Cơ bản về GraphBuilder

Trong LangGraph, bạn sẽ sử dụng một **GraphBuilder** để định nghĩa các bước (nodes), luồng xử lý (edges), điều kiện phân nhánh (routers), và điểm bắt đầu/kết thúc của một ứng dụng dựa trên LLM.

```
from langgraph.graph import StateGraph


builder = StateGraph(StateType)
```

Ở đây `StateType` là schema mô tả state (thường là TypedDict hoặc pydantic model).

Ví dụ

```
from typing import TypedDict, Literal


class DrinkState(TypedDict):
    preference: Literal["coffee", "tea", "unknown"]
```

1. builder.add_node(name, node_callable)

Thêm một bước xử lý (node) vào graph.

Cú pháp:

```
builder.add_node("tên_node", hàm_xử_lý)
```

Ví dụ

```
def ask_name(state):
    state["name"] = input("Tên bạn là gì? ")
    return state


builder.add_node("ask_name", ask_name)
```

2. builder.add_edge(from_node, to_node)

Tạo liên kết (đường đi) giữa hai node.

Cú pháp:

```
builder.add_edge("node_nguồn", "node_đích")
```

Ví dụ:

```
builder.add_edge("ask_name", "ask_preference")
```

3. builder.add_conditional_edges(from_router_node, conditions_dict)

from_router_node: là một hàm xử lý

Tạo đường đi có điều kiện (routing) từ một node tới nhiều node tùy thuộc vào kết quả của node đó.

Cú pháp:

```
builder.add_conditional_edges("router_node", {
    "kết_quả_1": "node_1",
    "kết_quả_2": "node_2",
    ...
})
```

Ví dụ:

```
builder.add_conditional_edges("router", {
    "coffee_node": "coffee_node",
    "tea_node": "tea_node",
    "fallback_node": "fallback_node"
})
```

=> Node `"router"` sẽ trả về một chuỗi (ví dụ `"coffee_node"`), và flow sẽ đi đến node tương ứng.

4. builder.set_entry_point(node_name)

Xác định điểm bắt đầu của graph.

Ví dụ

```
builder.set_entry_point("start")
```

5. builder.set_finish_point(node_name)

Đánh dấu node kết thúc, tức là sau node này thì flow dừng lại.

Có thể có nhiều node kết thúc!

Ví dụ

```
builder.set_finish_point("coffee_node")
builder.set_finish_point("tea_node")
```

6. builder.compile()

Biên dịch graph của bạn thành một `CompiledGraph` có thể chạy được.

Ví dụ

```
app = builder.compile()
```

Sau đó

```
app.invoke({"preference": "unknown"}
```

7. Tổng kết

Hàm	Mô tả
`add_node(name, node_fn)`	Thêm một bước xử lý đơn vào graph
`add_edge(from_node, to_node)`	Tạo liên kết tuyến tính giữa 2 node
`add_conditional_edges(name, mapping)`	Tạo node phân nhánh có điều kiện
`set_entry_point(name)`	Xác định điểm bắt đầu của flow
`set_finish_point(name)`	Xác định điểm kết thúc
`add_branch(name, sub_graph)`	Thêm một graph phụ (dạng phân nhánh) vào một node
`add_recursion(name, recursive_graph)`	Cho phép gọi đệ quy một graph con
`compile()`	Biên dịch toàn bộ graph thành một ứng dụng chạy được
`add_generator_node(name, generator_node)`	Thêm một node sử dụng generator (yield từng bước)
`add_parallel_nodes(nodes: dict)`	Chạy nhiều node song song trong một bước
`add_conditional_edges_with_default(name, conditions, default)`	Giống `add_conditional_edges` nhưng có nhánh mặc định
`add_stateful_node(name, node_fn)`	Dùng khi node cần giữ trạng thái riêng biệt
`add_async_node(name, async_node)`	Thêm node xử lý bất đồng bộ (`async def`)

Tác giả: Đỗ Ngọc Tú

Công Ty Phần Mềm VHTSoft

Phiên bản #4
Được tạo 18 tháng 4 2025 08:08:23 bởi Đỗ Ngọc Tú
Được cập nhật 29 tháng 4 2025 18:32:49 bởi Đỗ Ngọc Tú