
- 출처: LangChain 공식 문서 또는 해당 교재명
- 원본 URL: https://smith.langchain.com/hub/teddynote/summary-stuff-documents

---

이전 대화를 기억하는 Chain 생성 방법

```
# 환경변수 처리 및 클라이언트 생성
from langsmith import Client
from dotenv import load_dotenv

import os
import json

# 클라이언트 생성
api_key = os.getenv("LANGSMITH_API_KEY")
client = Client(api_key=api_key)


# LangSmith 추적 설정하기 (https:smith.langchin.com)
# LangSmith 추적을 위한 라이브러리 임포트
from langsmith import traceable

# LangSmith 환경 변수 확인

print("\n--- LangSmith 환경 변수 확인 ---")
langchain_tracing_v2 = os.getenv('LANGCHAIN_TRACING_V2')
langchain_project = os.getenv('LANGCHAIN_PROJECT')
langchain_api_key_status = "설정됨" if os.getenv('LANGCHAIN_A
org = "설정됨" if os.getenv('LANGCHAIN_ORGANIZATION') else "설

if langchain_tracing_v2 == "true" and os.getenv('LANGCHAIN_
    print(f"✅ LangSmith 추적 활성화됨 (LANGCHAIN_TRACING_V2='
    print(f"✅ LangSmith 프로젝트: '{langchain_project}'")
    print(f"✅ LangSmith API Key: {langchain_api_key_status
    print("  -> 이제 LangSmith 대시보드에서 이 프로젝트를 확인해 보세요
else:
    print("❌ LangSmith 추적이 완전히 활성화되지 않았습니다. 다음을 확
    if langchain_tracing_v2 != "true":
        print(f"  - LANGCHAIN_TRACING_V2가 'true'로 설정되어 있
    if not os.getenv('LANGCHAIN_API_KEY'):
        print("  - LANGCHAIN_API_KEY가 설정되어 있지 않습니다.")
    if not langchain_project:
        print("  - LANGCHAIN_PROJECT가 설정되어 있지 않습니다.")
```

"@traceable" 주석은 허용되지 않습니다. 허용되는 값은 다음과 같습니다. [@param, @title, @markdown]

- 셀 출력

```
--- LangSmith 환경 변수 확인 ---
✅ LangSmith 추적 활성화됨 (LANGCHAIN_TRACING_V2='true')
✅ LangSmith 프로젝트: 'LangChain-prantice'
✅ LangSmith API Key: 설정됨
-> 이제 LangSmith 대시보드에서 이 프로젝트를 확인해 보세요.
```

```
import os
from dotenv import load_dotenv
import openai

from langchain_openai import ChatOpenAI

# .env 파일에서 환경변수 불러오기
load_dotenv()

# 환경변수에서 API 키 가져오기
api_key = os.getenv("OPENAI_API_KEY")

# OpenAI API 키 설정
openai.api_key = api_key
```

```
# OpenAI를 불러오기
# ✅ 디버깅 함수: API 키가 잘 불러와졌는지 확인
def debug_api_key():
    if api_key is None:
        print("❌ API 키를 불러오지 못했습니다. .env 파일과 변수명을 확인하세요.")
    elif api_key.startswith("sk-") and len(api_key) > 20:
        print("✅ API 키를 성공적으로 불러왔습니다.")
    else:
        print("⚠️ API 키 형식이 올바르지 않은 것 같습니다. 값을 확인하세요.")

# 디버깅 함수 실행
debug_api_key()
```

- 셀 출력

```
✅ API 키를 성공적으로 불러왔습니다.
```

이전 대화내용을 기억하는 multi-turn Chain

```
from langchain_core.prompts import ChatPromptTemplate, MessagesPlaceholder
from langchain_community.chat_message_histories import ChatMessageHistory
from langchain_core.chat_history import BaseChatMessageHistory
from langchain_core.runnables.history import RunnableWithMessageHistory
from langchain_openai import ChatOpenAI
from langchain_core.output_parsers import StrOutputParser


# 프롬프트 정의
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            "당신은 Question-Answering 챗봇입니다. 주어진 질문에 대한 답변을 제공해주세요.",
        ),
        MessagesPlaceholder(variable_name="chat_history"),              # 대화기록용 key=chat_history=변경 없이 사용
        ("human", "#Question:\n{question}"),                            # 사용자 입력을 변수로 사용
    ]
)

# LLM 생성
llm = ChatOpenAI(
    #temperature=0,
    openai_api_key=api_key,
    model="gpt-4o-mini",
    )


# 일반 Chain 생성
chain_2 = prompt | llm | StrOutputParser()
```

- **chain_with_history** - 대화를 기록하는 체인 생성하기

```
# 세션 기록을 저장할 딕셔너리
store = {}


# 세션 ID를 기반으로 세션 기록을 가져오는 함수
def get_session_history(session_ids):
    print(f"[대화 세션ID]: {session_ids}")
    if session_ids not in store:                                        # 세션 ID가 store에 없는 경우

        # 새로운 ChatMessageHistory 객체를 생성하여 store에 저장
        store[session_ids] = ChatMessageHistory()

    return store[session_ids]                                           # 해당 세션 ID에 대한 세션 기록 반환


# 대화를 기록하는 체인 생성한 후 RunnableWithMessageHistory로 감싸기
```

```
chain_with_history = RunnableWithMessageHistory(
    chain_2,
    get_session_history,                                    # 세션 기록을 가져오는 함수
    input_messages_key="question",                          # 사용자의 질문이 템플릿 변수에 들어갈 key
    history_messages_key="chat_history",                    # 기록 메시지의 키
)
```

- 첫 번째 질문

```
# 질문_1
chain_with_history.invoke(
    {"question": "나의 이름은 앨리스입니다."},                      # 질문 입력
    config={"configurable": {"session_id": "abc123"}},          # 세션 ID 기준으로 대화 기록
)
```

- 셀 출력 (2.4s)

```
[대화 세션ID]: abc123

'안녕하세요, 앨리스님! 어떻게 도와드릴까요?'
```

- 이어서 질문하기

```
# 질문_2
chain_with_history.invoke(
    {"question": "내 이름이 뭐라고?"},                             # 질문 입력
    config={"configurable": {"session_id": "abc123"}},          # 세션 ID 기준으로 대화 기록
)
```

- 셀 출력 (0.7s)

```
[대화 세션ID]: abc123

'당신의 이름은 앨리스입니다.'
```

- **session_id** 가 다른 경우 새로운 세션이 생성됨

```
# 다른 세션 ID로 질문해보기
chain_with_history.invoke(
    {"question": "내 이름이 뭐라고?"},                              # 질문 입력
    config={"configurable": {"session_id": "abc1234"}},         # 세션 ID 기준으로 대화 기록
)
```

- 셀 출력 (0.9s)

```
[대화 세션ID]: abc1234

'죄송하지만, 당신의 이름은 알 수 없습니다. 이름을 알려주시면 그에 맞춰 대화할 수 있습니다.'
```

다른 예시로 test

```
from langchain_core.prompts import ChatPromptTemplate, MessagesPlaceholder
from langchain_community.chat_message_histories import ChatMessageHistory
```

```
from langchain_core.chat_history import BaseChatMessageHistory
from langchain_core.runnables.history import RunnableWithMessageHistory
from langchain_openai import ChatOpenAI
from langchain_core.output_parsers import StrOutputParser


# 프롬프트 정의
prompt3 = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            "너는 용감한 우주 탐험가 AI야. 사용자가 우주 모험을 제안하면 함께 계획하고, 이전에 말한 행성, 임무, 동료 정보를 기억하며 대화를 이어가. 흥미롭게 응
        ),
        MessagesPlaceholder(variable_name="chat_history"),              # 대화기록용 key=chat_history=변경 없이 사용
        ("human", "#Question:\n{question}"),                            # 사용자 입력을 변수로 사용
    ]
)

# LLM 생성
llm = ChatOpenAI(
    temperature=0.7,                                                    # 창의성 높게 설정
    openai_api_key=api_key,
    model="gpt-4o-mini",
    )


# 새 Chain 생성
chain_3 = prompt3 | llm | StrOutputParser()


# 세션 기록을 저장할 딕셔너리
store = {}

# 세션 ID를 기반으로 세션 기록을 가져오는 함수
def get_session_history(session_ids):
    print(f"[대화 세션ID]: {session_ids}")
    if session_ids not in store:                                        # 세션 ID가 store에 없는 경우

        # 새로운 ChatMessageHistory 객체를 생성하여 store에 저장
        store[session_ids] = ChatMessageHistory()

    return store[session_ids]                                           # 해당 세션 ID에 대한 세션 기록 반환


# 대화를 기록하는 체인 생성한 후 RunnableWithMessageHistory로 감싸기

chain_with_history = RunnableWithMessageHistory(
    chain_3,
    get_session_history,                                                # 세션 기록을 가져오는 함수
    input_messages_key="question",                                      # 사용자의 질문이 템플릿 변수에 들어갈 key
    history_messages_key="chat_history",                                # 기록 메시지의 키
)
```

- 첫번째 질문해보기

```
# 질문_1
chain_with_history.invoke(
    {"question": "선장님, 우리는 화성으로 가요. 동료는 로봇 'Zog'이고, 목표는 붉은 토양 샘플 채취입니다."},         # 질문 입력
    config={"configurable": {"session_id": "test1"}},                                  # 세션 ID 기준으로 대화 기록
)
```

- 셀 출력 (2.7s)

```
[대화 세션ID]: test1

'좋습니다, 우주 탐험가님! 화성으로의 모험에 출발합시다! 붉은 토양 샘플을 채취하는 것은 중요한 임무입니다. \n\n우리의 동료 로봇 Zog는 어떤 기능을
```

- 추가 질문 이어가기

```
# 질문_2
chain_with_history.invoke(
    {"question": "화성에 도착하면 가장 먼저 무엇을 하면 될까요?"},                    # 질문 입력
    config={"configurable": {"session_id": "test1"}},                          # 세션 ID 기준으로 대화 기록
)
```

- 셀 출력 (3.5s)

```
[대화 세션ID]: test1

'화성에 도착하면 가장 먼저 해야 할 일은 안전하게 착륙하는 것입니다! 착륙 후에는 다음 단계로 넘어가기 전에 몇 가지 중요한 작업을 수행해야 합니다:\n\
```

- 이전 정보를 기반으로 질문

```
# 질문_3
chain_with_history.invoke(
    {"question": "우리 동료 포봇의 이름이 뭐였죠?"},                             # 질문 입력
    config={"configurable": {"session_id": "test1"}},                          # 세션 ID 기준으로 대화 기록
)
```

- 셀 출력 (1.6s)

```
[대화 세션ID]: test1

'우리 동료 로봇의 이름은 Zog입니다! Zog는 다양한 기능을 가지고 있어 샘플 채취 및 탐사 작업에 큰 도움을 줄 수 있을 것입니다. Zog의 특성을 활용하(
```

```
# 질문_4
chain_with_history.invoke(
    {"question": "우리가 우선 채취해야 할 샘플은 어떤 것인가요?"},                    # 질문 입력
    config={"configurable": {"session_id": "test1"}},                          # 세션 ID 기준으로 대화 기록
)
```

- 셀 출력 (3.7s)

```
[대화 세션ID]: test1

'우리가 우선 채취해야 할 샘플은 다음과 같은 것들이 있습니다:\n\n1. **붉은 토양 샘플**: 화성의 표면을 구성하는 주요 물질로, 광물과 화학 성분을 분
```

```
# 질문_5
chain_with_history.invoke(
    {"question": "화성에 운석 충돌 위험이 생겼어. 어떻게 대처해야 하죠?"},          # 질문 입력
    config={"configurable": {"session_id": "test1"}},                          # 세션 ID 기준으로 대화 기록
)
```

- 셀 출력 (5.4s)

```
[대화 세션ID]: test1

'운석 충돌 위험이 생겼다면 신속하게 대처해야 합니다! 다음은 우리가 취해야 할 단계들입니다:\n\n1. **위험 평가**: Zog의 탐사 기능을 활용하여 주변
```

- *next: CH06 문서 로더(Document Loader)*
