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

---

7. `@chain` 데코레이터로 `Runnable` 구성

1) `@chain` 데코레이터를 사용하여 `Runnable` 객체 생성

- **`@chain` 데코레이터** 추가 → 임의의 함수 = 체인 변환
  - *`ㄴ` `함수`를 `RunnableLambda`로 `래핑`하는 것과 기능적으로 동일

- **환경 설정**

```
# 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.
```

- **`동적 속성 지정`**
  - ① **`model_name`**: `configurable_fields()` → `model_name`

- *참고:* *`LangChain - gemini 공식 가이드`*

```
from langchain_core.output_parsers import StrOutputParser
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.runnables import chain
```

```
from langchain_google_genai import ChatGoogleGenerativeAI

from dotenv import load_dotenv
import os

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

# LLM 생성하기
gemini_lc = ChatGoogleGenerativeAI(
    model="gemini-2.5-flash-lite",
    temperature=0,
)
```

- 기본 `LLM` 생성하기 (`gemini_lc`) - `gemini-2.5.flash-lite`

```
E0000 00:00:1760004674.190538 2209967 alts_credentials.cc:93] ALTS creds ignored. Not running on GCP and untruste
```

- **`ChatPromptTemplate` 클래스 → 2개의 `프롬프트 템플릿` 정의하기**
  - `prompt1` = 주어진 주제에 대한 짧은 설명
  - `prompt2` = `emoji`를 활용한 인스타그램 게시글 생성 요청

```
# 프롬프트 템플릿 정의하기
prompt1 = ChatPromptTemplate.from_template("{topic} 에 대해 짧게 한글로 설명해주세요.")
prompt2 = ChatPromptTemplate.from_template(
    "{sentence} 를 emoji를 활용한 인스타그램 게시글로 만들어주세요."
)
```

- **`custom_chain` 함수** = 입력 텍스트를 기반으로 사용자 정의 체인 실행하기
  - **`@chain` 데코레이터 → `사용자 정의 함수 데코레이팅` → `함수 = Runnable 객체`**

```
@chain
def custom_chain(text):
    # chain1 = 첫 번째 프롬프트, gemini 모델, 문자열 출력 파서 연결
    chain1 = prompt1 | gemini_lc | StrOutputParser()
    output1 = chain1.invoke({"topic": text})

    # chain2 = 두 번째 프롬프트, gemini 모델, 문자열 출력 파서 연결
    chain2 = prompt2 | gemini_lc | StrOutputParser()
    # 두 번째 체인 호출 → 파싱된 첫 번째 결과 전달 → 최종 결과 반환함
    return chain2.invoke({"sentence": output1})
```

- **`custom_chain`** = `runnable` *(실행 가능한 객체)* → **`invoke()`로 실행하기**
  - `LangSmith` 추적 링크
  - `LangSmith` 추적 기록: **`custom_chain`** 추적 정보 ⭕ → `gemini 모델 호출 중첩`

```
# custom_chain 호출해보기

print(custom_chain.invoke("양자역학"))
```

- **`custom_chain`** 생성 결과 - (`4.8s`)

```
✨ 양자역학: 아주 작은 세계의 신비로운 규칙 ⚛️

우리가 사는 거시적인 세계와는 전혀 다른, 놀랍고 신비로운 규칙들이 숨 쉬는 곳이 있습니다. 바로 **원자나 전자처럼 아주 작은 입자들의 세계**인데요, 이

🤔 **양자역학의 핵심 아이디어는 무엇일까요?**

*   **양자화 (Quantization) 🧱:** 에너지, 운동량 같은 물리량이 연속적이지 않고 **딱딱 끊어지는 덩어리(양자)**로 존재해요. 마치 계단을 오르
*   **파동-입자 이중성 (Wave-Particle Duality) 🌊✨:** 빛이나 전자 같은 입자들이 때로는 **파동처럼 부드럽게 퍼져나가고**, 때로는 **단단
*   **불확정성 원리 (Uncertainty Principle) ❓➡️💨:** 입자의 **위치와 운동량을 동시에 정확하게 알 수는 없어요.** 하나를 더 정확히 보려
*   **확률론적 해석 (Probabilistic Interpretation) 🎲:** 입자가 어떤 상태에 있을지 **정확히 예측하는 대신, 그럴 확률**만을 계산할 수

💡 **왜 양자역학이 중요할까요?**
```

양자역학은 현대 과학 기술의 **심장**과 같아요! ❤️

*   우리가 매일 사용하는 **스마트폰, 컴퓨터의 반도체, 레이저, 트랜지스터** 📱💻💡 모두 양자역학 덕분에 작동합니다.
*   **원자력 발전 ⚛️, MRI 🏥** 같은 첨단 기술에도 필수적이에요.
*   미래를 바꿀 **양자 컴퓨터 💻✨, 양자 통신 📡**의 핵심 이론이기도 합니다!

간단히 말해, 양자역학은 **아주 작은 세계의 독특한 규칙을 이해하고, 이를 바탕으로 놀라운 기술을 만들어내는 학문**입니다. 🔬➡️💡

#양자역학 #물리학 #과학 #작은세계 #신비로운규칙 #양자화 #파동입자이중성 #불확정성원리 #미래기술 #과학스타그램 #공부스타그램 #흥미로운과학

```
# custom_chain 호출해보기_2

print(custom_chain.invoke("RAG"))
```

- **`custom_chain`** 생성 결과_2 - (`5.1s`)

✨ **AI 똑똑하게 만드는 비법 공개! 🚀 RAG (검색 증강 생성) 🤖**

질문만 하면 척척 답하는 AI, 어떻게 가능할까요? 🤔 바로 **RAG (Retrieval-Augmented Generation)** 덕분이에요! 💡

쉽게 말해, AI가 질문을 받으면 바로 답을 지어내는 게 아니라, 마치 우리가 검색 엔진으로 정보를 찾아보고 답하는 것처럼, **먼저 관련 정보를 쏙쏙 찾아

RAG의 핵심은 이거예요! 👇

1. **검색 🔍:** 질문과 딱 맞는 정보를 외부 데이터베이스나 문서에서 찾아내요.
2. **증강 ➕:** 찾은 정보를 AI 모델에게 더 많은 맥락을 이해하도록 쏙쏙 넣어줘요.
3. **생성 ✍️:** 풍부해진 정보로 더 정확하고 유용한 답변을 짠! 하고 만들어내죠.

RAG를 쓰면 뭐가 좋을까요? 👍

*   **정확도 UP! 💯:** 최신 정보나 전문 지식으로 잘못된 답변은 이제 그만!
*   **환각 현상 DOWN! 🚫:** AI가 없는 말을 지어내는 일은 줄어들어요.
*   **투명성 UP! 🧐:** 어떤 정보를 참고했는지 알 수 있어 믿음직스러워요.
*   **최신 정보 OK! 💥:** 학습 데이터에 없던 정보도 실시간으로 활용 가능!

이런 곳에서 활용돼요! 🤩

*   챗봇이 제품 질문에 정확한 답변을! 🛍️
*   질문에 대한 답변과 함께 출처도 짠! 📚
*   전문 분야 질문에 심도 있는 답변을! 🔬

RAG는 똑똑한 LLM(거대 언어 모델)의 성능을 한 단계 업그레이드하는 마법 같은 기술이랍니다! ✨

#RAG #검색증강생성 #인공지능 #AI #LLM #기술 #똑똑한AI #정보검색 #챗봇 #미래기술

```
# custom_chain 호출해보기_3

print(custom_chain.invoke("Runnable 객체"))
```

- **`custom_chain`** 실행 결과_3 (`4.4s`)

🚀 자바에서 "무엇을 할 것인가"를 정의하는 방법: Runnable 객체! 🚀

안녕하세요 개발자 여러분! 👋 오늘은 자바에서 **스레드가 어떤 일을 해야 할지** 정의하는 아주 중요한 개념, **Runnable 객체**에 대해 알아볼 거예

Runnable은 마치 **요리 레시피**와 같아요! 🍳
`run()`이라는 단 하나의 메소드 안에 **"이 스레드가 해야 할 모든 일"**을 꼼꼼하게 적어두는 거죠. 📝

핵심 포인트! ✨

*   **작업 정의 🎯:** Runnable은 실제 **수행할 코드**를 담고 있어요.
*   **`run()` 메소드 🏃:** 이 안에 모든 실행 로직이 쏙!
*   **스레드와 짝꿍 🤝:** Runnable 자체로는 혼자 실행되지 않아요. 주로 **`Thread` 객체**와 함께 사용되어, 새로운 스레드에서 `run()` 메

간단히 말해,

Runnable은 **"이 스레드가 해야 할 일 목록"**이라고 생각하면 이해하기 쉬워요! ✅

예시로 살펴볼까요? 👇

```java
// Runnable 인터페이스를 구현하는 클래스
class MyRunnable implements Runnable {
    @Override
    public void run() {
        System.out.println("✨ 새로운 스레드에서 실행되는 작업입니다! ✨");
    }
}

// 메인 메소드
public class Main {
    public static void main(String[] args) {
        MyRunnable myTask = new MyRunnable(); // Runnable 객체 생성! 🤓
        Thread thread = new Thread(myTask); // Thread 객체에 Runnable 전달! 🎁
        thread.start(); // 스레드 시작! (run() 메소드 실행!) 🚀
    }
}
```

위 예시에서 `MyRunnable`은 `run()` 메소드 안에 "새로운 스레드에서 실행되는 작업입니다!"라는 메시지를 출력하는 작업을 정의했어요. 그리고 이 `M

정리하자면,

Runnable 객체는 자바에서 **스레드가 수행할 작업을 정의하는 인터페이스**이며, `run()` 메소드 안에 실제 실행 코드를 작성합니다. 💻

#자바 #프로그래밍 #개발자 #스레드 #Runnable #멀티스레딩 #코딩 #개발상식 #프로그래밍팁 #Java #Programming #Developer #Thread #Multi

---

- next: ***08. RunnableWithMessageHistory***
