ChatGPT插件开发：TODO待办事项案例实战

> 版本号：v1.0 获取最新版本
> @TechDIYLife： YouTube主页，Github主页

【观看视频解说】

本章内容，将为你分享如何在本地开发环境中运行并调试OpenAI官方给出的案例。通过学习官方案例，可以帮助你了解如何开发ChatGPT插件的API服务。通过本章内容，你将了解到：

- 配置API服务开发的本地环境
- 运行，并测试官方的案例

要了解ChatGPT插件开发的基本知识和开发流程，请阅读上一章内容：ChatGPT插件开发（基础知识）。

目录索引

1 概要

【本节视频解说】

OpenAI官方给出了四个插件案例， 涵盖了不同的身份验证方案和用例。从简单的无身份验证待办事项列表插件到更强大的检索插件，通过这些例子我们就可以了解OpenAI希望通过插件实现的功能。 四个插件分别是： - 简单的无身份验证待办事项列表插件 - 带服务级身份验证的简单待办事项列表插件 - 简单的体育统计插件 - 语义搜索和检索插件。
这些插件，可以在本地计算机上或通过云开发环境（如GitHub Codespaces、Replit或CodeSandbox）上运行。本章内容，我们通过运行和调试前两个插件，也就是待办事项插件，来学习

如何开发ChatGPT插件。

2 前提知识

阅读本章内容最好拥有以下的相关知识：

- 了解基本Python编程知识
- 阅读上一章《ChatGPT插件开发》了解插件开发的基本知识和开发流程

3 环境安装：Windows系统

【本节视频解说】

1. 安装Python开发环境，推荐基于Anaconda的环境。 下载地址：
https://www.anaconda.com/products/distribution
2. 安装Visual Studio Code（或者其他你熟悉的开发环境，文本编辑器等均可）。
   下载地址：https://code.visualstudio.com/

4 案例1：无身份验证的待办事项插件

【本节视频解说】

OpenAI官方给的第一个案例是一个无身份验证的待办事项插件，此插件支持待办事项的创建，删除，以及读取的基本操作。除此之外代码中还提供了Logo，清单文件和规范文件的访问方式。
下面，我们就来在本地安装的环境中运行此案例。

第一步：安装开发库

首先，需要安装两个相关的开发库quart和quart_cors，步骤如下：

1. 打开终端： 点击 `开始 --> Anaconda prompt`
2. 激活环境：输入命令 `conda activate yourEnv` （如果你的开发环境不是base，需要执行此步来激活环境）
3. 安装quart库： `pip install quart`
4. 安装quart_cors库： `pip install quart_cors`

第二步：创建API代码文件 plugin_todo_v1.py
开发库安装完成以后，打开Visual Studio Code，创建一个新的文件命名为 plugin_todo_v1.py，并将以下代码贴到文件中。并修

改 `app = quart_cors.cors(quart.Quart(__name__), allow_origin="https://chat.openai.com")`
将 `allow_origin` 访问限制部分删除。

```
import json
import quart
import quart_cors
from quart import request
#app = quart_cors.cors(quart.Quart(__name__), allow_origin="https://chat.openai.com")
app = quart_cors.cors(quart.Quart(__name__)) # 删除了只需要来自openai.com的访问限制

_TODOS = {} #定义存储TODO的变量

@app.post("/todos/<string:username>") # 响应POST请求，用来添加用户的TODO
async def add_todo(username): # 添加TODO函数
    request = await quart.request.get_json(force=True)
    if username not in _TODOS:
        _TODOS[username] = []
    _TODOS[username].append(request["todo"])
    return quart.Response(response='OK', status=200)

@app.get("/todos/<string:username>") #响应GET请求，读取用户的TODO数据
async def get_todos(username):
    return quart.Response(response=json.dumps(_TODOS.get(username, [])), status=200)

@app.delete("/todos/<string:username>")
async def delete_todo(username):
    request = await quart.request.get_json(force=True)
    todo_idx = request["todo_idx"]
    # fail silently, it's a simple plugin
    if 0 <= todo_idx < len(_TODOS[username]):
        _TODOS[username].pop(todo_idx)
    return quart.Response(response='OK', status=200)

@app.get("/logo.png") #响应读取logo的请求
async def plugin_logo():
    filename = 'logo.png'
    return await quart.send_file(filename, mimetype='image/png')

@app.get("/.well-known/ai-plugin.json") #响应读取manifest文件的请求
async def plugin_manifest():
    host = request.headers['Host']
    with open("manifest.json") as f:
        text = f.read()
        text = text.replace("PLUGIN_HOSTNAME", f"https://{host}")
        return quart.Response(text, mimetype="text/json")

@app.get("/openapi.yaml")  #响应读取openAPI规范文件的请求
async def openapi_spec():
    host = request.headers['Host']
    with open("openapi.yaml") as f:
        text = f.read()
        text = text.replace("PLUGIN_HOSTNAME", f"https://{host}")
        return quart.Response(text, mimetype="text/yaml")
```

```
def main():
    app.run(debug=True, host="0.0.0.0", port=5002) #启动服务

if __name__ == "__main__":
    main()
```

第三步：创建API测试代码 plugin_todo_v1_test.py

API服务代码准备好以后，开始准备测试代码。新建 plugin_todo_v1_test.py 文件，并将下面的代码拷贝到文件中。下面的代码，分别用来测试添加，删除和读取待办事项。

```
import requests
import json
import sys

# 测试程序，接受一个或两个参数
# 参数1 opt:add, list,  del
# 参数2：   add时，  todo的文本
#           list ,  N/A
#           del  ,  todo_id{0, 1, 2...}

opt=sys.argv[1] #

# 设置请求头
headers = {
    "Content-Type": "application/json"
}

if opt=="add":
    todo=sys.argv[2]
    # 准备数据
    data = {
        "todo": todo
    }
    # 发送POST请求
    response = requests.post("http://127.0.0.1:5002/todos/john", headers=headers, data=json.dumps(data))

if opt=="list":
    # 发送POST请求
    response = requests.get("http://127.0.0.1:5002/todos/john", headers=headers)

if opt=="del":
    _id=int(sys.argv[2])
    data = {
        "todo_idx": _id
    }
    response = requests.delete("http://127.0.0.1:5002/todos/john", headers=headers, data=json.dumps(data))


# 打印响应结果
print(response.status_code)
print(response.content)
print(response)
```

这段 Python 代码是一个使用 requests 库发送 HTTP GET 请求的示例。该请求发送到本地主机上的 http://127.0.0.1:5002/todos/john API 端点。requests.get 方法是用于发送 HTTP GET 请求的函数。它接受两个参数。第一个参数是请求的 URL，这里是 http://127.0.0.1:5002/todos/john。第二个参数是一个可选参数，用于指定请求头部信息，这里是 headers 变量。在发送请求之后，响应数据被存储

在 response 变量中。您可以使用 response 对象访问响应状态码、响应头部和响应体。例如，要获取响应体的 JSON 数据，可以使用 response.json() 方法。

第四步：测试
测试步骤如下：

1. 进入Anaconda prompt终端界面，并激活开发环境（比如：conda activate yourEnvName）
2. 切换到程序所在目录（使用 cd 命令）
3. 启动API服务，执行命令：python plugin_todo_v1.py
4. 测试：重新打开一个新的Anaconda prompt终端，分别输入以下命令，
    - 添加测试：python plugin_todo_v1.py add “todo1”
    - 列表打印：python plugin_todo_v1.py list
    - 删除测试： python plugin_todo_v1.py del 0

5 案例2：服务级身份验证的待办事项插件

【本节视频解说】

案例1中，并没有任何的身份验证。 为了保护数据的安全，通常需要添加身份验证功能。身份验证的方式有多种，OpenAI推荐的有服务级和OAuth两种验证方式。案例2提供了一个服务级身份验证方式。
操作步骤如下：
第一步：API服务端设置
复制 plugin_todo_v1.py 文件并重命名为plugin_todo_v2.py，并添加以下代码：

```
_SERVICE_AUTH_KEY = "yourAuthKey"

# Bearer 是身份验证协议 OAuth 2.0 中的一种类型
def assert_auth_header(req):
    assert req.headers.get(
        "Authorization", None) == f"Bearer {_SERVICE_AUTH_KEY}"
```

以上代码定义了一个 _SERVICE_AUTH_KEY 字符串变量，用来存储身份验证的令牌。然后，代码定义了一个名为 assert_auth_header 的函数，该函数用于验证请求头中是否包含了正确的身份验证令牌。

在函数内部，它检查传入的请求对象 req 的 header 中是否包含了一个名为 "Authorization" 的字段，并将其值与 _SERVICE_AUTH_KEY 拼接成一个 "Bearer" 令牌进行比较。如果两个值不相等，那么该函数将会抛出一个 AssertionError，表示身份验证失败。

为 add_todo，get_todos，delete_todo 函数中添加用来验证身份的代码。

```
assert_auth_header(quart.request)
```

在 add_todo，get_todos 和 delete_todo 函数中，assert_auth_header 函数用于验证请求头部是否包含了正确的身份验证令牌。如果身份验证失败，那么 assert_auth_header 函数将会抛出 AssertionError，并且函数不会继续执行。

通过将 assert_auth_header 函数嵌入到 add_todo，get_todos 和 delete_todo 函数中，可以确保这些函数只有在正确的身份验证令牌被传递时才会执行。这可以提高您的 API 的安全性，并保护其免受未经授权的访问。

第二步：测试代码部分修改

下面需要对测试代码进行相应的修改， 复制 plugin_todo_v1_test.py 为 plugin_todo_v2_test.py，并在文件中添加以下代码：

```
# 设置身份验证密钥
_SERVICE_AUTH_KEY = "yourAuthKey"

# 设置请求头
headers = {
    "Content-Type": "application/json",
    "Authorization": f"Bearer {_SERVICE_AUTH_KEY}"
}
```

和上面的类似，_SERVICE_AUTH_KEY 是身份验证密钥。 headers 是一个字典，用于存储请求头部信息。在这个示例中，它包含了两个键值对。第一个键值对是 "Content-Type"，其值为 "application/json"，指定了请求体的 MIME 类型。第二个键值对是 "Authorization"，其值为 "Bearer" 令牌，它包含了 _SERVICE_AUTH_KEY 的值，并指定了身份验证类型。

第三步：测试
测试步骤如下：

1. 进入Anaconda prompt终端界面，并激活开发环境（比如：conda activate yourEnvName）
2. 切换到程序所在目录（使用 cd 命令）
3. 启动API服务，执行命令：python plugin_todo_v2.py
4. 测试：重新打开一个新的Anaconda prompt终端，分别输入以下命令，
    - 添加测试：python plugin_todo_v2.py add “todo1”
    - 列表打印：python plugin_todo_v2.py list
    - 删除测试： python plugin_todo_v2.py del 0

参考文献

- 官方文档： https://platform.openai.com/docs/plugins/introduction

- Quart开发库： https://pgjones.gitlab.io/quart/
- Quart_cors开发库：https://github.com/pgjones/quart-cors/
- OpenAPI规范格式：https://swagger.io/tools/open-source/getting-started/