.display = 'none';
58              status.innerHTML += `${data.message}<br>`;
59              window.eventSource.close();
60          }
61      };
62
```

templates/index.html – 오류 처리

Youtube 영상 내용 요약 웹으로 서비스하기

```
63      window.eventSource.onerror = function(err) {
64          console.error('EventSource 오류:', err);
65          status.innerHTML += '<br>오류 발생: 서버와 연결이 끊어졌습니다.';
66          loader.style.display = 'none';
67          window.eventSource.close();
68      };
69  });
70  </script>
71
72  </body>
73  </html>
```

참고: Server Sent Event란?

Youtube 영상 내용 요약 웹으로 서비스하기

- 클라이언트 API와 푸시(Push) 데이터를 정의한 사양.
- 서버로부터 푸시된 데이터를 수집하여 일반적인 DOM이벤트처럼 처리가 가능함
- SSE를 사용하면 서버의 갱신된 데이터를 클라이언트에 보여주기 위해 AJAX와 setInterval()을 사용할 필요가 없음.

- 이벤트 소스 객체 생성
  - var eventSource = new EventSource("message.jsp");.
- 이벤트 소스에 onmessage 핸들러 지정
  - eventSource.onmessage = function(event) { }
  - eventSource.addEventListener("message", function(event) { }, false);

이벤트 핸들러	이벤트 핸들러 이벤트 타입
onopen	open
onmessage	message
onerror	error

EventSource의 이벤트

참고: SSE 푸시 데이터 형식

- SSE는 EventSource가 자동으로 해석할 수 있도록 서버로부터의 푸시되는 데이터의 규칙이 정해져 있음.
  - mime 타입은 text/event-stream, charset 파라미터는 필요 없음.
  - 문자 인코딩은 UTF-8.
  - 줄 바꿈 코드는 '\r\n' '\n' '\r' 모두 유효함.
  - ':'(콜론)으로 시작하는 줄은 주석.
  - 빈 줄이나 파일 끝은 이벤트 구분자가 됨.
  - 주석이나 빈 줄 이외는 '필드 이름: 필드 값' 이라는 형식으로 작성.
  - 필드 값이 없을 경우 콜론 다음을 생략할 수 있음.
  - 가독성을 좋게 하기 위해 콜론과 필드 값 사이에 공백 하나를 포함할 수 있음.
  - 개행 문자 두 개로 메시지의 끝을 표시해야 함.

참고: SSE 응답데이터 형식의 유효한 필드

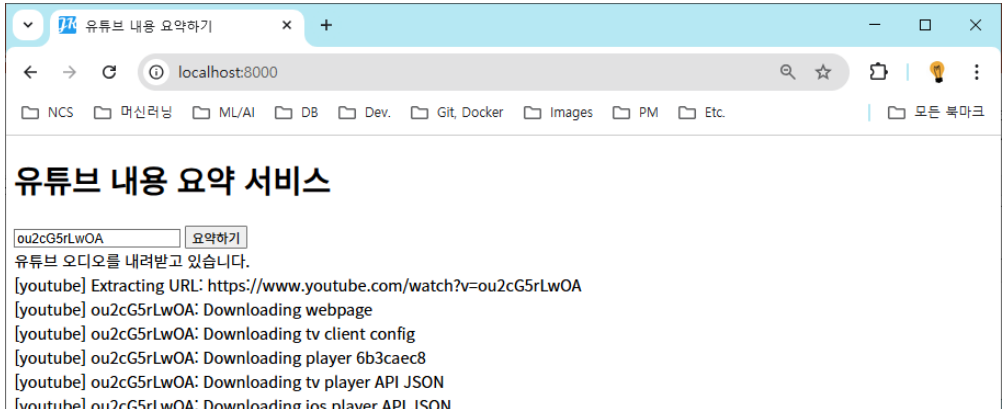
- 응답 데이터 형식에 있어서 유효한 필드 이름이 또한 정해져 있음

- event
  - 브라우저가 발생시킬 이벤트. 기본은 message 임.
- data
  - 서버가 보낼 메시지
  - 한 번에 여러 개를 보낼 수 있음
  - 값이 없으면 수신측에서 message 이벤트가 발생하지 않음
- id
  - 이벤트 ID이다.
  - 마지막으로 수신한 이벤트의 ID는 MessageEvent의 lastEventId 속성으로 참조할 수 있으며 요청 헤더에 'Last-Event-ID'로 포함됨.
- retry
  - 다음 재시도 까지 시간 간격을 밀리 초 단위로 지정.
  - 생략 시 다음 재시도 간격은 브라우저에 따라 다를 수 있음.

서버의 로그를 웹 화면에 출력하려면...

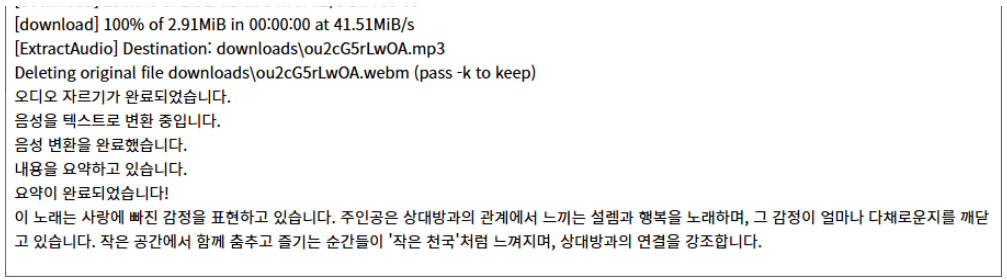
서버의 로그를 웹 화면에 출력하려면...

- 예: https://www.youtube.com/watch?v=ou2cG5rLwOA
- subprocess 모듈 이용 다운로드
- 로그를 실시간으로 웹에 출력




```
38    async def event_stream():
39        # Step 1: 유튜브 오디오 다운로드 중
40        yield f"data: {json.dumps({'status': 'downloading_audio', 'message': '유튜브 오디오를 내려받고 있습니다.'}, ensure_ascii=False)}\n\n"
41
42        # 오디오 다운로드
43        audio_path = download_youtube_mp3(video_id)
44        yield f"data: {json.dumps({'status': 'download_complete', 'message': '오디오 다운로드가 완료되었습니다.'}, ensure_ascii=False)}\n\n"
45        await asyncio.sleep(1) # 너무 빠르게 메시지가 전송되는 것을 방지하고 자연스럽게 출력되도록 조절.
46
```

이 코드가 다음 페이지의 파란색 박스 내용으로 바뀝니다.



서버의 로그를 웹 화면에 출력하려면...

Youtube 영상 내용 요약 웹으로 서비스하기

이 코드는 Python의 subprocess.Popen을 사용하여 yt-dlp 명령을 실행하고, YouTube 동영상을 다운로드하며 발생하는 로그를 실시간으로 웹 브라우저에 전송하는 역할을 합니다.

```python
async def event_stream():
    # Step 1: 유튜브 오디오 다운로드 중
    yield f"data: {json.dumps({'status': 'downloading_audio', 'message': '유튜브 오디오를 내려받고 있습니다.'}, ensure_ascii=False)}\n\n"
    await asyncio.sleep(1) # 너무 빠르게 메시지가 전송되는 것을 방지하고 자연스럽게 출력되도록 조절.

    # YouTube 다운로드 실행 (스트리밍 로그 전송)
    process = subprocess.Popen(
        ["yt-dlp", "-x", "--audio-format", "mp3", "-o", "downloads/%(id)s.%(ext)s", "-k", f"https://www.youtube.com/watch?v={video_id}"],
        stdout=subprocess.PIPE,
        stderr=subprocess.PIPE,
        text=True
    )

    # yt-dlp의 로그를 실시간으로 웹 브라우저에 출력
    while True:
        output = process.stdout.readline()
        if output == "" and process.poll() is not None:
            break
        if output:
            yield f"data: {json.dumps({'status': 'downloading_audio', 'message': output.strip()}, ensure_ascii=False)}\n\n"
            await asyncio.sleep(1) # 너무 빠르게 메시지가 전송되는 것을 방지하고 자연스럽게 출력되도록 조절.

    if process.returncode != 0:
        error_message = process.stderr.read()
        yield f"data: {json.dumps({'status': 'error', 'message': f'유튜브 다운로드 오류: {error_message}'}, ensure_ascii=False)}\n\n"
        return

    audio_path = f"downloads/{video_id}.mp3"
```

yt-dlp 프로세스가 종료되었는지 확인

- -x: 오디오만 추출(비디오 다운로드 후 오디오 파일로 변환)
- --audio-format mp3: mp3 형식으로 변환
- -o "downloads/%(id)s.%(ext)s": 파일을 "downloads" 폴더에 저장, 유튜브 비디오 ID를 파일명으로 사용
- -k: 원본 비디오 파일을 삭제하지 않음 (예: .webm 파일 유지)
- f"https://www.youtube.com/watch?v={video_id}": 다운로드할 유튜브 영상의 URL
- stdout=subprocess.PIPE: yt-dlp의 실행 결과(출력)를 Python에서 읽을 수 있도록 stdout을 파이프(Pipe)로 설정
- stderr=subprocess.PIPE: 실행 중 오류 메시지를 캡처할 수 있도록 설정
- text=True: 출력을 바이트가 아닌 문자열 형태로 읽기 위해 설정

templates/index.html – 입력 양식

Youtube 영상 내용 요약 웹으로 서비스하기

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>유튜브 내용 요약하기</title>
    <link rel="icon" href="/static/favicon.ico" type="image/x-icon">
    <style>
        .status-message {
            font-size: 16px;
            font-weight: bold;
            margin: 5px 0;
        }
    </style>
</head>
<body>
    <h1>유튜브 내용 요약 서비스</h1>
    <input type="text" id="videoId" value="KPFVuGJ4WII" placeholder="유튜브 비디오 아이디를 입력하세요." size="20">
    <button id="submit">요약하기</button>

    <div id="status"></div>
    <div id="response" style="margin-top: 20px;"></div>

```

templates/index.html – 이벤트 처리

Youtube 영상 내용 요약 웹으로 서비스하기

```
24      <script>
25          document.getElementById('submit').addEventListener('click', function() {
26              const videoId = document.getElementById('videoId').value;
27              const statusContainer = document.getElementById('status');
28              const responseContainer = document.getElementById('response');
29
30              if (!videoId) {
31                  alert('유튜브 비디오 아이디를 입력하세요.');
32                  return;
33              }
34
35              // 초기화
36              responseContainer.innerHTML = '';
37              statusContainer.innerHTML = '';
38
39              // 기존 EventSource가 있다면 닫기
40              if (window.eventSource) {
41                  window.eventSource.close();
42              }
43
44              // 새로운 EventSource 생성
45              window.eventSource = new EventSource(`/youtube/summary/${videoId}`);
46
```

templates/index.html – 서버의 응답 메시지 처리

Youtube 영상 내용 요약 웹으로 서비스하기

```
47      window.eventSource.onmessage = function(event) {
48          const data = JSON.parse(event.data);
49
50          // 다운로드 진행 중 메시지는 같은 줄에 덮어쓰기
51          if (data.status !== 'processing' && data.status !== 'complete') {
52              if (data.message.includes("[download]")) {
53                  updateDownloadStatus(data.message);
54              } else {
55                  updateStatus(data.message);
56              }
57          }
58
59          // 요약 진행 중인 경우
60          if (data.status === 'processing') {
61              responseContainer.innerText += data.token;
62          }
63
64          // 요약 완료
65          if (data.status === 'complete') {
66              updateStatus(data.message);
67              window.eventSource.close();
68          }
69      };
```

templates/index.html – 오류 처리

Youtube 영상 내용 요약 웹으로 서비스하기

```
71          window.eventSource.onerror = function(err) {
72              console.error('EventSource 오류:', err);
73              updateStatus('오류 발생: 서버와 연결이 끊어졌습니다.');
74              window.eventSource.close();
75          };
76      });
```

templates/index.html – 메시지 덮어쓰기 처리

Youtube 영상 내용 요약 웹으로 서비스하기

```
78          // 상태 메시지 업데이트 함수 (일반 메시지 추가)
79          function updateStatus(message) {
80              const statusContainer = document.getElementById('status');
81              const statusDiv = document.createElement('div');
82              statusDiv.className = 'status-message';
83              statusDiv.innerText = message;
84              statusContainer.appendChild(statusDiv);
85          }
86
87          // 다운로드 상태 업데이트 (덮어쓰기)
88          function updateDownloadStatus(message) {
89              let downloadStatus = document.getElementById('download-status');
90              if (!downloadStatus) {
91                  downloadStatus = document.createElement('div');
92                  downloadStatus.id = 'download-status';
93                  downloadStatus.className = 'status-message';
94                  document.getElementById('status').appendChild(downloadStatus);
95              }
96              downloadStatus.innerText = message;
97          }
98      </script>
99  </body>
100 </html>
```

templates/index.html – 리팩토링한 코드

Youtube 영상 내용 요약 웹으로 서비스하기

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>유튜브 내용 요약하기</title>
    <link rel="icon" href="/static/favicon.ico" type="image/x-icon">
    <style>
        .status-message {
            font-size: 16px;
            font-weight: bold;
            margin: 5px 0;
        }
    </style>
</head>
<body>
    <h1>유튜브 내용 요약 서비스</h1>
    <input type="text" id="videoId" value="KPFVuGJ4WII" placeholder="유튜브 비디오 아이디를 입력하세요." size="20">
    <button id="submit">요약하기</button>

    <div id="status"></div>
    <div id="response" style="margin-top: 20px;"></div>

    <script>
        document.getElementById('submit').addEventListener('click', () => {
            const videoId = document.getElementById('videoId').value.trim();
            if (!videoId) return alert('유튜브 비디오 아이디를 입력하세요.');

            // 초기화
            document.getElementById('status').innerHTML = '';
            document.getElementById('response').innerHTML = '';

            // 기존 EventSource 종료
            if (window.eventSource) {
                window.eventSource.close();
            }

            window.eventSource = new EventSource(`/youtube/summary/${videoId}`);
```

templates/index.html – 리팩토링한 코드

Youtube 영상 내용 요약 웹으로 서비스하기

```
            window.eventSource.onmessage = ({ data }) => {
                const { status, message, token } = JSON.parse(data);

                if (status === 'processing') {
                    document.getElementById('response').innerText += token;
                } else {
                    updateStatus(message, status === 'downloading_audio');
                }

                if (status === 'complete') window.eventSource.close();
            };

            window.eventSource.onerror = () => {
                updateStatus('오류 발생: 서버와 연결이 끊어졌습니다.');
                window.eventSource.close();
            };
        });

        function updateStatus(message, overwrite = false) {
            const statusContainer = document.getElementById('status');
            if (overwrite) {
                let downloadStatus = document.getElementById('download-status');
                if (!downloadStatus) {
                    downloadStatus = document.createElement('div');
                    downloadStatus.id = 'download-status';
                    downloadStatus.className = 'status-message';
                    statusContainer.appendChild(downloadStatus);
                }
                downloadStatus.innerText = message;
            } else {
                statusContainer.innerHTML += `<div class="status-message">${message}</div>`;
            }
        }
    </script>
</body>
</html>
```

생성형 인공지능(OpenAPI) 활용 – Youtube 영상 요약하기

OpenAI Chat GPT

OpenAPI 활용

https://chat.openai.com/