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

---

2. `Runnable 구조`(그래프) `검토`

1) `Runnables 구조 검토`

- `Runnable` 생성 후 → 검사 → 어떤 일이 일어나는지 과정을 파악할 수 있음

- **`환경설정`**

```
# API 키를 환경변수로 관리하기 위한 설정 파일
from dotenv import load_dotenv

# API 키 정보 로드
load_dotenv()                           # True
```

```
from langsmith import Client
from langsmith import traceable

import os

# 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_API_KEY') else "설정되지 않음" # API 키 값은 직접 출력하지 않음

if langchain_tracing_v2 == "true" and os.getenv('LANGCHAIN_API_KEY') and langchain_project:
    print(f"✅ LangSmith 추적 활성화됨 (LANGCHAIN_TRACING_V2='{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 확인하세요:")
    if langchain_tracing_v2 != "true":
        print(f"  - LANGCHAIN_TRACING_V2가 'true'로 설정되어 있지 않습니다 (현재: '{langchain_tracing_v2}').")
    if not os.getenv('LANGCHAIN_API_KEY'):
        print("  - LANGCHAIN_API_KEY가 설정되어 있지 않습니다.")
    if not langchain_project:
        print("  - LANGCHAIN_PROJECT가 설정되어 있지 않습니다.")
```

- 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.
```

- 사전에 `VS Code` 터미널에 설치할 것

```
pip install -qU faiss-cpu tiktoken

# 그래프를 그리기 위한 라이브러리 설치
pip install -qU grandalf
```

✦

```
from langchain_community.vectorstores import FAISS
from langchain_core.output_parsers import StrOutputParser
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.runnables import RunnablePassthrough, RunnableLambda
from langchain_google_genai import ChatGoogleGenerativeAI
from langchain_huggingface import HuggingFaceEmbeddings

from dotenv import load_dotenv
import os
import warnings                                    # 5.4s
```

```
# 임베딩(Embedding) 생성

from langchain_huggingface import HuggingFaceEmbeddings
import warnings                                                    # 경고 무시

warnings.filterwarnings("ignore")                                  # HuggingFace Embeddings 사용

embeddings = HuggingFaceEmbeddings(
    model_name="sentence-transformers/all-MiniLM-L6-v2",
    model_kwargs={'device': 'cpu'},    \
    encode_kwargs={'normalize_embeddings': True})

print("✅ hugging-face 임베딩 모델 로딩 완료!")
```

- ✅ hugging-face 임베딩 모델 로딩 완료! (`6.8s`)

```
# API 키 확인
if not os.getenv("GOOGLE_API_KEY"):
    os.environ["GOOGLE_API_KEY"] = input("Enter your Google API key: ")

# LLM 초기화
gemini_lc = ChatGoogleGenerativeAI(
    model="gemini-2.5-flash-lite",
    temperature=0,                                          # temperature = 0으로 설정
    max_output_tokens=4096,
)
```

- `gemini-2.5-flash-lite`로 `LLM` 생성하기

```
E0000 00:00:1759826376.951511 2658484 alts_credentials.cc:93] ALTS creds ignored. Not running on GCP and untruste
```

```
# 텍스트로부터 FAISS 벡터 저장소를 생성함

vectorstore = FAISS.from_texts(
    ["Alice is an AI engineer who loves programming!"],
    embedding=embeddings
)                                           # 0.1s
```

```
# 벡터 저장소를 검색기로 사용하기

retriever = vectorstore.as_retriever()
```

```
# 템플릿 정의하기

template = """Answer the question based only on the following context:
{context}

Question: {question}"""
```

```
# 템플릿으로부터 채팅 프롬프트 생성하기

prompt = ChatPromptTemplate.from_template(template)
```

```
print(type(prompt))                # <class 'langchain.prompts.chat.ChatPromptTemplate'>
```

```
# LLM 모델 초기화

gemini_lc = ChatGoogleGenerativeAI(
    model="gemini-2.5-flash-lite",
    temperature=0,                                          # temperature = 0으로 설정
    max_output_tokens=4096,
)
```

```
# 검색 체인 구성

chain = (
    {"context": retriever, "question": RunnablePassthrough()}
    | prompt
    | gemini_lc
    | StrOutputParser()
)
```

---

2) `그래프 구성 확인`

- **`chain.get_graph()`** 메서드 → 그래프 구성 확인
  - 체인의 각 노드, 노으 간의 연결을 나타내는 그래프 객체 반환
    - **그래프의 `노드` = 체인의 `각 단계`**
    - **그래프의 `엣지` = 단계 간의 `데이터의 흐름`**

```
# 체인의 그래프에서 노드 가져오기

chain.get_graph().nodes
```

- `chain.get_graph().nodes` (`0.0s`)

```
{'3a47829b6af64089ae0efacf91792810': Node(id='3a47829b6af64089ae0efacf91792810', name='Parallel<context,question>
'6bfd7625e89c4fd9a21a310536d3381c': Node(id='6bfd7625e89c4fd9a21a310536d3381c', name='Parallel<context,question>C
'4a920571229044cdbea3a45f13bd369c': Node(id='4a920571229044cdbea3a45f13bd369c', name='VectorStoreRetriever', data
'ccac6fe387d14ce6943ee0354406e9c9': Node(id='ccac6fe387d14ce6943ee0354406e9c9', name='Passthrough', data=Runnable
'dd89b2c15667467baf81ea6ddbf56299': Node(id='dd89b2c15667467baf81ea6ddbf56299', name='ChatPromptTemplate', data=C
'fdb22a5260b143a19fadc21d59e003f8': Node(id='fdb22a5260b143a19fadc21d59e003f8', name='ChatGoogleGenerativeAI', da
'c9e4cf780d974908a8fbd1caf38795ed': Node(id='c9e4cf780d974908a8fbd1caf38795ed', name='StrOutputParser', data=StrO
'5e11b2ded7ac43f4963333530b9f2d2d': Node(id='5e11b2ded7ac43f4963333530b9f2d2d', name='StrOutputParserOutput', dat
```

```
# 체인의 그래프에서 엣지 가져오기

chain.get_graph().edges
```

- `chain.get_graph().edges` - (`0.0s`)

```
[Edge(source='28d3448e94f64949810466ae33b24bb5', target='6f1d4b98c2c2429fbb175f8c4c31f611', data=None, conditiona
Edge(source='6f1d4b98c2c2429fbb175f8c4c31f611', target='d9c48cd636524d6cba07aa83a3c4a45a', data=None, conditional
Edge(source='28d3448e94f64949810466ae33b24bb5', target='4022fd2fd5de4e04b442f42ec1bd248a', data=None, conditional
Edge(source='4022fd2fd5de4e04b442f42ec1bd248a', target='d9c48cd636524d6cba07aa83a3c4a45a', data=None, conditional
Edge(source='d9c48cd636524d6cba07aa83a3c4a45a', target='22e7826d3a6845a4b42daae54f1db225', data=None, conditional
Edge(source='22e7826d3a6845a4b42daae54f1db225', target='4119523b2bf54e56b1c64001057a0ad7', data=None, conditional
Edge(source='3672bd85a2774a7bb16aaad83f973d1c', target='73e690c3e20c431295d9ae503ca16db5', data=None, conditional
Edge(source='4119523b2bf54e56b1c64001057a0ad7', target='3672bd85a2774a7bb16aaad83f973d1c', data=None, conditional
```

---

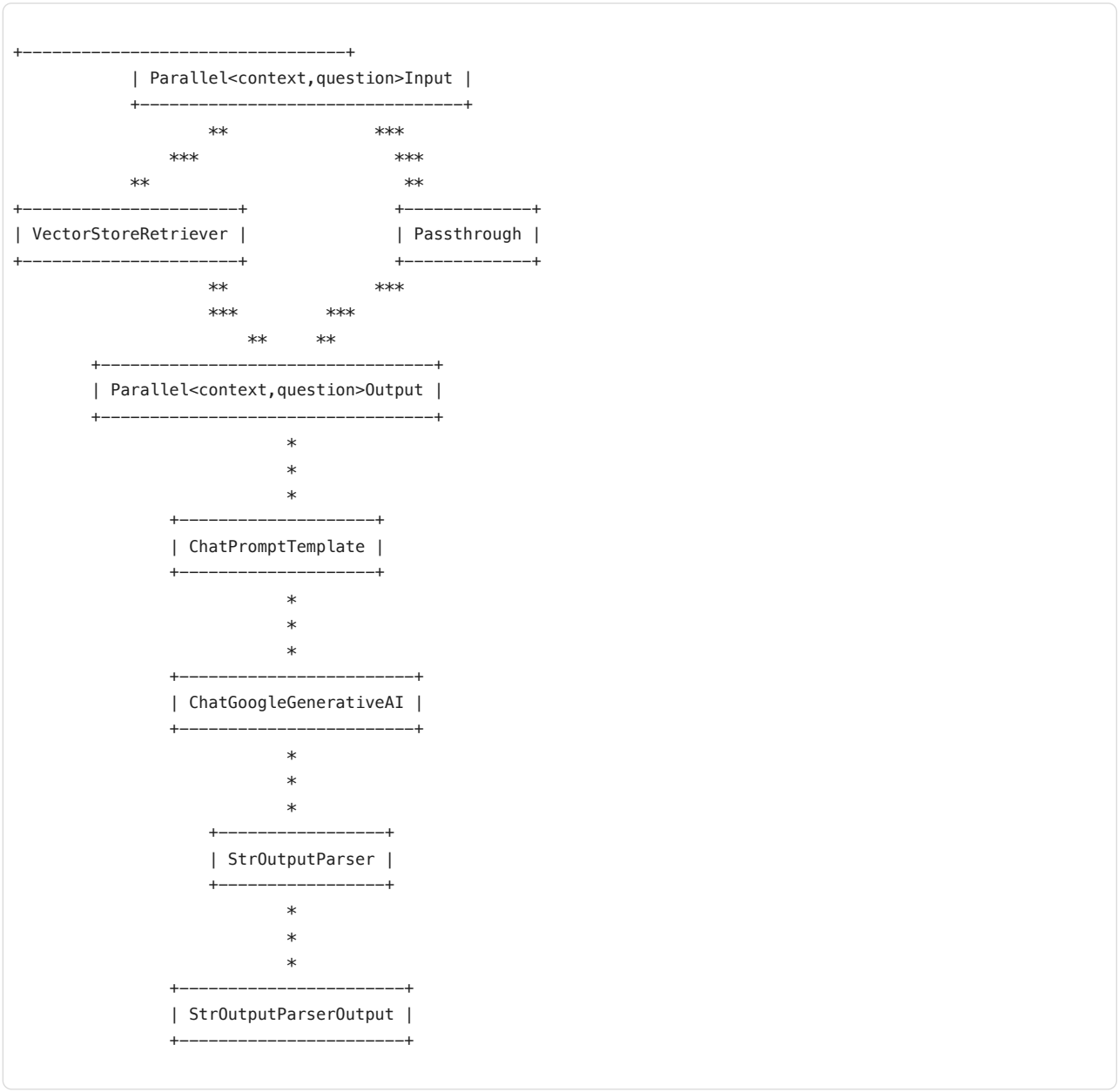
4) `그래프 출력`

- 숫자 → 이해하기 쉬운 형태 = `그래프` 로 확인해보기

```
# 체인의 그래프를 ASCII 형식으로 출력해보기

chain.get_graph().print_ascii()
```

- `chain.get_graph().print_ascii()` - (`0.0s`)



---

5) `프롬프트 가져오기`

- **`chain.get_prompt()`** = 체인에서 사용되는 프롬프트 객체의 **`리스트`** 반환

```
# 체인에서 사용되는 프롬프트 가져오기

chain.get_prompts()
```

- `chain.get_prompts()` - (`0.0s`)

```
[ChatPromptTemplate(input_variables=['context', 'question'], input_types={}, partial_variables={}, messages=[Huma
```

- next: ***`03. RunnableLambda`***
