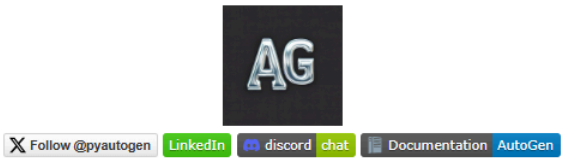
AutoGen入门——快速实现多角色、多用户、多智能体对话系统

1.前言



AutoGen

AutoGen is a framework for creating multi-agent AI applications that can act autonomously or work alongside humans.

如https://github.com/microsoft/autogen所述，autogen是一 多智能体 的框架，属于微软旗下的产品。

依靠AutoGen我们可以快速构建出一个多智能体应用，以满足我们各种业务场景。

本文将以几个示例场景，使用AutoGen快速构建出多智能体应用，一起体验下它的具体用法。

2.环境说明

用到的工具如下：

- python，3.11
- AutoGen，0.4.2
- chainlit，2.0.2
- 大模型，deepseek

安装以下依赖

> pip install -U “autogen-agentchat” “autogen-ext[openai]”

autogen的版本为0.4.2

用到的UI交互界面为Chainlit，安装chainlit命令为：

> pip install chainlit

3.示例一，单智能体

应用场景：挂号**导诊台机器人**，输入症状描述和需求，输出应该挂号的科室。

代码如下：

```
import chainlit as cl
from autogen_agentchat.agents import AssistantAgent
from autogen_agentchat.teams import RoundRobinGroupChat
from autogen_ext.models.openai import OpenAIChatCompletionClient

@cl.on_chat_start
async def main():
    await cl.Message(content="您好，这里是超级无敌大医院，有什么可以帮您？").send()

async def run_team(query: str):
    model_client = OpenAIChatCompletionClient(model="deepseek-chat", base_url="https://api.deepseek.com",
                                              api_key="PEPLACE-YOUR-API-KEY", model_info={
            "vision": False,
            "function_calling": False,
            "json_output": True,
            "family": "unknown",

```

```
        }, )
    assistant_agent = AssistantAgent("assistant", model_client=model_client,
                                     system_message="你是一所口腔医院的导诊台机器人，负责解答用户的挂号问题，用户描述症状需求，你回答应该挂的科室。"
                                                    "在本医院中有以下科室：牙体牙髓科、口腔修复科、口腔外科、口腔种植科、儿童口腔专科。"
                                                    "如果用户的问题与挂号咨询不符合，回答：“您的描述与症状无关，暂不支持”")
    team = RoundRobinGroupChat(participants=[assistant_agent], max_turns=1)
    response_stream = team.run_stream(task=query)
    async for msg in response_stream:
        if hasattr(msg, "source") and msg.source != "user" and hasattr(msg, "content"):
            msg = cl.Message(content=msg.content, author="Agent Team")
            await msg.send()

@cl.on_message
async def main(message: cl.Message):
    await run_team(message.content)
```

AI写代码

运行：

```
chainlit run .\nurses_station_ai.py -w
```

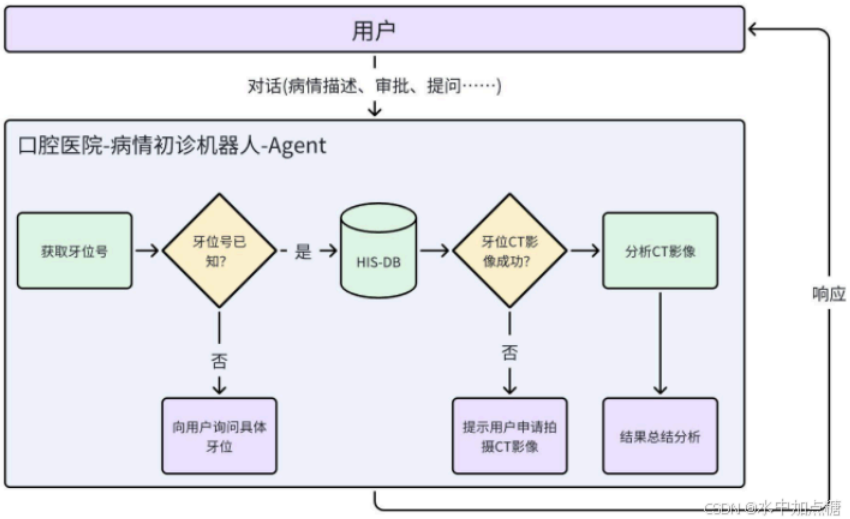
运行效果：

4.示例二，智能体与FunctionCall

示例应用场景：**病情初诊机器人**，输入症状描述，系统**根据症状**查询此患者的**相关历史资料**，如无则建议先进行资料预备，并推送至**人工团队**进行**流程审核**

4.1 示例流程

模拟流程图如下：



病情初诊 机器人　Agent**根据用户**描述**的症状**和对应的**牙位号**获取出对应的CT影像信息，如**信息不足**则**主动**向用户**发起询问**，或**询问用户**是否需要**发起审批。**如信息已具备，则根据CT影像信息进行自动诊断，并将分析出结果展示给用户。

4.2 示例代码与演示

编写python文件： `endodontics_dentistry_ai.py` 。代码如下：

```python
import chainlit as cl
from autogen_agentchat.agents import AssistantAgent
from autogen_agentchat.teams import RoundRobinGroupChat
from autogen_ext.models.openai import OpenAIChatCompletionClient

@cl.password_auth_callback
def auth_callback(username: str, password: str):
    if (username, password) == ("admin", "admin"):
        return cl.User(
            identifier="admin", metadata={"role": "admin", "provider": "credentials"}
        )
    elif (username, password) == ("puhaiyang", "123456"):
        return cl.User(
            identifier="puhaiyang", metadata={"role": "admin", "provider": "credentials"}
        )
    else:
        return None

@cl.on_chat_start
async def main():
    await cl.Message(content="您好，这里是牙体牙髓科，您牙齿哪里不适？").send()

async def x_p_search(tooth_position: str) -> str:
    """Find information on the web"""
    app_user = cl.user_session.get("user")
    print(f"模拟查询{app_user.identifier}的{tooth_position}牙片数据")
    if tooth_position == "46":
        return "牙根尖处有阴影，疑似感染，需要进一步分析诊断"
    else:
        return f"{tooth_position}无影像"

async def run_team(query: str):
    model_client = OpenAIChatCompletionClient(model="deepseek-chat", base_url="https://api.deepseek.com",
                                              api_key="PEPLACE-YOUR-API-KEY", model_info={
            "vision": False,
            "function_calling": True,
            "json_output": True,
            "family": "unknown",
        }, )
    assistant_agent = AssistantAgent("assistant", model_client=model_client, tools=[x_p_search],
                                     system_message="你是一个牙体牙髓科的病情诊断机器人，负责对用户输入的症状描述分析原因，在分析病因前先询问出用户是具体哪
                                                    "在知道了具体的牙位号后，再调用x_p_search工具进行问题回答，传入给x_p_search工具的参数需要自动转为牙位
                                                    "如果用户的问题与病情咨询无关，回答：“您的描述与症状无关，暂不支持”")
    team = RoundRobinGroupChat(participants=[assistant_agent], max_turns=1)
    response_stream = team.run_stream(task=query)
    async for msg in response_stream:
        if hasattr(msg, "source") and (msg.type == "ToolCallExecutionEvent" or msg.type == "ToolCallRequestEvent"):
            # functionCall事件消息不显示给用户
            continue
        if hasattr(msg, "source") and msg.source != "user" and hasattr(msg, "content"):
            if msg.content.endswith("无影像"):
```

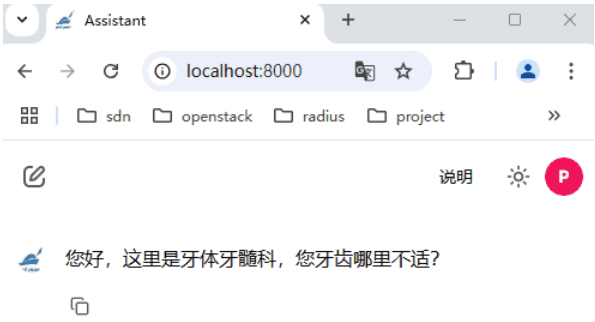
```
                res = await cl.AskActionMessage(
                    content=f"{msg.content}，是否需要帮您申请拍摄此牙的CT影像？",
                    actions=[
                        cl.Action(name="continue", payload={"value": "申请"}, label="✅ 申请牙片"),
                        cl.Action(name="cancel", payload={"value": "取消"}, label="❌ 取消"),
                    ],
                ).send()

                if res and res.get("payload").get("value") == "申请":
                    await cl.Message(
                        content="牙片申请已提交!待审核通过后前往第3影像室进行拍摄。",
                    ).send()
            else:
                msg = cl.Message(content=msg.content, author="Agent Team")
                await msg.send()

@cl.on_message
async def main(message: cl.Message):
    await run_team(message.content)

```

AI写代码

运行：

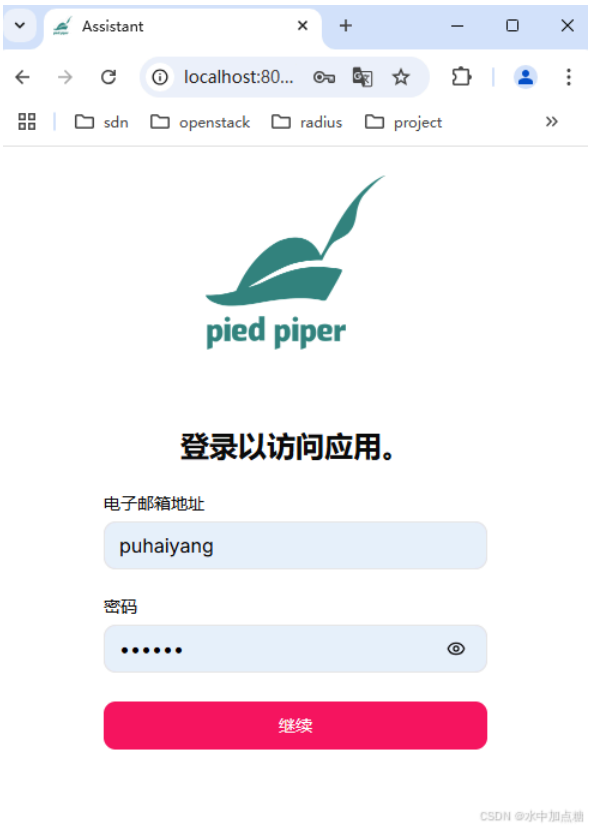
> chainlit run .\endodontics_dentistry_ai.py -w

运行效果



4.3 chainlit认证配置

上面的示例代码中使用到了chainlit中的认证(Authentication)功能，并在代码中模拟了两个用户。
当首次访问智能体应用的界面时会弹出登录界面，以让我们先输入登录信息后才能使用此应用。



首次运行chainlit需要生成密钥

> chainlit create-secret

之后在项目根目录创建 **.env** 文件，填入前面生成的secret信息，如：

```
CHAINLIT_AUTH_SECRET="WaElB8_~5Bif=~Yz,-y0d01~J-r$P_hoj3ihfCr_c2qwtv?J@>.7tEF.Tb9CE$*A"
```

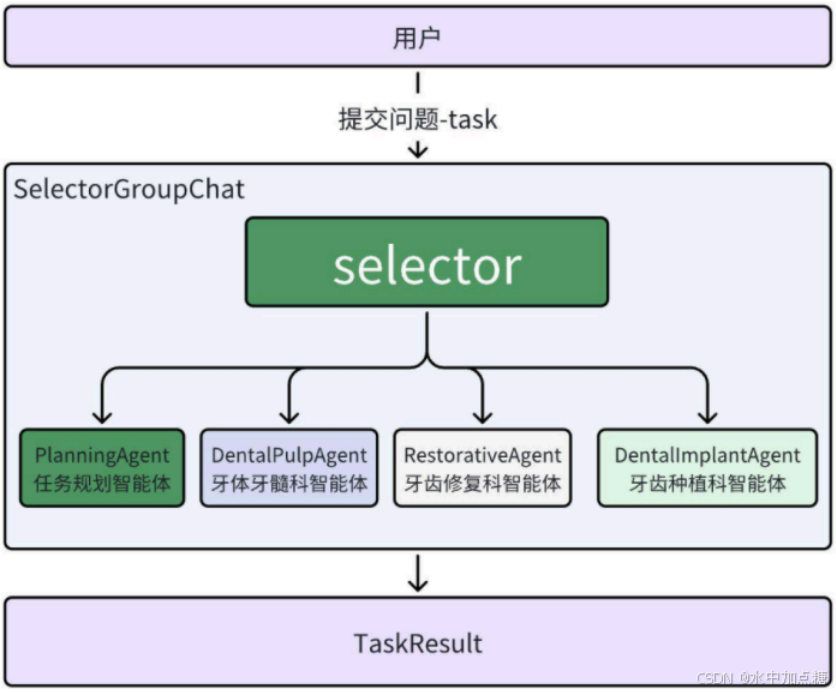
AI写代码

5. 示例三，多智能体自动选择

此功能涉及到的技术点为Selector Group Chat。

示例应用场景：用户**同一时间**向**多个不同科室**的专家医生**咨询问题**，每轮向用户解答的医生都是**问题相关度最高**的科室医生

5.1 示例流程



如：牙体牙髓科AI、修复科AI、种植牙AI形成一个专家团队，用户向这个专家团队提问，专家团队每次派一个**最专业**的代表来解答问题。

5.2 示例代码与演示

编写python文件： `dentistry_selector_ai.py` 。代码如下：

```
import chainlit as cl

from autogen_agentchat.agents import AssistantAgent
from autogen_agentchat.conditions import MaxMessageTermination, TextMentionTermination
from autogen_agentchat.teams import SelectorGroupChat
from autogen_ext.models.openai import OpenAIChatCompletionClient

model_client = OpenAIChatCompletionClient(model="deepseek-chat", base_url="https://api.deepseek.com",
                                          api_key="PEPLACE-YOUR-API-KEY", model_info={
        "vision": False,
        "function_calling": True,
        "json_output": True,
        "family": "unknown",
    }, )

planning_agent = AssistantAgent("PlanningAgent",
                                description="用于规划的Agent，当一个任务到达时此Agent是第一个参与者",
                                model_client=model_client,
                                system_message="""
                                你是一个任务规划智能体。
                                你的工作是将复杂的任务分解为更小的、可管理的子任务。
                                你的团队成员有3个，分别是：
                                    DentalPulpAgent：牙体牙髓科智能体
                                    RestorativeAgent：牙齿修复科智能体
                                    DentalImplantAgent：牙齿种植科智能体

                                你只计划和委派任务，而不自己执行它们

                                分配任务时，请使用此格式：
                                1. <agent> : <task>

                                当所有智能体把任务完成后，再总结结果以"TERMINATE"结束。
                                """
                                )

dental_pulp_agent = AssistantAgent("DentalPulpAgent",
                                   description="牙体牙髓科智能体",
                                   model_client=model_client,
                                   system_message="""
                                你是一个口腔医院的牙体牙髓科智能体。
                                你可以解答关于牙体牙髓科中患者提出的问题，你的解答非常专业，且可靠。
                                """
                                   )

restorative_agent = AssistantAgent("RestorativeAgent",

```

```
                                    description="牙齿修复科智能体",
                                    model_client=model_client,
                                    system_message="""
                                你是一个口腔医院的牙齿修复科智能体。
                                你可以解答关于牙齿修复中患者提出的问题，比如牙冠、烤瓷牙、嵌体修复等。你的解答非常专业，且可靠。
                                """
                                    )

dental_implant_agent = AssistantAgent("DentalImplantAgent",
                                      description="牙齿种植科智能体",
                                      model_client=model_client,
                                      system_message="""
                                你是一个口腔医院的牙齿种植科的智能体。
                                你可以解答关于牙齿种植科中患者提出的问题，你的解答非常专业，且可靠。
                                """
                                      )

@cl.on_chat_start
async def main():
    await cl.Message(content="您好，这里是口腔医院专家团队，有什么可以帮您？").send()

async def run_team(query: str):
    text_mention_termination = TextMentionTermination("TERMINATE")
    max_messages_termination = MaxMessageTermination(max_messages=25)
    termination = text_mention_termination | max_messages_termination

    team = SelectorGroupChat(
        [planning_agent, dental_pulp_agent, restorative_agent, dental_implant_agent],
        model_client=model_client,
        termination_condition=termination,
    )

    response_stream = team.run_stream(task=query)
    async for msg in response_stream:
        if hasattr(msg, "source") and msg.source != "user" and hasattr(msg, "content"):
            msg = cl.Message(content=msg.content, author=msg.source)
            await msg.send()


@cl.on_message
async def main(message: cl.Message):
    await run_team(message.content)
```

AI写代码

运行：

```
chainlit run .\dentistry_selector_ai.py -w
```

分别提出问题：

```
什么是烤瓷牙?
什么是根管治疗?
```

从上面运行结果可知，当问到**根管治疗**相关问题时，会由**DentalPulpAgent**(牙体牙髓科智能体)来回答问题。
当问到**烤瓷牙**相关问题时，会由**RestorativeAgent**(牙齿修复科智能体)来回答问题。

之所以能做到**自动切换智能体**，其原因为在问题执行前会由**任务规划智能体**(PlanningAgent)进行预规划，由它根据问题的描述分配到对应的智能体。

6.AutoGen Studio工作流UI

前面几个示例都是用**python代码**的方式创建智能体，对于用户有一定的编码要求，且流程处理上也不够直观。

比较好的是AutoGen中也提供了与Dify类似UI界面操作的方式，即：AutoGen Studio。

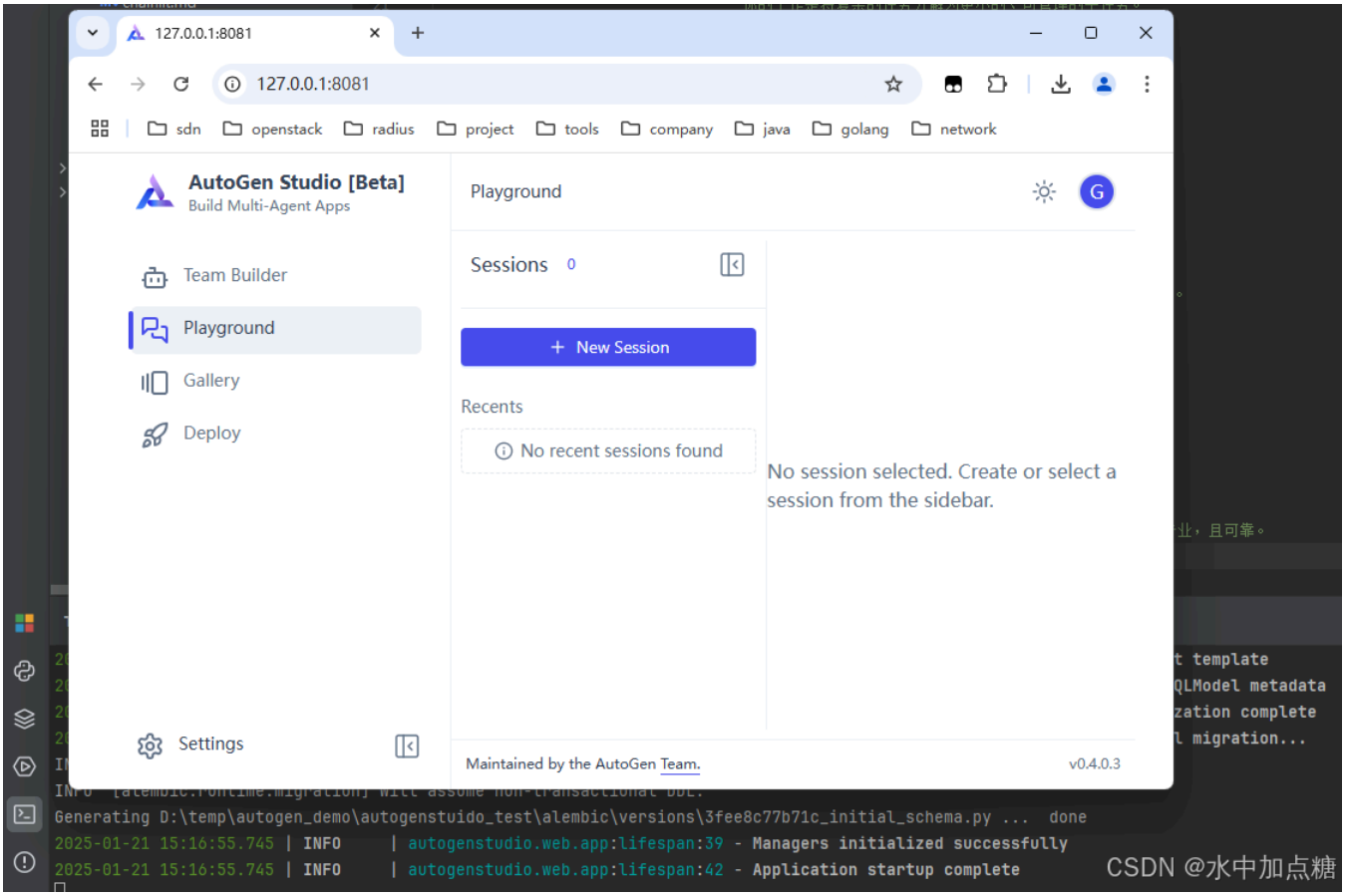
安装时，直接使用如下命令安装：

```
pip install -U autogenstudio
```

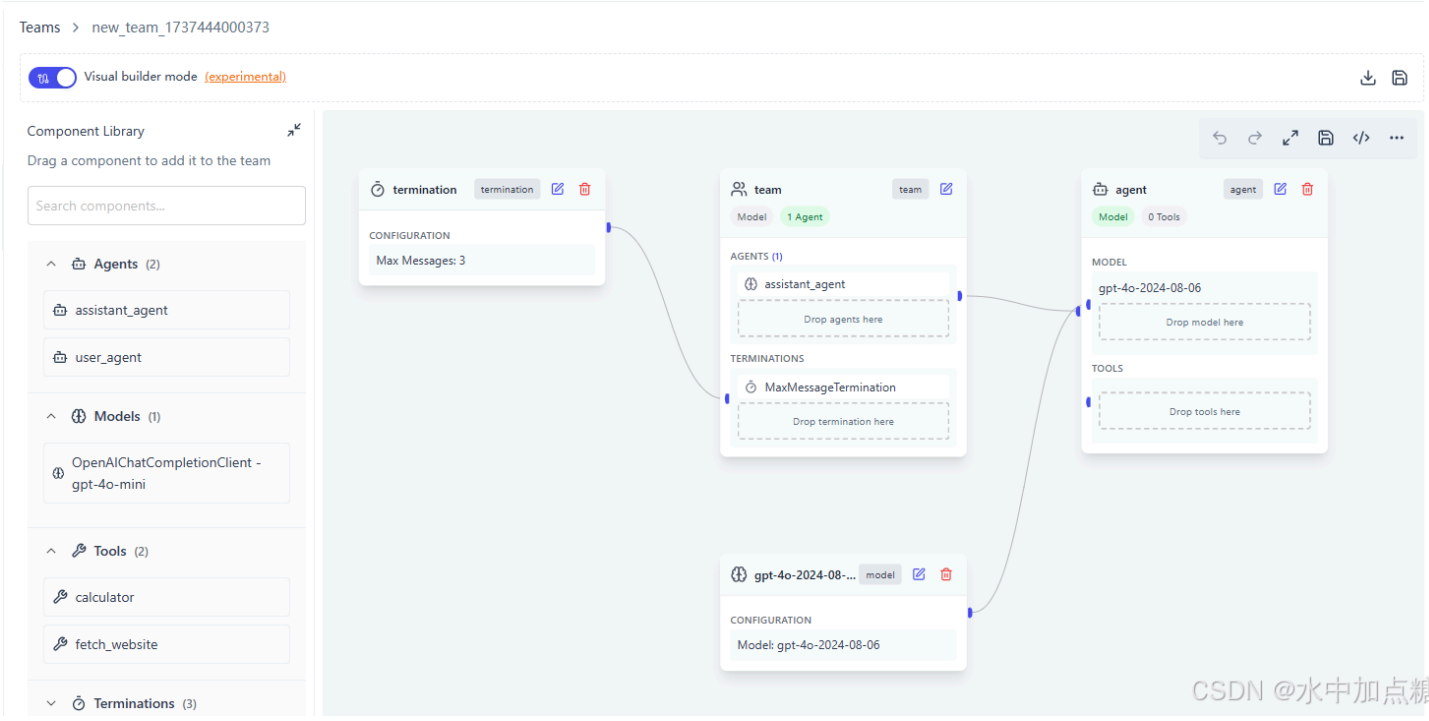
启动时指定**监听端口**与**运行目录**，如：

```
autogenstudio ui --port 8081 --appdir autogenstuido_test
```

之后访问 http://127.0.0.1:8081/ ，即可进入**AutoGen Studio**界面



使用时可以看到**AutoGen Studio**的一些功能上还有**experimental**标签，且功能支持度目前还不是很多



但从支持**流程与节点编辑**这些**功能点**来看还是非常棒的，待后续更新后再继续体验