# Langchain

一个用于开发由大型语言模型 (LLMs) 驱动的应用程序的框架。

## PromptTemplate

用法

```python
from langchain.prompts import PromptTemplate

# 定义一个简单的模板
template = "请用简明的语言介绍一下{topic}。"

# 创建 PromptTemplate 对象
prompt_template = PromptTemplate(
    input_variables=["topic"],  # 模板中使用的变量
    template=template  # 模板字符串
)
```

## LLMChain

当你使用 LLMChain 时，chain.invoke() 返回的是字典（dict）

## StructuredOutputParser

StructuredOutputParser 是 LangChain 框架中用于将大语言模型（LLM）的非结构化文本输出转换为结构化数据（如 JSON、XML、自定义对象等）的核心组件。

## BaseChatOpenAI

专门用于对接 OpenAI 聊天模型的基类，它定义了与 OpenAI 聊天模型交互的核心逻辑。

作用（实现对大模型的封装）

- 统一接口规范
- 封装 API 调用细节
- 参数标准化

## 模型链结构

### 大语言模型链条

基于LLMChain将提示模版和大模型进行组合，接收输入并拼接prompt，然后调用大模型回答，属于基本链。

输入 -》 链 =》 输出

### 简单顺序链

SimpleSequentialChain 是 LangChain 中最基础的顺序链类型，专为单输入单输出的多步骤任务设计。其核心逻辑是将多个链（Chains）按线性顺序串联，前一个链的输出自动成为下一个链的输入，形成一条流水线式处理流程

SimpleSequentialChain

- 单数据流传递
- 顺序执行
- 调试友好性

### 顺序链

SequentialChain

**二、与 SimpleSequentialChain 的区别**

| 特性 | SimpleSequentialChain | SequentialChain |
| --- | --- | --- |
| 输入/输出数量 | 单输入单输出 | 多输入多输出 |
| 数据流复杂度 | 线性传递 | 支持分支、合并与条件判断 |
| 适用场景 | 简单文本生成流水线 | 多步骤任务（如数据分析、多模态处理） |
| 代码复杂度 | 低（无需变量映射） | 中高（需显式定义输入/输出关系） |

### 路由链

子主题 1

## 记忆机制

### ConversationBufferMemory

存储所有对话历史，每次将完整记录附加到当前输入的提示词中

```python
from langchain.memory import ConversationBufferMemory
from langchain.chains import ConversationChain

memory = ConversationBufferMemory()  # 默认存储全部历史
conversation = ConversationChain(llm=llm, memory=memory)
```

memory.clear()：清除记忆

### ConversationBufferWindowMemory

通过参数k控制最近保留的对话数量

```python
from langchain.memory import ConversationBufferWindowMemory

# 设置窗口大小为 2（保留最近 2 轮对话）
memory = ConversationBufferWindowMemory(k=2)
```

### ConversationTokenBufferMemory

### ConversationSummaryBufferMemory