

# 腾讯云代码助手 实践教学



腾讯云

## 【 版权声明 】

©2013–2025 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

## 【 商标声明 】



及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

## 【 服务声明 】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

## 【 联系我们 】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或 95716。

# 文档目录

## 实践教程

### 应用开发实践

产品使用小技巧

实战 Python Web API

快速搭建在线相册应用

快速制作企微提醒机器人

自定义知识库实践：智能体框架 Swarm

快速开发一个微信小程序游戏

### 具体功能实践

跨文件能力扩展上下文

代码助手与终端的结合应用实践

单元测试应用实践

代码诊断应用实践

### 企业管理实践

构建企业专属知识库

# 实践教程

## 应用开发实践

### 产品使用小技巧

最近更新时间：2025-04-14 09:54:23

腾讯云代码助手自上线以来，已为超数万名开发者提供智能编程服务。在过去的一年里，通过开发者的反馈和自身实践，我们总结了一些使用腾讯云代码助手提升生产力的小技巧。与学习新框架或库类似，掌握如何高效利用 AI 工具也可以显著提高我们的工作效率。在《产品使用小技巧》系列中，我们将持续更新，分享日常使用腾讯云代码助手的技巧和实践，希望这些建议能够帮助开发者更高效地使用腾讯云代码助手，充分利用 AI 提升日常研发的效率和体验。

## 代码补全篇

大模型根据用户提供的 Prompt 输出结果，这意味着，我们提供的 Prompt 越丰富、准确，大模型的输出就越好，因此，尽可能提供详细的 Prompt 是关键，和使用 ChatGPT 时需要把所有背景信息显式地输入给大模型不同，腾讯云代码助手可以从您的 IDE 中自动推断需要用到的上下文，并将其作为 Prompt 输入给大模型。接下来我们将分享一些基本技巧，以便您在使用腾讯云代码助手时能提供更加详细的上下文。

### 技巧一：打开或导入相关代码文件

在发送给大模型的 Prompt 中，除了当前代码文件的上下文以外，腾讯云代码助手还会将 IDE 的其他信息也加入到 Prompt 中，其中比较重要的有两个：

- IDE 所有打开文件中与当前代码相似的代码片段。
- 导入包中的相关代码。

因此，确保所有相关文件都是打开状态，并且 import 所有需要用到的代码文件，这样腾讯云代码助手就可以获取更多的上下文信息，从而推荐地更加准确。

### 技巧二：提供顶层注释

就像在汇报时通常需要先提供一个简短的概述一样，我们在文件顶部添加顶层的注释，描述文件的主要功能和预期目标。这有助于腾讯云代码助手理解整体上下文，提供更相关的代码建议。请在注释中包含您需要的详细内容并提供规范描述，以便腾讯云代码助手能接收到更多信息并提供更好的建议。

```
Generate Function(Enter) | Disable Generate Function Hints
1 # 这个文件创建了一个主页组件，允许用户查找任何食谱的营养信息。
2 # 具体要求如下：
3 # 1. 页面包含一个标题 "查找任何食谱的营养信息"
4 # 2. 页面包含一个文本区域和一个按钮。当点击按钮时，文本区域中的内容会发送到服务器。
5 # 3. 服务器返回的营养信息会显示在文本区域下面。
6
7 import requests
8 from flask import Flask, request, jsonify, render_template_string
9
10 app = Flask(__name__)
11
12 @app.route('/')
13 def home():
14     return render_template_string('''
        <!DOCTYPE html>
        <html>
        <head>
            <title>查找任何食谱的营养信息</title>
        </head>
        <body>
            <h1>查找任何食谱的营养信息</h1>
            <form action="/nutrition" method="POST">
                <textarea name="recipe" rows="10" cols="80"></textarea>
                <br>
                <input type="submit" value="查找">
            </form>
            <br>
            <div id="nutrition"></div>
    ''')
```

### 技巧三：提供有意义的变量和函数名

1. 使用有意义的变量和函数名有助于腾讯云代码助手更好地理解您的代码意图。

如果您有一个名为“foo”或者“calculate”的函数，腾讯云代码助手可能无法提供最佳的代码补全效果，因为它无法从命名中推断出您的意图。因此，在使用 AI 工具的过程中，要遵守良好的命名规范，例如：getUserData，清晰明了，大大提升了代码阅读性。良好的命名规范，可以提升代码阅读性，降低维护成本，同时也能让代码助手更容易理解，增强代码补全效果。

```
1 class User:
2     def __init__(self, username, email, password):
3         self.username = username
4         self.email = email
5         self.password = password
6
7 class UserManager:
8     def __init__(self):
9         self.users = []
10
11     def add_user(self, user):
12         self.users.append(user)
13         print(f"User {user.username} added successfully.")
14
15     def remove_user(self, username):
16         self.users = [user for user in self.users if user.username != username]
17         print(f"User {username} removed successfully.")
18
19     def find_user(self, username):
20         for user in self.users:
21             if user.username == username:
22                 return user
23         return None
24
25     def list_users(self):
26         for user in self.users:
```

2. 在代码补全场景中，注释是一个向模型提出明确需求的好方法。

函数名的描述能力有限，过长的函数名会影响代码的简洁性，因此函数注释可以帮助填补细节。在函数前添加准确的注释，描述函数的功能和参数，使得腾讯云代码助手能够更好地理解函数的目的和实现细节，从而提供更具体的建议。请注意，明确、简洁的注释更有利于腾讯云代码助手生成符合您预期的内容。

```
1 import hashlib
2 import sqlite3
3 from typing import Optional
4
5 # 数据库连接
6 conn = sqlite3.connect('users.db')
7 c = conn.cursor()
8
9 # 创建用户表
10 c.execute('''
11 CREATE TABLE IF NOT EXISTS users (
12     username TEXT PRIMARY KEY,
13     password_hash TEXT NOT NULL
14 )
15 ''')
16 conn.commit()
17
18 # 生成给定密码的哈希值
19 def get_hash(password: str) -> str:
    return hashlib.sha256(password.encode('utf-8')).hexdigest()
```

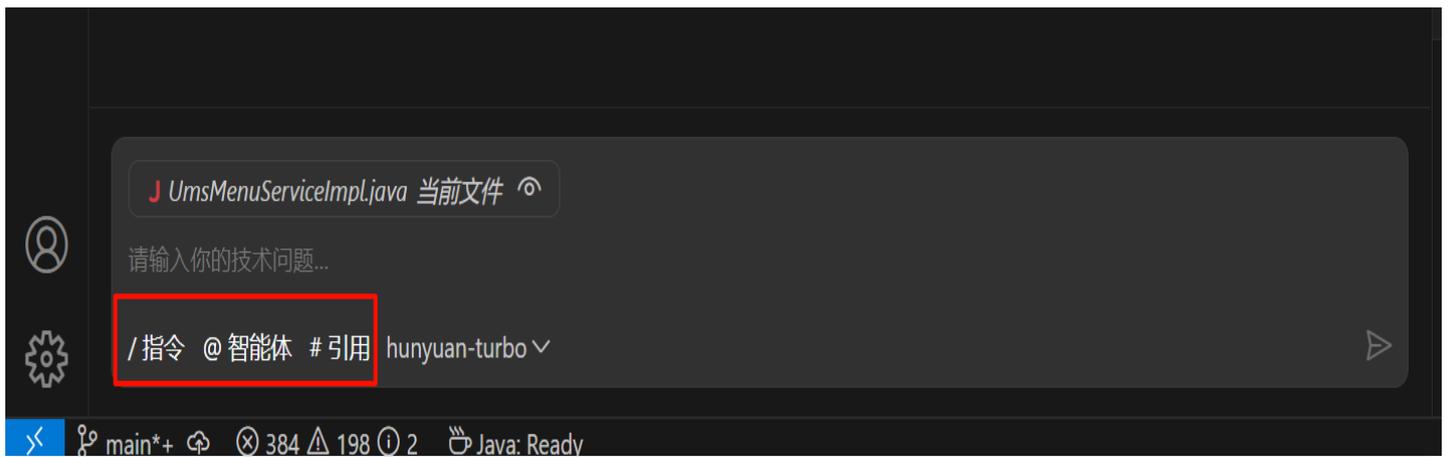
## 技术对话篇

腾讯云代码助手技术对话提供了类似 ChatGPT 的体验，支持多轮对话以及发起不同的对话，使您不用离开 IDE 环境就能得到对话式 AI 的帮助。另外，腾讯云代码助手技术对话可以利用编辑器中的代码文件提升回答效果，我们也提供了一系列内置指令帮助用户更便捷地引用编辑器中的上下文以及完成常见任务。

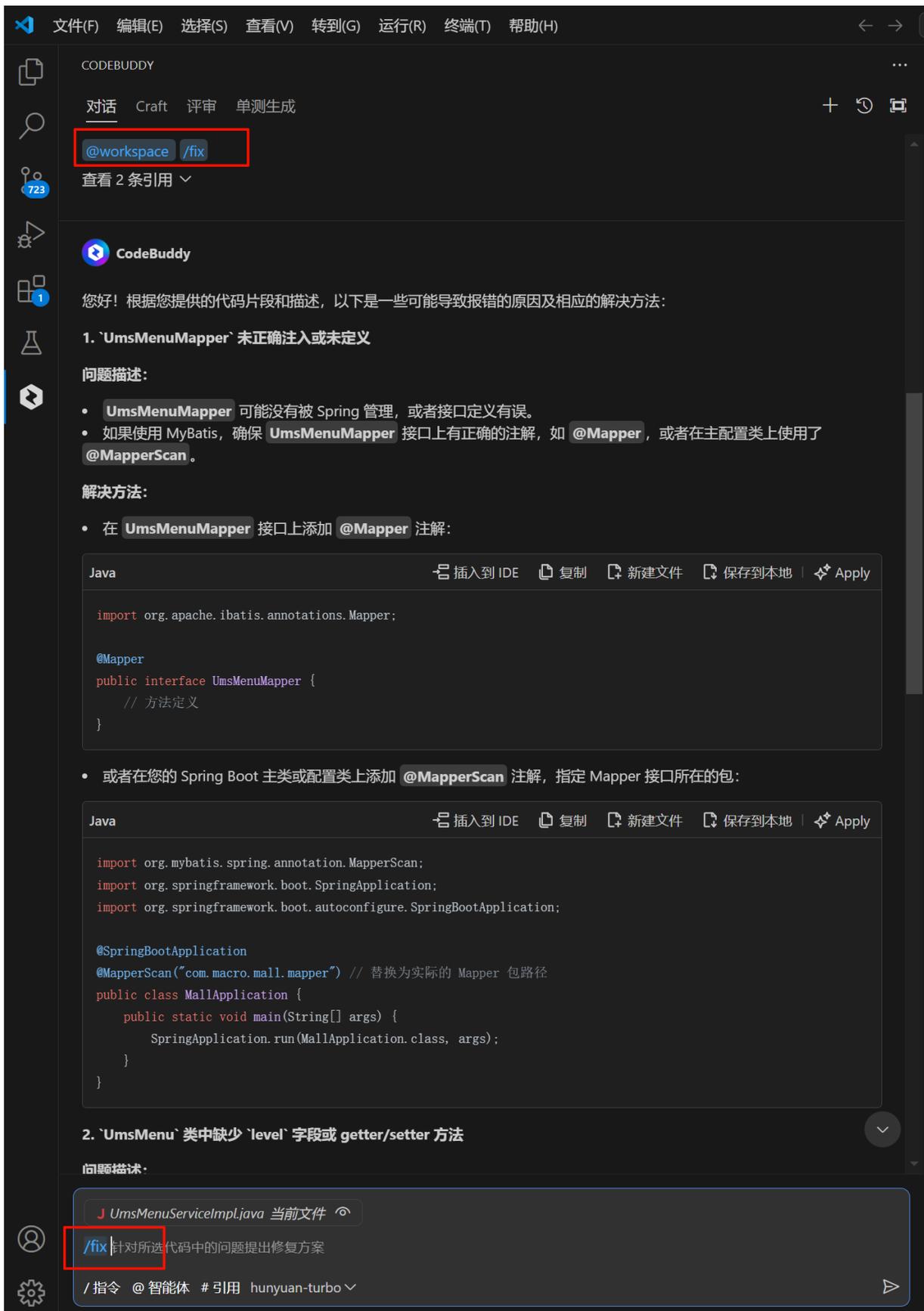
### 技巧一：使用指令提升沟通效率

腾讯云代码助手技术对话目前提供了3种类型的指令：

- 智能体 (@)：具有额外的知识、上下文与能力。
- 斜杠命令 (/)：常见提示词的快捷方式，例如我们提供了一些简单命令帮助您快速完成解释代码、修复代码等工作。
- 引用 (#)：通过 # 指令，可以选择添加当前项目文件、知识库或者指定文件作为上下文，增强模型输出的质量。



在技术对话中，可以直接选择指令进行快速对话。



## 技巧二：指定需要使用的包或模块

当您需要使用特定版本的包时，最好在对话中主动指定，这有助于腾讯云代码助手技术对话在给出建议时使用您希望使用的包，因为有时候腾讯云代码助手技术对话会使用较低版本的包给出建议。



CODEBUDDY

对话 Craft 评审 单测生成

使用tensorflow2.0实现卷积网络

查看 2 条引用

CodeBuddy

您好！看起来您当前提供的代码是关于Java后端服务（具体是 `UmsMenuServiceImpl.java`）的一部分，而您的问题涉及使用TensorFlow 2.0实现卷积神经网络（Convolutional Neural Network, CNN）。这两者属于不同的技术领域，Java后端服务通常用于处理业务逻辑和数据管理，而TensorFlow则用于构建和训练机器学习模型。

鉴于您的问题，我将专注于如何使用TensorFlow 2.0实现一个简单的卷积神经网络。如果您确实需要在Java后端中集成TensorFlow模型，后续我也会简要提及相关的步骤。

### 使用TensorFlow 2.0实现卷积神经网络

以下是一个使用TensorFlow 2.0构建和训练简单CNN模型的示例。这个示例使用经典的MNIST手写数字数据集。

#### 1. 安装TensorFlow

首先，确保您已经安装了TensorFlow 2.0。可以使用pip进行安装：

```
Bash 插入到 IDE 复制 新建文件 保存到本地 插入到终端 Apply
```

```
pip install tensorflow==2.0.0
```

#### 2. 导入必要的库

```
Python 插入到 IDE 复制 新建文件 保存到本地 Apply
```

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
```

#### 3. 加载和预处理数据

```
Python 插入到 IDE 复制 新建文件 保存到本地 Apply
```

```
# 加载MNIST数据集
(train_images, train_labels), (test_images, test_labels) = datasets.mnist.load_data()

# 归一化像素值到0-1之间
train_images = train_images.reshape((60000, 28, 28, 1)).astype('float32') / 255.0
test_images = test_images.reshape((10000, 28, 28, 1)).astype('float32') / 255.0

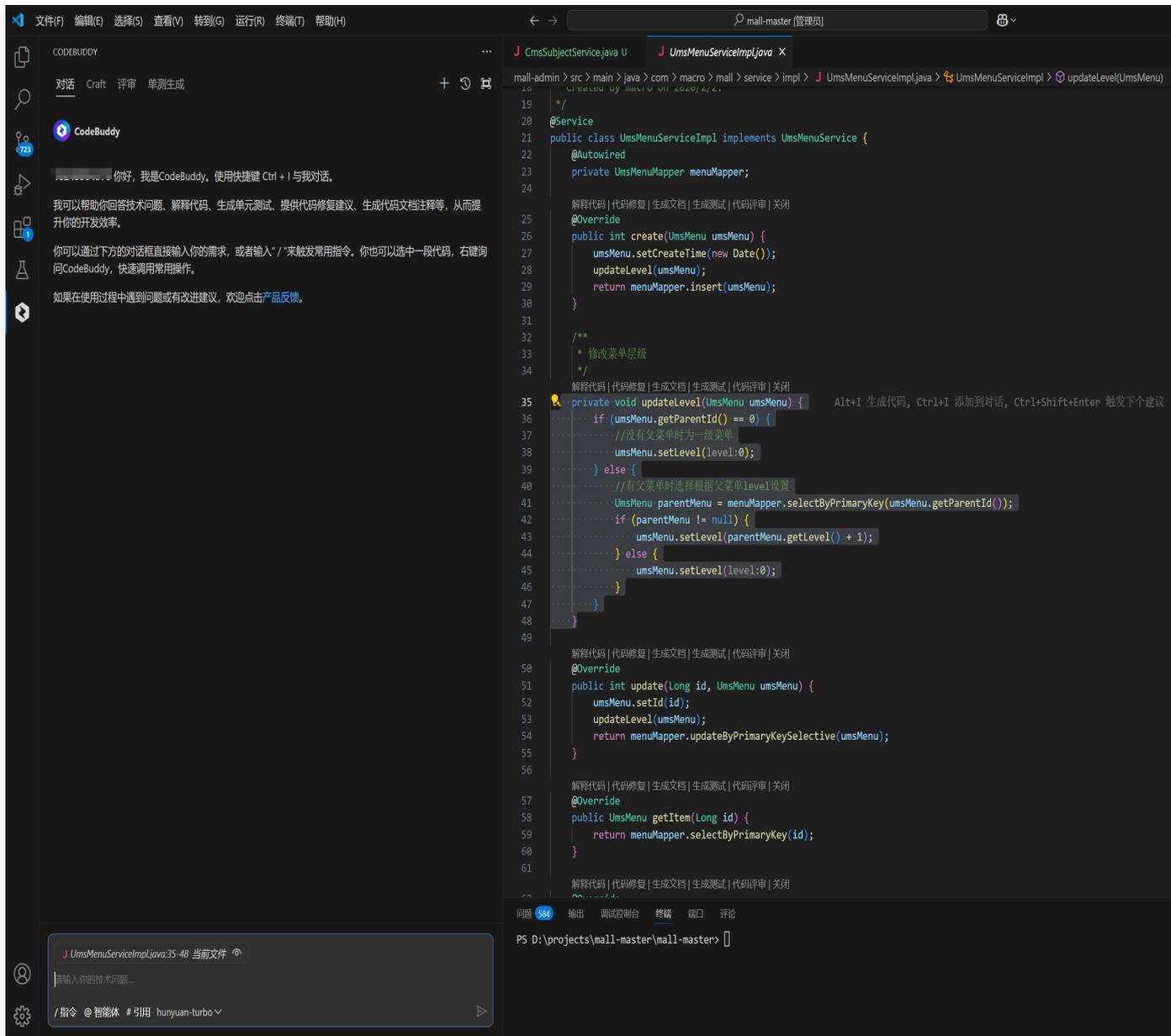
# 类别数
num_classes = 10
```

## 技巧三：使用上下箭头浏览历史提问

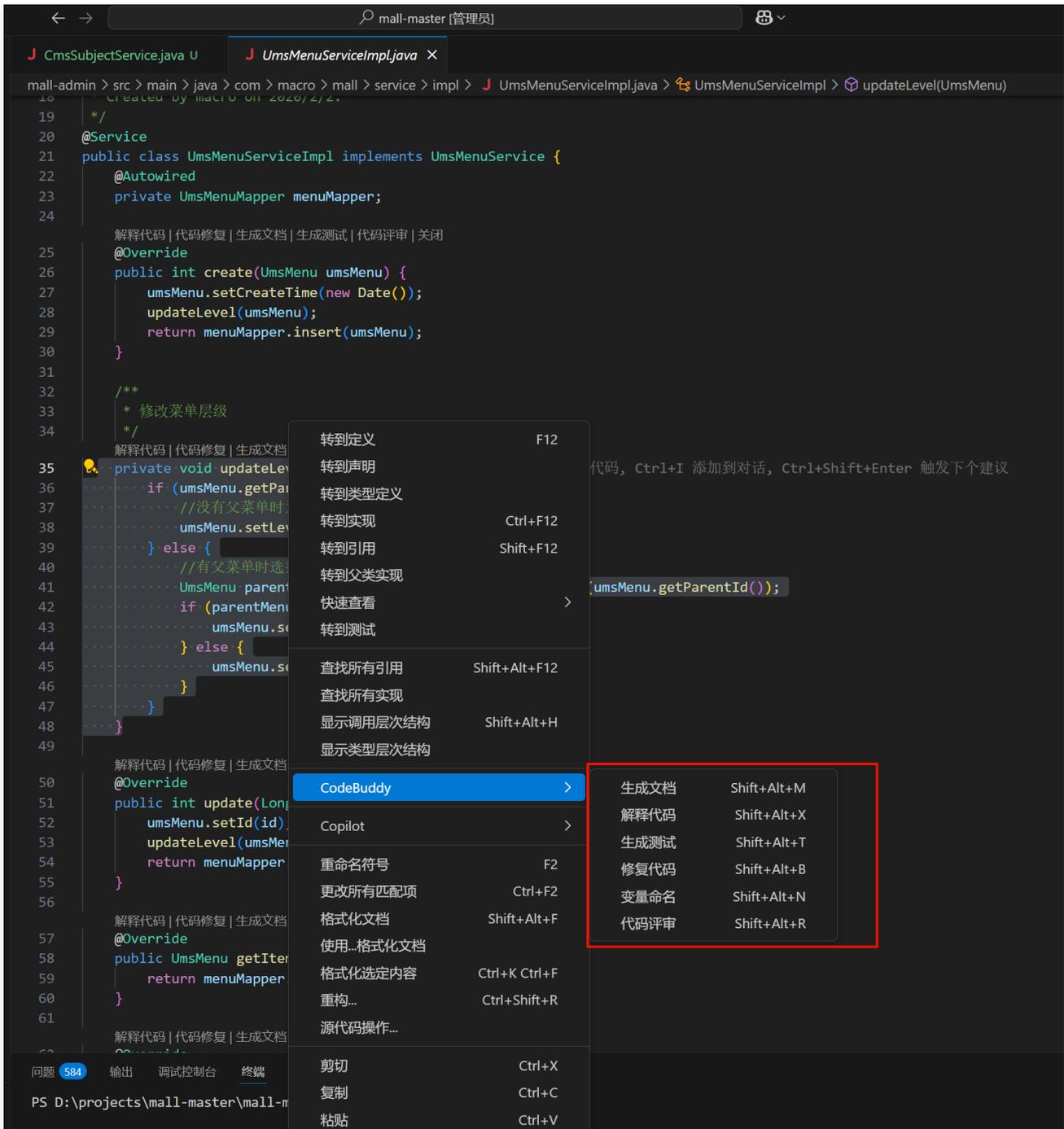
就像在终端中可以利用上下箭头浏览之前输入的命令一样，在腾讯云代码助手技术对话的输入框中，您也可以通过上下箭头浏览您之前的问题。

## 技巧四：选中相关代码，向 AI 提问建议

### 1. 选中片段，选择页面中对应功能向 AI 提问。

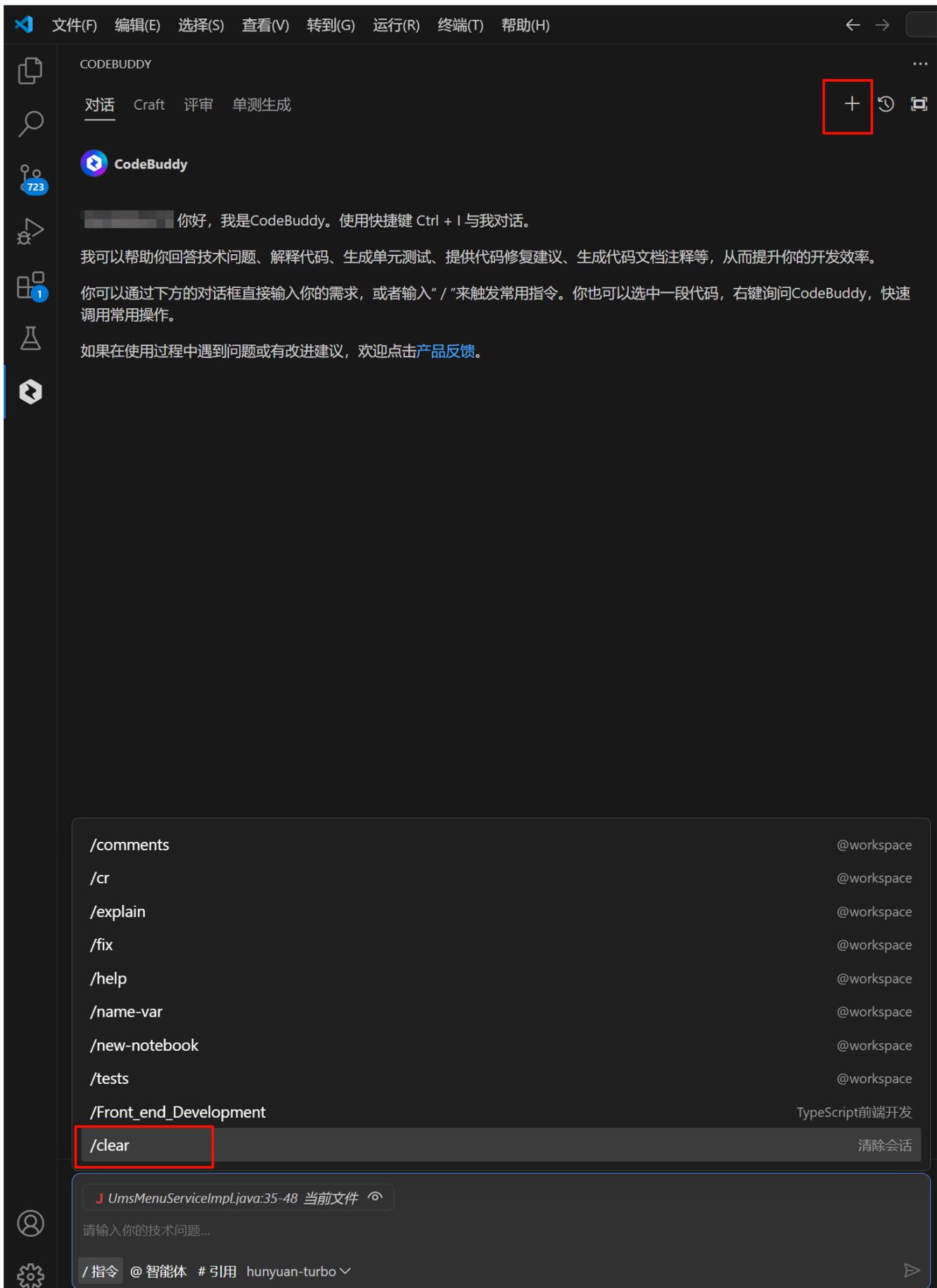


### 2. 或者右键选择对应功能向 AI 提问，得到解决方案。



## 技巧五：分不同会话询问关于不同主题的问题

通过新增对话或者采用对话中快捷指令 `/clear` 重新开启新对话，避免历史对话上下文带来的影响。



# 实战 Python Web API

最近更新时间：2025-04-14 09:54:23

本文旨在探索如何使用腾讯云代码助手的代码补全和对话能力，快速掌握一个 Python Web 应用程序开发，并提供 HTTP API 的预览站点。

## 学习 Prompt 提示方法

要充分利用腾讯云代码助手，需要了解 Prompt 的编写的策略，怎么样的提示工程，对返回的代码的质量在很大程度的影响，结果好与坏，取决于提示的清晰和准确程度。

那么如何充分利用腾讯云代码助手的潜力呢？本模块将指导完成有效提示的 Prompt 的描述，以获得最佳的代码建议，帮助日常开发中更快更好地编码。

## 提示工程基础和实践教学

在本单元中，将介绍：

- 什么是提示工程 Prompt Engineer？
- 提示工程要点。
- 提示工程的实践教学。
- 腾讯云代码助手如何从提示工程中学习进化。

## 什么是提示工程 Prompt Engineer？

- 提示工程用自然语言来编写清晰指令的过程，以指导腾讯云代码助手生成适应于项目特定需求的上下文相关代码。需要用更精确的自然语言，来确保代码在语法、功能和上下文方面都是正确的。不然，整个开发过程会因为错误的建议让效率变得低下。对提示工程有了了解后，怎么写好自然语言，让代码助手能快速理解，从而节省了时间和精力。
- 现在您已经了解了什么是提示工程，下面来学习一下它的3S 要点。

## 提示工程3S 要点

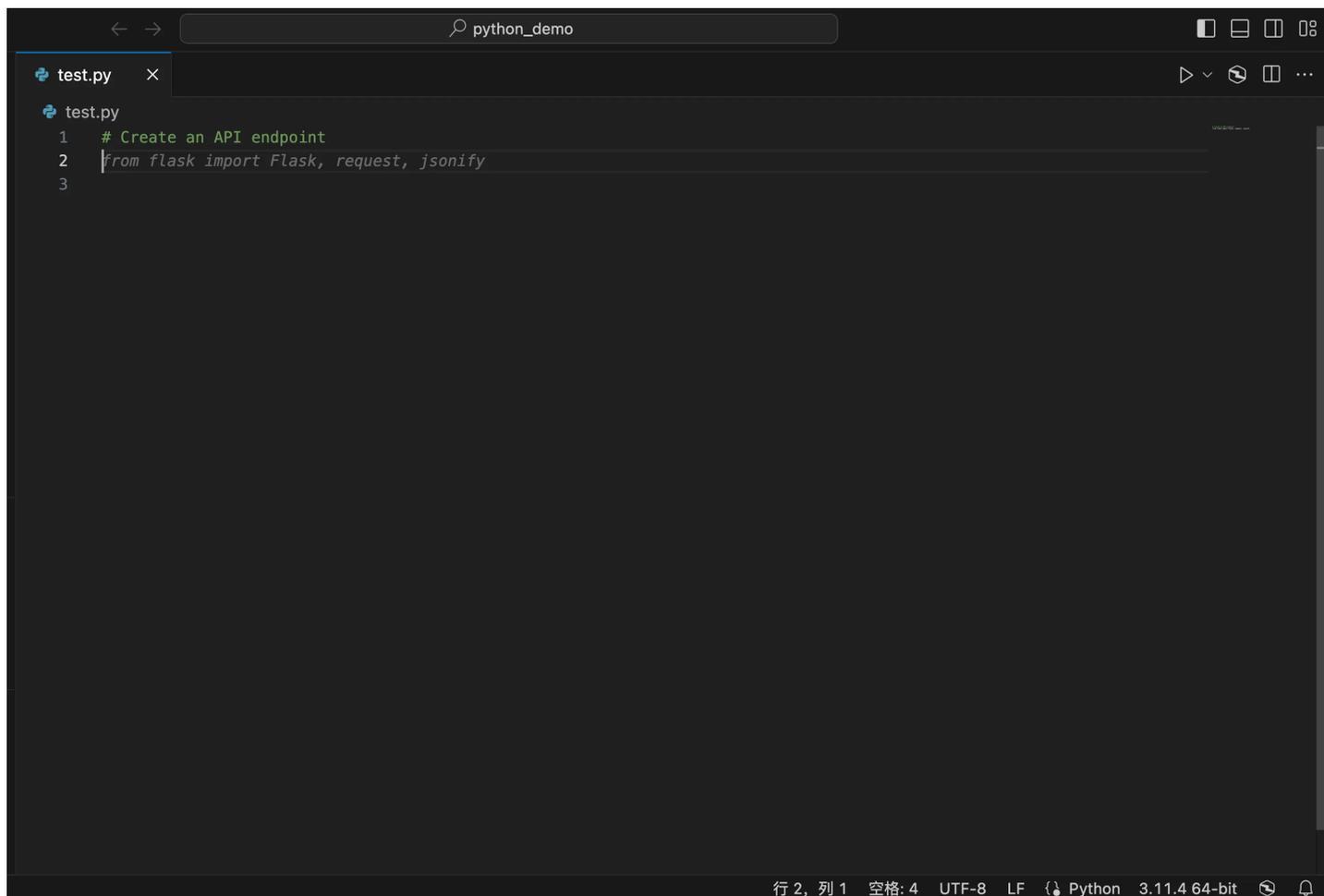
提示工程的基本原理，可以总结为3个 S 如下。这些核心规则是创建有效提示的基础。

- 单个 Single：始终将提示集中在单个、定义明确的任务或问题上。
- 具体 Specific：确保说明明确且详细，最好能附带一个示例或者模拟信息结构。具体且具象带来理解会带来更精确的代码建议。
- 简短 Short：在具体的同时，保持提示简明扼要。这种平衡确保了清晰度，而不会使腾讯云代码助手超载或使交互复杂化。

## 提示工程的实践教学

腾讯云代码助手的输出质量取决于制作提示的程度。因此，设计有效的提示对于确保实现预期结果至关重要。例如，如果出现以下提示：

```
# Create an API endpoint
```



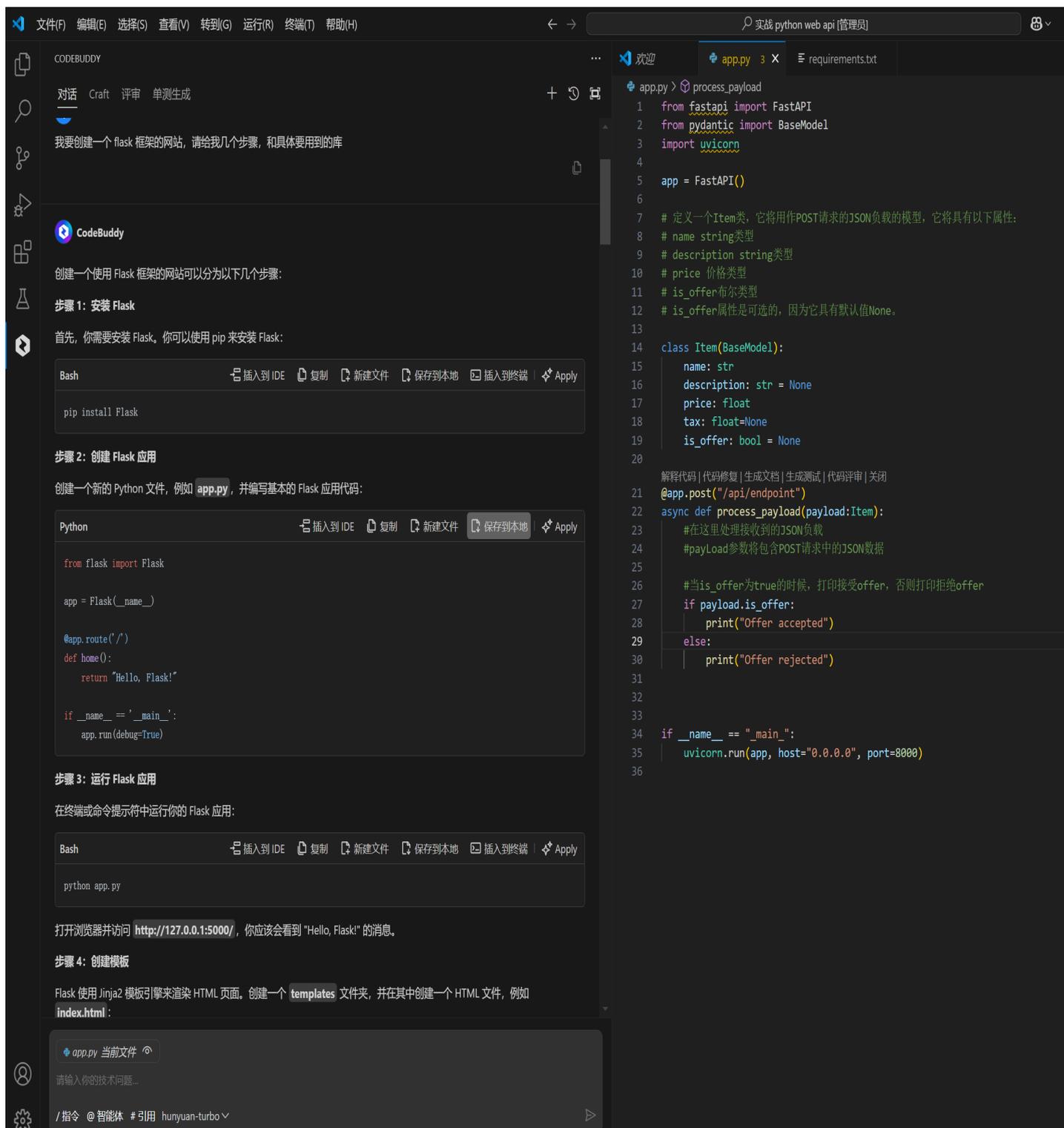
```
python_demo  
test.py  
test.py  
1 # Create an API endpoint  
2 from flask import Flask, request, jsonify  
3  
行 2, 列 1 空格: 4 UTF-8 LF Python 3.11.4 64-bit
```

由于提示不明确且模糊，腾讯云代码助手的结果可能不是很清楚。例如，它可能使用您不知道的框架，或者需要您无法识别的数据的端点。于是最坏情况下会不吐出任何建议，比较好的也只会吐出一行，并根据您的建议确认与否来进一步理解您的需求。

接下来，我们用腾讯云代码助手，实现 Python Web 网站。

## 在对话框中善用提示词

在一个空的项目中，对话会起到加速编码热身的效果，例如让对话帮助您快速生成一个带有原始代码的脚手架，或者给您一些方向性建议。在这个空项目中，我通过清晰的描述，拿到了方向性的尝试。



CODEBUDDY

对话 Craft 评审 单测生成

我要创建一个 flask 框架的网站，请给我几个步骤，和具体要用到的库

CodeBuddy

创建一个使用 Flask 框架的网站可以分为以下几个步骤：

步骤 1: 安装 Flask

首先，你需要安装 Flask，你可以使用 pip 来安装 Flask：

```
Bash pip install Flask
```

步骤 2: 创建 Flask 应用

创建一个新的 Python 文件，例如 `app.py`，并编写基本的 Flask 应用代码：

```
Python from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return "Hello, Flask!"

if __name__ == '__main__':
    app.run(debug=True)
```

步骤 3: 运行 Flask 应用

在终端或命令提示符中运行你的 Flask 应用：

```
Bash python app.py
```

打开浏览器并访问 `http://127.0.0.1:5000/`，你应该会看到 "Hello, Flask!" 的消息。

步骤 4: 创建模板

Flask 使用 Jinja2 模板引擎来渲染 HTML 页面。创建一个 `templates` 文件夹，并在其中创建一个 HTML 文件，例如 `index.html`：

app.py 当前文件

请输入你的技术问题...

/指令 @智能体 #引用 hunyuan-turbo

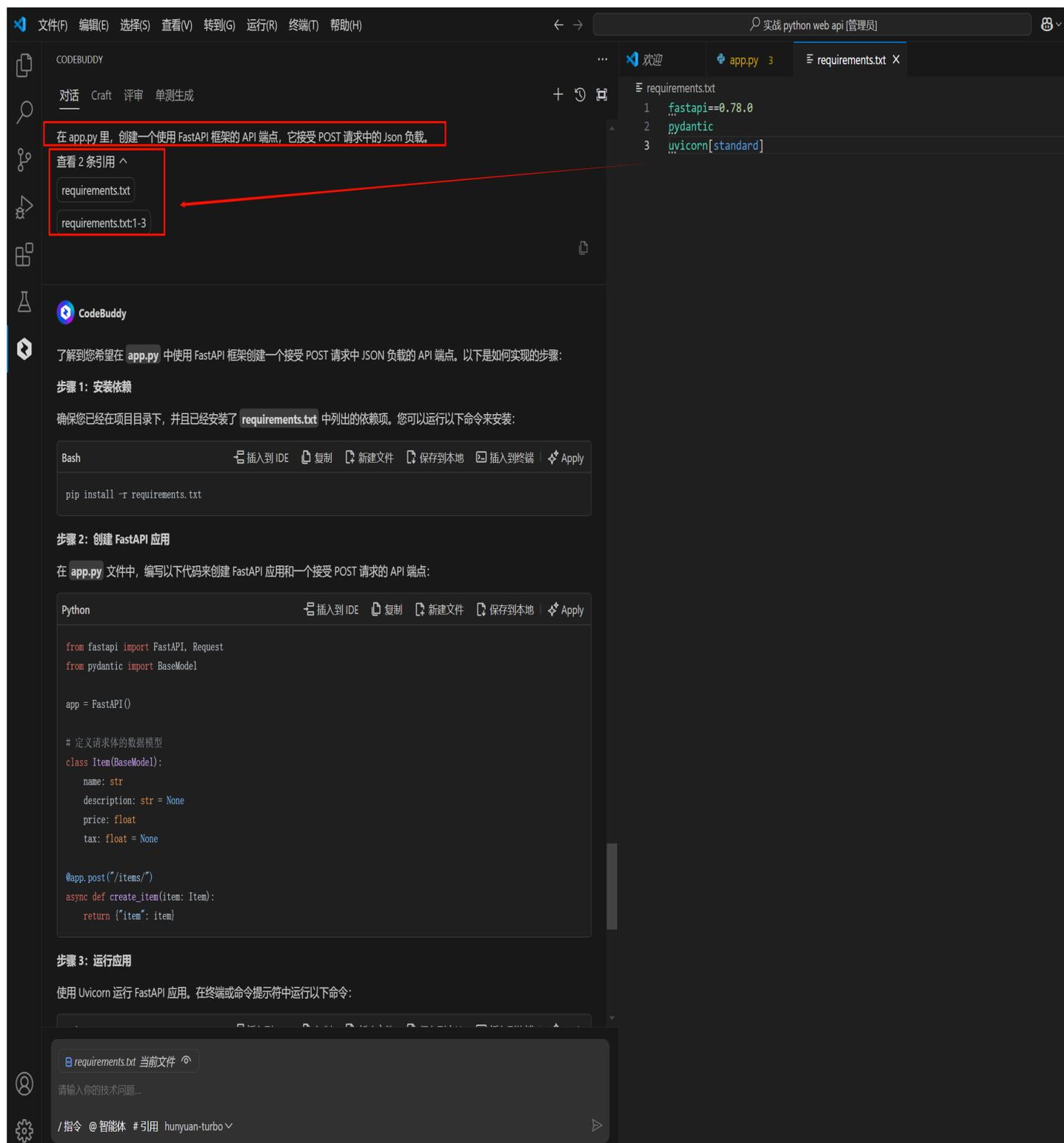
```
app.py > process_payload
1 from fastapi import FastAPI
2 from pydantic import BaseModel
3 import uvicorn
4
5 app = FastAPI()
6
7 # 定义一个Item类，它将用作POST请求的JSON负载的模型，它将具有以下属性：
8 # name string类型
9 # description string类型
10 # price 价格类型
11 # is_offer布尔类型
12 # is_offer属性是可选的，因为它具有默认值None。
13
14 class Item(BaseModel):
15     name: str
16     description: str = None
17     price: float
18     tax: float=None
19     is_offer: bool = None
20
21 @app.post("/api/endpoint")
22 async def process_payload(payload:Item):
23     #在这里处理接收到的JSON负载
24     #payload参数将包含POST请求中的JSON数据
25
26     #当is_offer为true的时候，打印接受offer，否则打印拒绝offer
27     if payload.is_offer:
28         print("Offer accepted")
29     else:
30         print("Offer rejected")
31
32
33
34 if __name__ == "__main__":
35     uvicorn.run(app, host="0.0.0.0", port=8000)
36
```

## 善用选中引用及进一步提问

这里有两个技巧：

1. 引用文件让会话模型的系统提问词中强制带有当前文件（当前窗口中的文字）的字符。
2. 不要吝啬，多在当前对话下进一步提问，纠正方向。用了 LangChain 的思想，把复杂任务拆成小任务，同时模型支持上下文有记忆能力，从而有更强大的推理性。

如下图，尝试打开一个 requirements.txt。这个文件用于定义 Python 文件所需库。通过它，让对话更清楚理解您想要做的需求。同时根据上文的对话继续回复，大多可以生成您想要的代码。



The screenshot shows the CodeBuddy IDE interface. The chat window on the left contains the following text:

在 app.py 里，创建一个使用 FastAPI 框架的 API 端点，它接受 POST 请求中的 Json 负载。

查看 2 条引用 ^

- requirements.txt
- requirements.txt:1-3

The main chat area displays the following instructions:

了解到您希望在 `app.py` 中使用 FastAPI 框架创建一个接受 POST 请求中 JSON 负载的 API 端点。以下是如何实现步骤：

**步骤 1: 安装依赖**

确保您已经在项目目录下，并且已经安装了 `requirements.txt` 中列出的依赖项。您可以运行以下命令来安装：

```
Bash pip install -r requirements.txt
```

**步骤 2: 创建 FastAPI 应用**

在 `app.py` 文件中，编写以下代码来创建 FastAPI 应用和一个接受 POST 请求的 API 端点：

```
Python from fastapi import FastAPI, Request
from pydantic import BaseModel

app = FastAPI()

# 定义请求体的数据模型
class Item(BaseModel):
    name: str
    description: str = None
    price: float
    tax: float = None

@app.post("/items/")
async def create_item(item: Item):
    return {"item": item}
```

**步骤 3: 运行应用**

使用 Uvicorn 运行 FastAPI 应用。在终端或命令提示符中运行以下命令：

```
uvicorn app:app
```

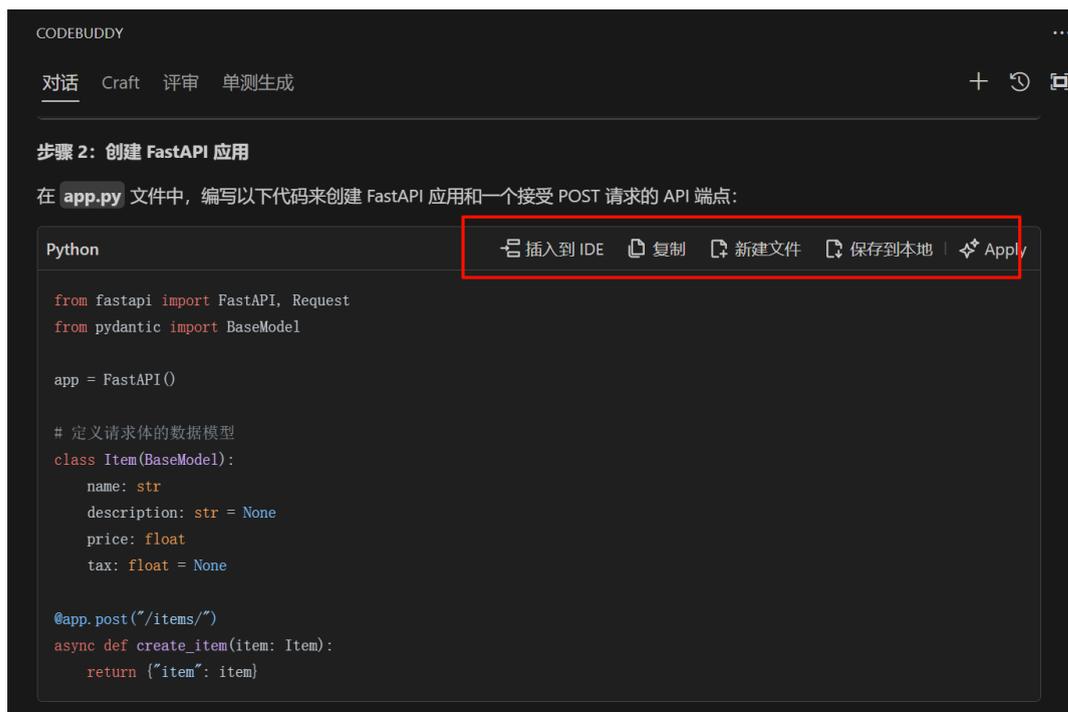
The code editor on the right shows the contents of `requirements.txt`:

```
requirements.txt
1 fastapi==0.78.0
2 pydantic
3 uvicorn[standard]
```

The terminal at the bottom shows the command prompt and the current file being edited:

```
/指令 @智能体 # 引用 hunyuan-turbo
```

在生成的示例代码框中，内置了快捷操作，方便一键插入到代码区域，如下：



CODEBUDDY

对话 Craft 评审 单测生成

步骤 2: 创建 FastAPI 应用

在 `app.py` 文件中, 编写以下代码来创建 FastAPI 应用和一个接受 POST 请求的 API 端点:

```
Python
```

```
from fastapi import FastAPI, Request
from pydantic import BaseModel

app = FastAPI()

# 定义请求体的数据模型
class Item(BaseModel):
    name: str
    description: str = None
    price: float
    tax: float = None

@app.post("/items/")
async def create_item(item: Item):
    return {"item": item}
```

插入到 IDE 复制 新建文件 保存到本地 Apply

## 代码补全提示词描述, 快速完成细节调整

满足 Specific 具体的特征, 如下图, 通过详细的注释, 且定义无二义性。例如在11行并没有描述清楚布尔类型是可选还是必选, 代码助手会帮助您进一步增加提示, 而不是立刻帮助您完成代码补全。

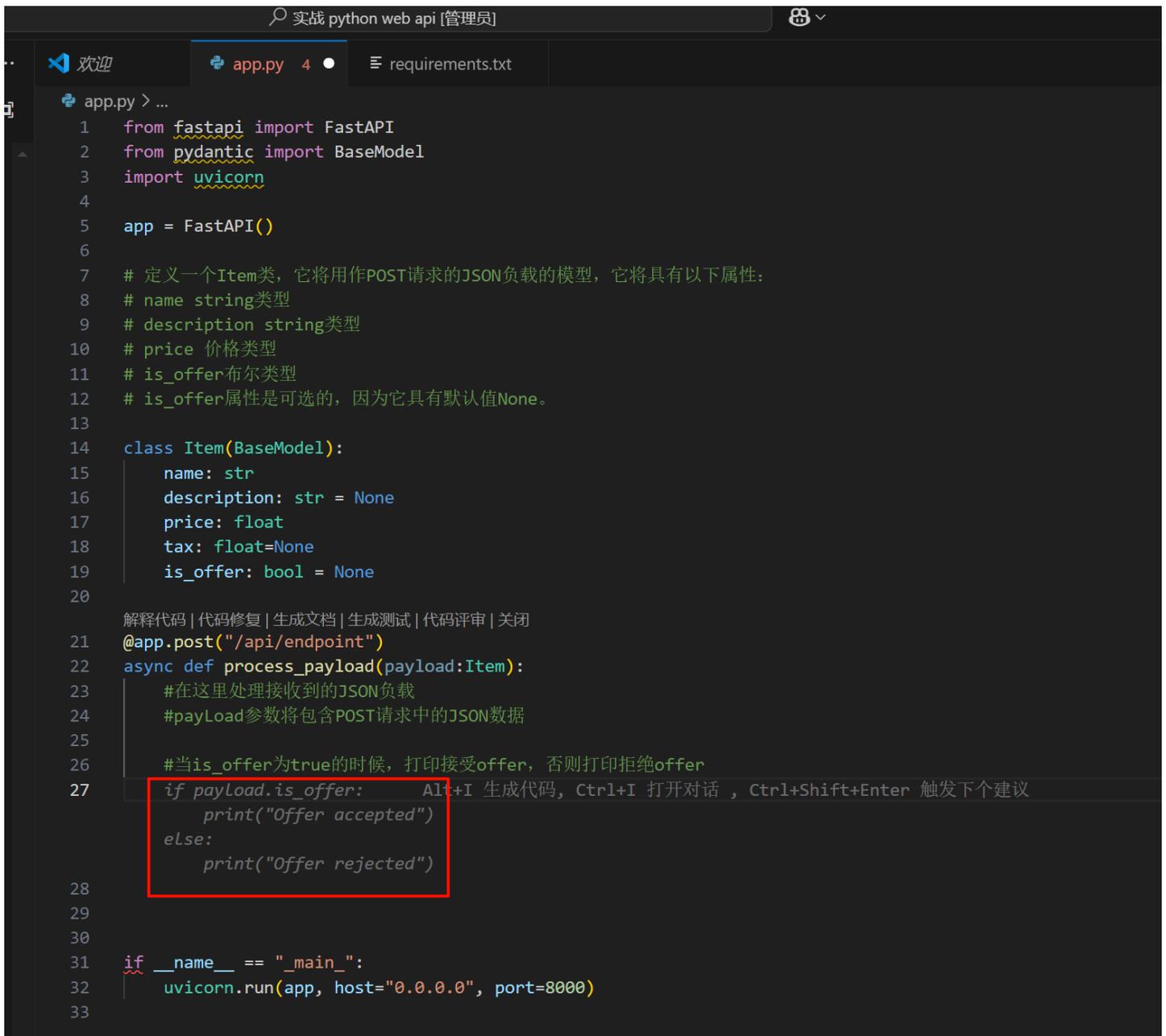
一旦完成后, 13行并不会立刻生成, 因为他们在思考是否有进一步补全提示的可能。您只需要再按个回车。即可触发14行的完整补全。

接下来, 删除 payload 的对象, 此时由于上下文均在一个文件内, 代码助手基于 FIM (中间补全的能力), 快速补齐了新的对象类型。确认后也不会影响当前代码格式。这就是应用端做了很多截断策略。

The screenshot shows the CodeBuddy IDE interface. On the left, there's a sidebar with navigation icons and a search bar. The main area is divided into three sections:

- Top Section:** A text area with instructions in Chinese, including "步骤 1: 安装依赖" (Step 1: Install dependencies) and "步骤 2: 创建 FastAPI 应用" (Step 2: Create FastAPI application). It includes a terminal window with the command `pip install -r requirements.txt`.
- Middle Section:** A code editor showing Python code for a FastAPI application. The code includes imports for `FastAPI`, `BaseModel`, and `uvicorn`. It defines an `Item` class with attributes `name`, `description`, `price`, `tax`, and `is_offer`. The `process_payload` endpoint is defined with a `payload: Item` parameter, which is highlighted with a red box.
- Bottom Section:** A terminal window with the command `uvicorn app:app --reload`.

记录下来继续进行补充，输入业务逻辑，要保持 Short（第三个S）简短。可能您这个需求很大，但由于当前为补全场景，需要更快的拆解需求并通过简短精炼的语句给出描述。如下：



```
实战 python web api [管理员]
app.py 4 requirements.txt
app.py > ...
1 from fastapi import FastAPI
2 from pydantic import BaseModel
3 import uvicorn
4
5 app = FastAPI()
6
7 # 定义一个Item类，它将用作POST请求的JSON负载的模型，它将具有以下属性：
8 # name string类型
9 # description string类型
10 # price 价格类型
11 # is_offer布尔类型
12 # is_offer属性是可选的，因为它具有默认值None。
13
14 class Item(BaseModel):
15     name: str
16     description: str = None
17     price: float
18     tax: float=None
19     is_offer: bool = None
20
21 解释代码 | 代码修复 | 生成文档 | 生成测试 | 代码评审 | 关闭
22 @app.post("/api/endpoint")
23 async def process_payload(payload:Item):
24     #在这里处理接收到的JSON负载
25     #payLoad参数将包含POST请求中的JSON数据
26     #当is_offer为true的时候，打印接受offer，否则打印拒绝offer
27     if payload.is_offer:
28         print("Offer accepted")
29     else:
30         print("Offer rejected")
31
32 if __name__ == "__main__":
33     uvicorn.run(app, host="0.0.0.0", port=8000)
```

连续回车，代码助手会做一些强制刷逻辑。当然后面也会提供快捷键和按钮，来更快捷的强制触发补全。其好处有：

- 补全未必正确的时候，让缓存失效，强制再返回一个结果。
- 一些二义性场景下可以强制再返回一个不同结果。
- 在空行处执行代码补全，会得到执行入口、测试函数等辅助业务的代码。

如下图，代码助手继续生成了补全建议。

```
实战 python web api [管理员]
app.py 3 requirements.txt
app.py > ...
1 from fastapi import FastAPI
2 from pydantic import BaseModel
3 import uvicorn
4
5 app = FastAPI()
6
7 # 定义一个Item类，它将用作POST请求的JSON负载的模型，它将具有以下属性：
8 # name string类型
9 # description string类型
10 # price 价格类型
11 # is_offer布尔类型
12 # is_offer属性是可选的，因为它具有默认值None。
13
14 class Item(BaseModel):
15     name: str
16     description: str = None
17     price: float
18     tax: float=None
19     is_offer: bool = None
20
21 解释代码 | 代码修复 | 生成文档 | 生成测试 | 代码评审 | 关闭
22 @app.post("/api/endpoint")
23 async def process_payload(payload: Item):
24     #在这里处理接收到的JSON负载
25     #payload参数将包含POST请求中的JSON数据
26
27     #当is_offer为true的时候，打印接受offer，否则打印拒绝offer
28     if payload.is_offer:
29         print("Offer accepted")
30     else:
31         print("Offer rejected")
32
33 #返回JSON响应 Alt+I 生成代码, Ctrl+I 打开对话, Ctrl+Shift+Enter 触发下个建议
34
```

代码完成编写后不知道怎么运行，可以在对话框中，向代码助手提问“我要怎么安装依赖并运行程序”。如下图：

The screenshot displays the CodeBuddy IDE interface. On the left, a chat window contains the following text:

对话 Craft 评审 单测生成

我要怎么安装依赖并运行程序。

查看 2 条引用

**CodeBuddy**

要安装依赖并运行您的 FastAPI 程序，请按照以下步骤操作：

**1. 安装依赖**

首先，确保您已经在项目目录下（即包含 `app.py` 和 `requirements.txt` 的目录）。然后，在终端或命令提示符中运行以下命令来安装 `requirements.txt` 中列出的依赖项：

```
Bash pip install -r requirements.txt
```

**requirements.txt** 文件应包含以下内容：

```
fastapi==0.78.0
pydantic
uvicorn[standard]
```

**2. 运行程序**

安装完依赖项后，您可以使用 Uvicorn 来运行您的 FastAPI 应用。在终端或命令提示符中运行以下命令：

```
Bash uvicorn app:app --reload
```

这里的 `app:app` 指的是 `app.py` 文件中的 `app` 实例。`--reload` 参数表示在代码更改时自动重新加载服务器，这对于开发非常有用。

**3. 访问 API 文档**

FastAPI 自动生成了交互式 API 文档，您可以在浏览器中访问以下 URL 来查看和测试您的 API：

- **Swagger UI:** <http://127.0.0.1:8000/docs>
- **ReDoc:** <http://127.0.0.1:8000/redoc>

**4. 测试 API 端点**

您可以使用 `curl` 或 Postman 来测试您的 API 端点。以下是使用 `curl` 的示例：

app.py 当前文件

请输入你的技术问题...

/指令 @智能体 # 引用 hunyuan-turbo

On the right, the code editor shows the `app.py` file with the following code:

```
app.py
app.py > process_payload
1 from fastapi import FastAPI
2 from pydantic import BaseModel
3 import uvicorn
4
5 app = FastAPI()
6
7 # 定义一个Item类，它将用作POST请求的JSON负载的模型。它将具有以下属性：
8 # name string类型
9 # description string类型
10 # price 价格类型
11 # is_offer布尔类型
12 # is_offer属性是可选的，因为它具有默认值None。
13
14 class Item(BaseModel):
15     name: str
16     description: str = None
17     price: float
18     tax: float=None
19     is_offer: bool = None
20
21 解释代码 | 代码修复 | 生成文档 | 生成测试 | 代码评审 | 关闭
22 @app.post("/api/endpoint")
23 async def process_payload(payload: Item):
24     #在这里处理接收到的JSON负载
25     #payload参数将包含POST请求中的JSON数据
26     #当is_offer为true的时候，打印接受offer，否则打印拒绝offer
27     if payload.is_offer:
28         print("Offer accepted")
29     else:
30         print("Offer rejected")
31
32 # 启动FastAPI应用程序 uvicorn app:app --reload # app:app是app.py文件中的app变量
33 if __name__ == "__main__":
34     uvicorn.run(app, host="127.0.0.1", port=8000)
35
36
37
38
```

At the bottom, the terminal output shows:

```
INFO: Waiting for application shutdown.
INFO: Application shutdown complete.
INFO: Finished server process [47448]
INFO: Stopping reloader process [3784]
PS D:\projects\实战 python web api>
* 还原的历史记录
```

## 运行效果

在终端中，依次执行命令，成功运行后，在浏览器中输入 `localhost:8000/docs` 来确认该端点显示是否正常工作，运行成功。

The image shows a development environment with a code editor on the left and a browser on the right. The code editor displays the following Python code for `app.py`:

```

1 from fastapi import FastAPI
2 from pydantic import BaseModel
3 import uvicorn
4
5 app = FastAPI()
6
7 # 定义一个Item类, 它将用作POST请求的JSON负载的模型, 它具有以下属性:
8 # name string类型
9 # description string类型
10 # price 价格类型
11 # is_offer布尔类型
12 # is_offer属性是可选的, 因为它具有默认值None.
13
14 class Item(BaseModel):
15     name: str
16     description: str = None
17     price: float
18     tax: float=None
19     is_offer: bool = None
20
21 @app.post("/api/endpoint")
22 async def process_payload(payload: Item):
23     #在这里处理接收到的JSON负载
24     #payload参数将包含POST请求中的JSON数据
25
26     #当is_offer为true的时候, 打印接受offer, 否则打印拒绝offer
27     if payload.is_offer:
28         print("Offer accepted")
29     else:
30         print("Offer rejected")
31
32 # 启动FastAPI应用程序 uvicorn app:app --reload # app:app是app.py
33 if __name__ == "__main__":
34     uvicorn.run(app, host="127.0.0.1", port=8000)
35
36
37
38

```

The browser window shows the Swagger UI for the application, displaying the following information:

- FastAPI 0.1.0 OAS3
- default
- POST /api/endpoint Process Payload
- Schemas:
  - HTTPValidationError: detail: Detail > [...]
  - Item:
    - name: Name > [...]
    - description: Description > [...]
    - price: Price > [...]
    - tax: Tax > [...]
    - is\_offer: Is Offer > [...]
  - ValidationError

The terminal output shows the application starting successfully and handling several GET requests to the documentation endpoints.

来试试业务逻辑写的对不对。单击 **Execute**, 运行正确。

default

POST /api/endpoint Process Payload

Parameters Cancel

No parameters

Request body required application/json

```
{
  "name": "string",
  "description": "string",
  "price": 0,
  "tax": 0,
  "is_offer": true
}
```

Execute Clear

## 小结

本文通过提示工程，教会您怎么用更好的提示工程，快速实战一个网站项目。

# 快速搭建在线相册应用

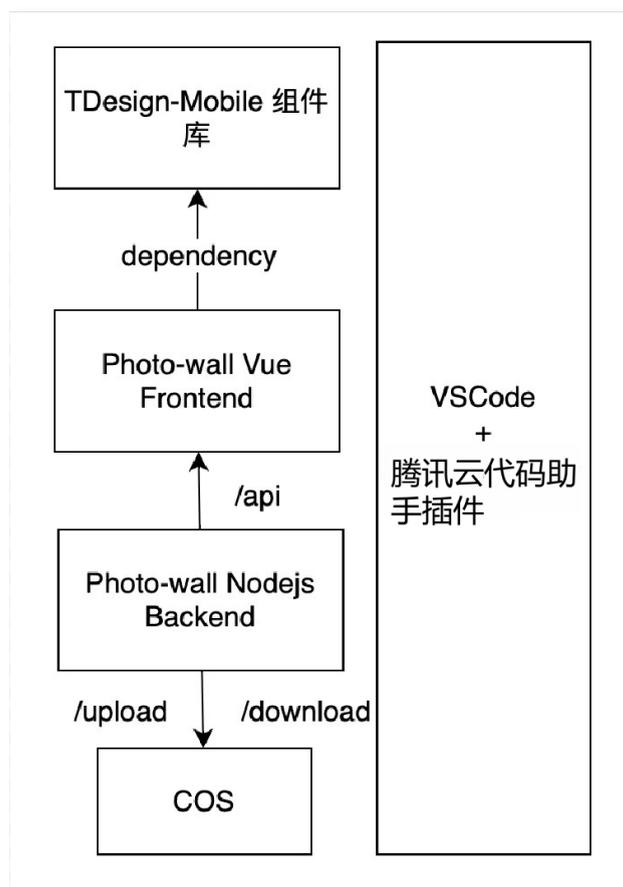
最近更新时间：2025-04-14 09:54:23

在云端开发平台 Cloud Studio 之上，通过代码助手的加持，可以快速搭建并开发一个过年都用得到的在线相册的小应用。请单击 [体验地址](#) 前往。

## 应用架构

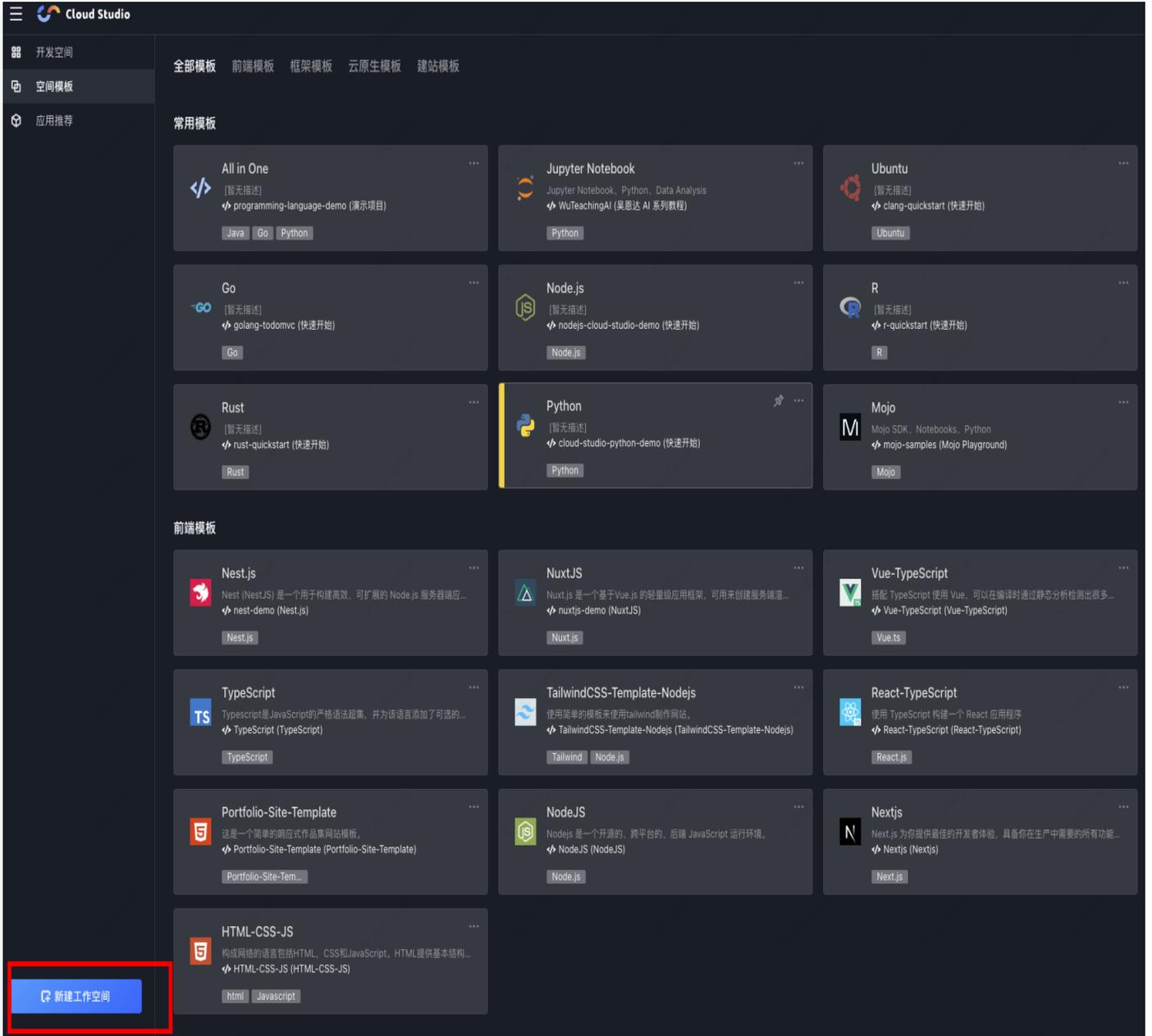
为了更快的进行开发，前端方面选择了 TDesign Mobile+Vue3，实现注册、登录、上传图片、看图片，并且支持移动端浏览器访问。

后端采用了 Node.js+ 数据库。数据库用于记录 ID 等 COS 文件信息。图片的 raw data 通过 COSClient SDK 存入 COS 桶中。



## 启动一个开发环境

1. 打开 [腾讯云 Cloud Studio](#)，单击左下角新建工作空间。



2. 选择一个 Node.js 开发环境，单击**新建**之后，即可进入开发环境。

### 新建工作空间

空间名称 \*

请输入工作空间名称

空间描述

简要描述一下这个工作空间的作用

0 / 255

工作类别 \*

托管空间 推荐

代码来源 \*

导入仓库  仓库地址  空

选择仓库服务商 [管理账号](#)

CODING  GitHub

开发环境 \* 版本

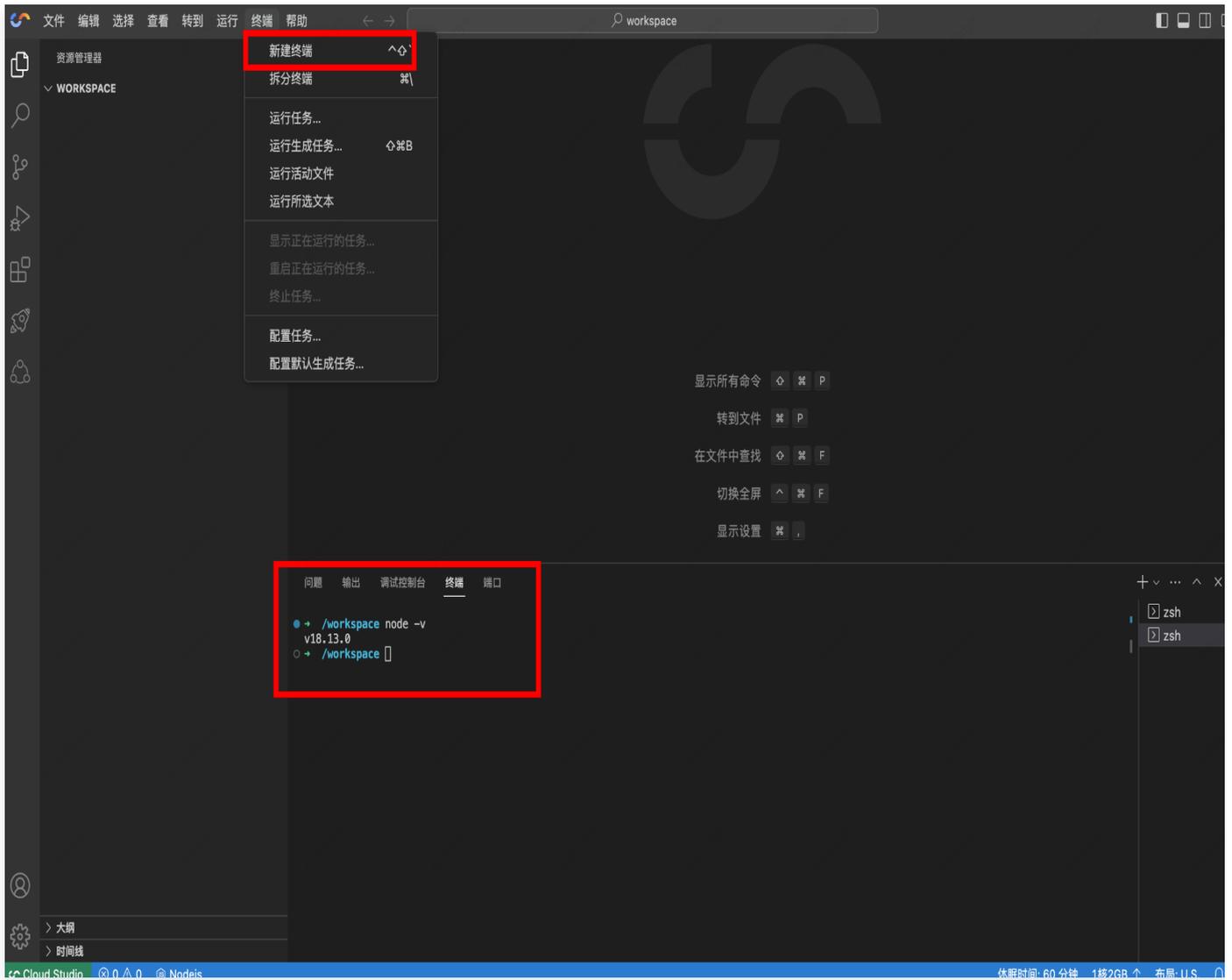
Node.js  node 18

规格配置 \*

标准版 2核4GB内存 / 8GB 存储

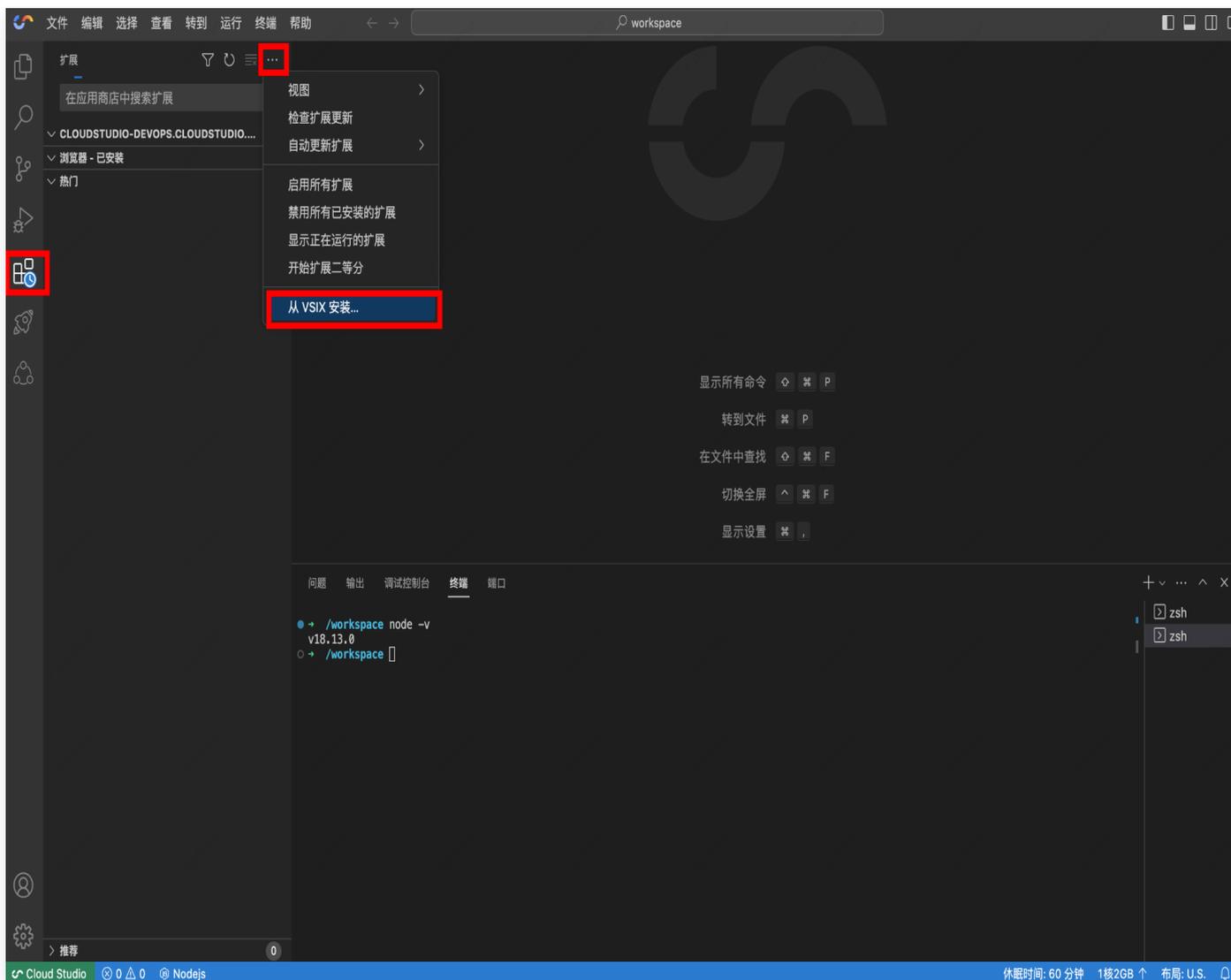
[新建](#) [取消](#)

3. 在界面导航栏中选择终端 > 新建终端，查看安装版本。

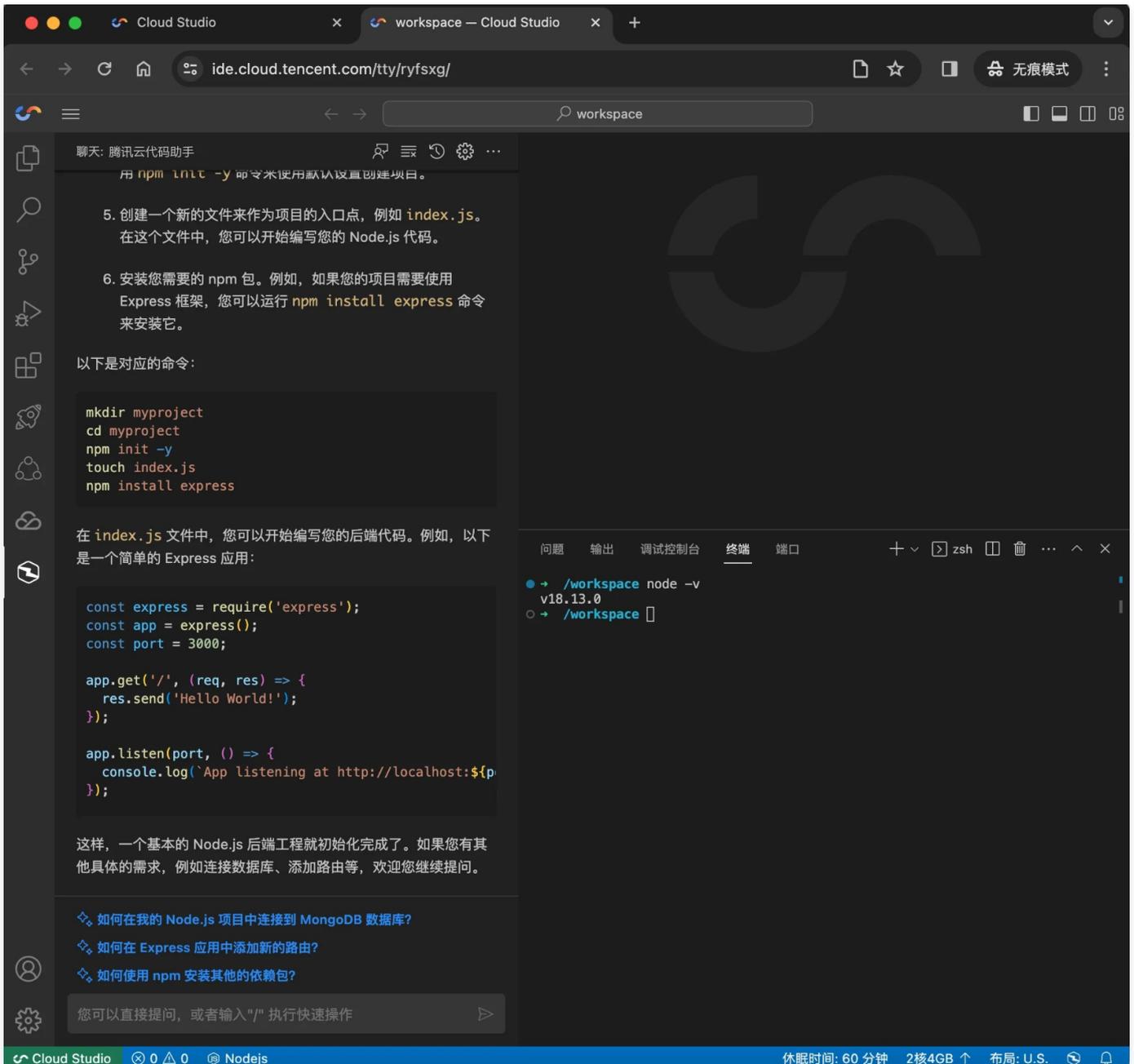


## 后端应用

1. 选择拓展图标 >  > 从 VSIX 安装，即可安装本地代码助手的插件安装包。



2. 接下来开始创建 Node.js 后端服务。基于代码助手的对话能力，输入提示词初始化一个 Node.js 项目的后端工程。



聊天: 腾讯云代码助手

用 `npm init -y` 命令使用默认设置创建项目。

5. 创建一个新的文件来作为项目的入口点, 例如 `index.js`。在这个文件中, 您可以开始编写您的 Node.js 代码。
6. 安装您需要的 npm 包。例如, 如果您的项目需要使用 Express 框架, 您可以运行 `npm install express` 命令来安装它。

以下是对应的命令:

```
mkdir myproject
cd myproject
npm init -y
touch index.js
npm install express
```

在 `index.js` 文件中, 您可以开始编写您的后端代码。例如, 以下是一个简单的 Express 应用:

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {
  res.send('Hello World!');
});

app.listen(port, () => {
  console.log(`App listening at http://localhost:${port}`);
});
```

这样, 一个基本的 Node.js 后端工程就初始化完成了。如果您有其他具体的需求, 例如连接数据库、添加路由等, 欢迎您继续提问。

- ❖ 如何在我的 Node.js 项目中连接到 MongoDB 数据库?
- ❖ 如何在 Express 应用中添加新的路由?
- ❖ 如何使用 npm 安装其他的依赖包?

您可以直接提问, 或者输入 "/" 执行快速操作

终端

```
➤ /workspace node -v
v18.13.0
○ /workspace
```

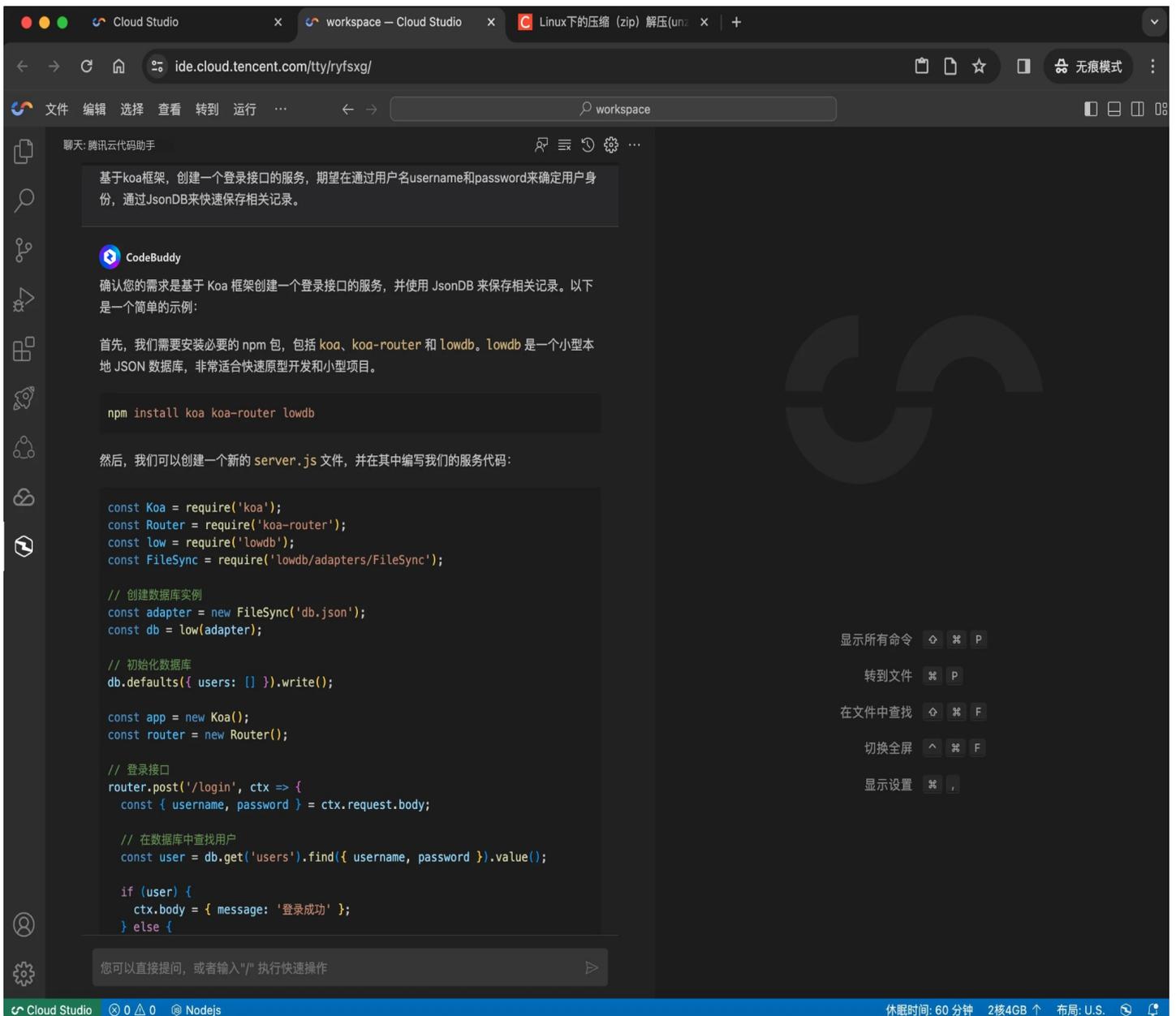
Cloud Studio 0 0 Nodejs 休眠时间: 60 分钟 2核4GB 布局: U.S.

后端服务至少应包含:

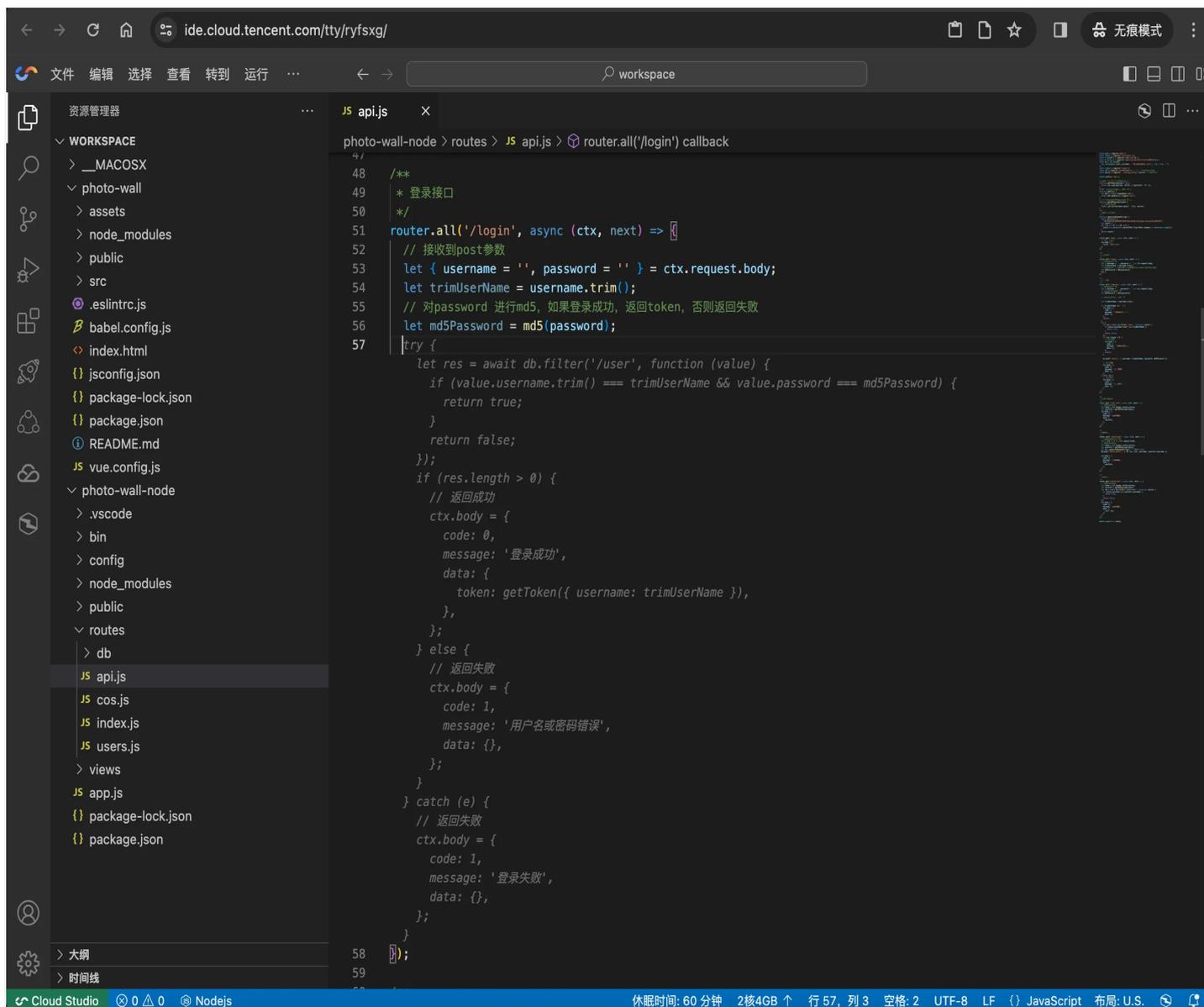
- 登录服务。
- 注册服务。
- 图片预览服务。
- 图片上传保存服务。

## 登录服务

在对话里继续提问: 基于 koa 框架, 创建一个登录接口的服务, 期望在通过用户名 `username` 和 `password` 来确定用户身份, 通过 `JsonDB` 来快速保存相关记录。



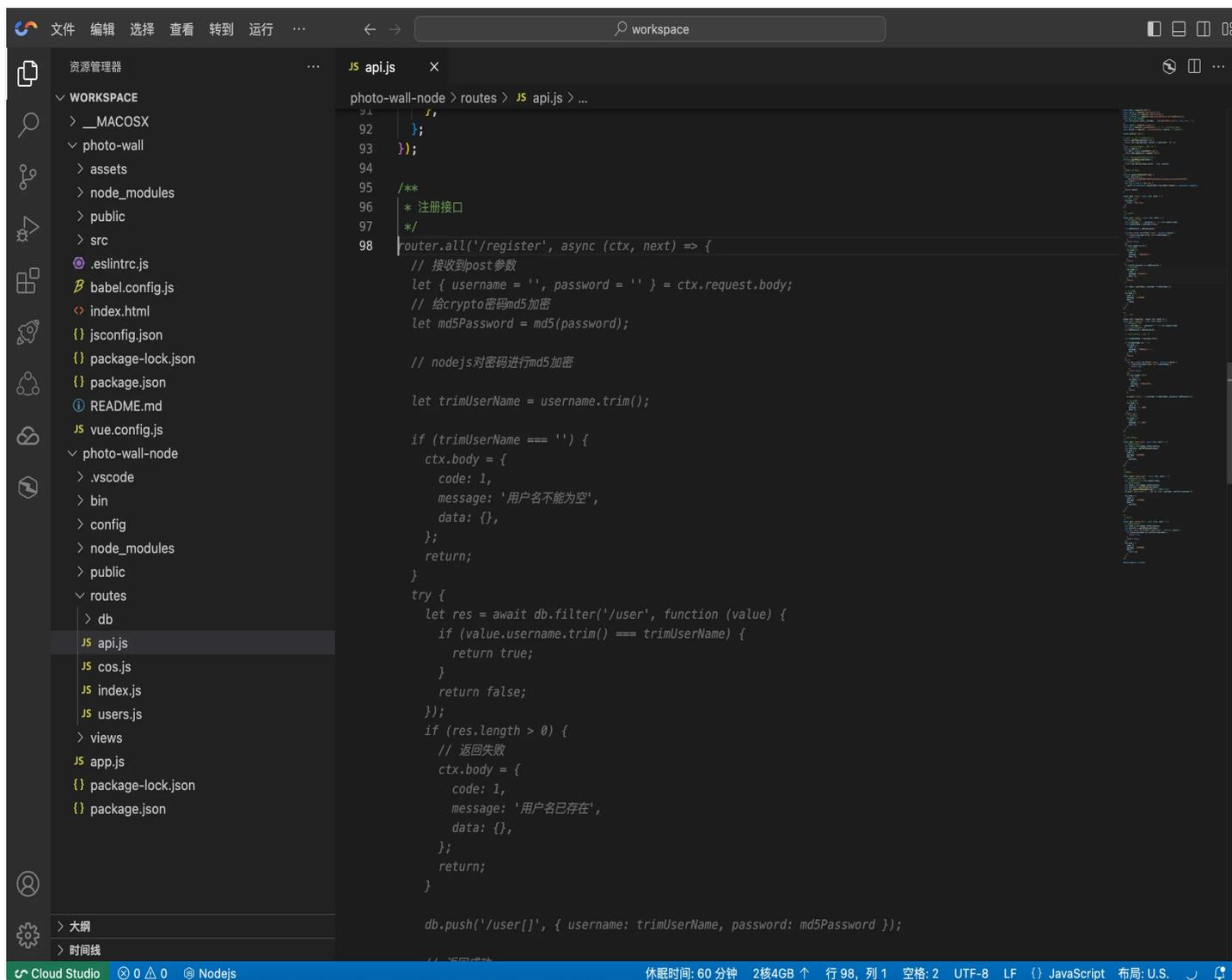
接着插入到当前工程, 进行代码补全的代码微调, 例如我们要对传过来的密码进行 md5 加密, 我通过行内注释输入 // 对 password 进行 md5, 如果登录成功, 返回 token, 否则返回失败, 接着代码助手触发了代码补全, 智能根据当前光标上下文, 进行行或者块补全。如下:



```
photo-wall-node > routes > JS api.js > router.all('/login') callback
48  /**
49   * 登录接口
50   */
51  router.all('/login', async (ctx, next) => {
52    // 接收到post参数
53    let { username = '', password = '' } = ctx.request.body;
54    let trimUserName = username.trim();
55    // 对password 进行md5, 如果登录成功, 返回token, 否则返回失败
56    let md5Password = md5(password);
57    try {
58      let res = await db.filter('user', function (value) {
59        if (value.username.trim() === trimUserName && value.password === md5Password) {
60          return true;
61        }
62        return false;
63      });
64      if (res.length > 0) {
65        // 返回成功
66        ctx.body = {
67          code: 0,
68          message: '登录成功',
69          data: {
70            token: getToken({ username: trimUserName }),
71          },
72        };
73      } else {
74        // 返回失败
75        ctx.body = {
76          code: 1,
77          message: '用户名或密码错误',
78          data: {},
79        };
80      }
81    } catch (e) {
82      // 返回失败
83      ctx.body = {
84        code: 1,
85        message: '登录失败',
86        data: {},
87      };
88    }
89  });
90  }
```

## 注册服务

AI 的联想能力（FIM – Fill in Middle），由于上文已完成注册用户的代码逻辑，当描述注册接口的时候，可以给予初步的补全建议。



```
91  },
92  });
93  });
94
95  /**
96  * 注册接口
97  */
98  router.all('/register', async (ctx, next) => {
  // 接收到post参数
  let { username = '', password = '' } = ctx.request.body;
  // 给crypto密码md5加密
  let md5Password = md5(password);

  // nodejs对密码进行md5加密

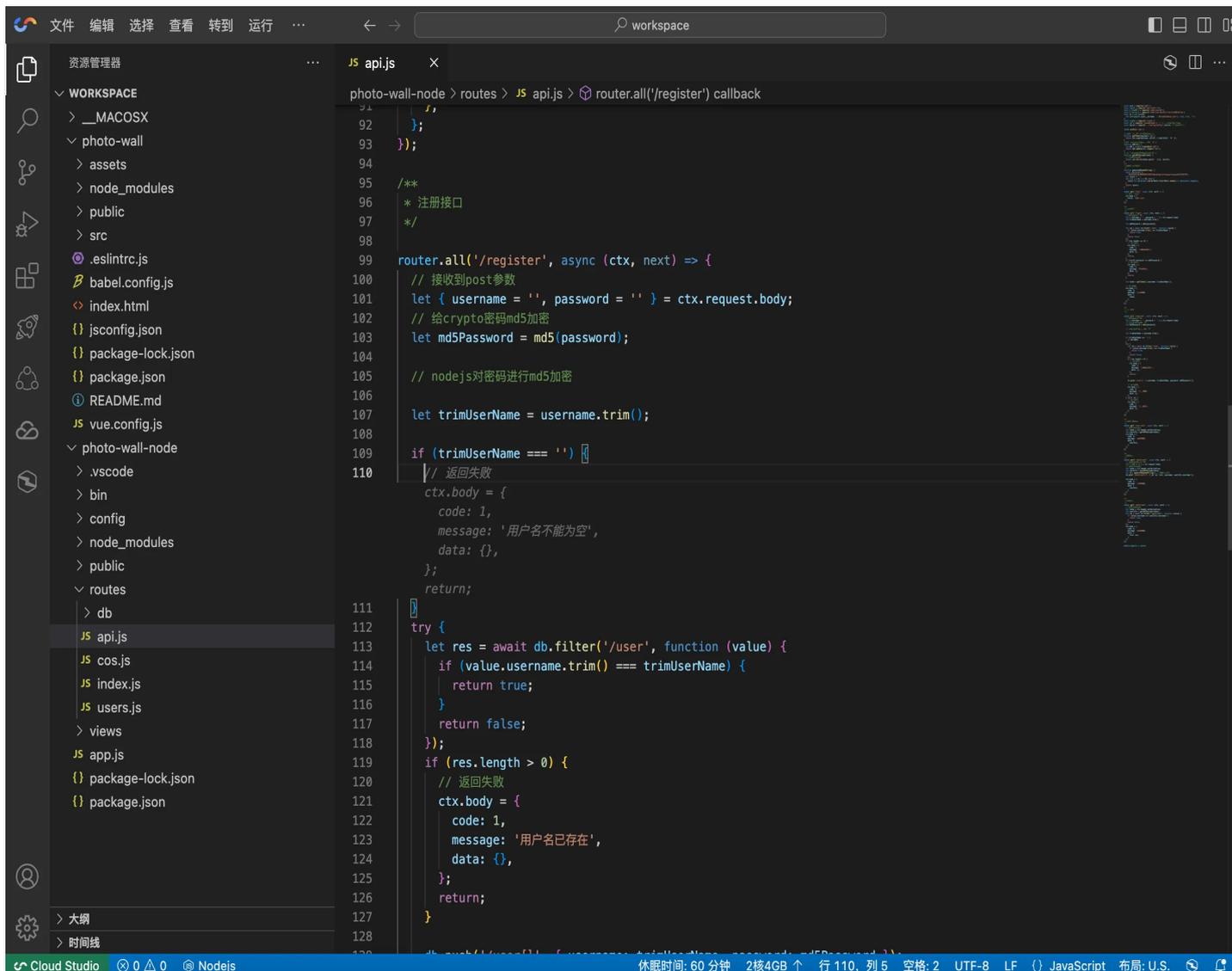
  let trimUserName = username.trim();

  if (trimUserName === '') {
    ctx.body = {
      code: 1,
      message: '用户名不能为空',
      data: {},
    };
    return;
  }

  try {
    let res = await db.filter('/user', function (value) {
      if (value.username.trim() === trimUserName) {
        return true;
      }
      return false;
    });
  } catch (err) {
    // 返回失败
    ctx.body = {
      code: 1,
      message: '用户名已存在',
      data: {},
    };
    return;
  }

  db.push('/user[]', { username: trimUserName, password: md5Password });
}
```

此处对代码逻辑进行调整。例如针对109行做一些代码调整，增加兜底逻辑。

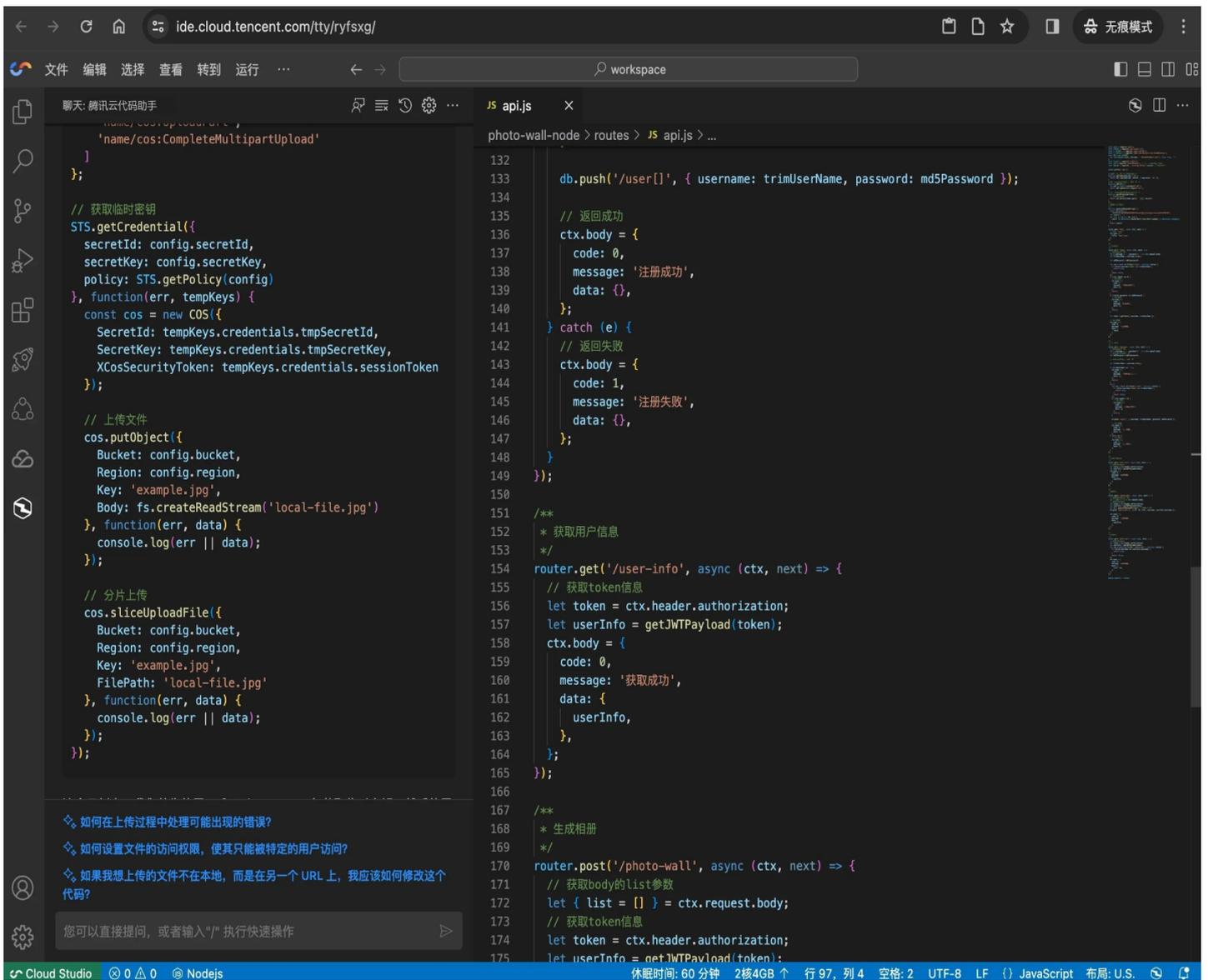


```
photo-wall-node > routes > JS api.js > router.all('/register') callback
92   };
93   });
94
95   /**
96    * 注册接口
97    */
98
99   router.all('/register', async (ctx, next) => {
100   // 接收到post参数
101   let { username = '', password = '' } = ctx.request.body;
102   // 给crypto密码md5加密
103   let md5Password = md5(password);
104
105   // nodejs对密码进行md5加密
106
107   let trimUserName = username.trim();
108
109   if (trimUserName === '') {
110     // 返回失败
111     ctx.body = {
112       code: 1,
113       message: '用户名不能为空',
114       data: {},
115     };
116     return;
117   }
118   try {
119     let res = await db.filter('user', function (value) {
120       if (value.username.trim() === trimUserName) {
121         return true;
122       }
123       return false;
124     });
125     if (res.length > 0) {
126       // 返回失败
127       ctx.body = {
128         code: 1,
129         message: '用户名已存在',
130         data: {},
131       };
132       return;
133     }
134   }
135   db.insert({ username: trimUserName, password: md5Password });
136 }
```

## COS 桶

上传的图片需要存放到 COS 桶，接下来向代码助手提问：**用腾讯云的 COS 桶 qcloud-cos-sts，存储图片。通过获取临时密钥，实现图片上传和分片。**





```
name/cos:CompleteMultipartUpload'
  }
};

// 获取临时密钥
STS.getCredential({
  secretId: config.secretId,
  secretKey: config.secretKey,
  policy: STS.getPolicy(config)
}), function(err, tempKeys) {
  const cos = new COS({
    SecretId: tempKeys.credentials.tmpSecretId,
    SecretKey: tempKeys.credentials.tmpSecretKey,
    XCosSecurityToken: tempKeys.credentials.sessionToken
  });

  // 上传文件
  cos.putObject({
    Bucket: config.bucket,
    Region: config.region,
    Key: 'example.jpg',
    Body: fs.createReadStream('local-file.jpg')
  }, function(err, data) {
    console.log(err || data);
  });

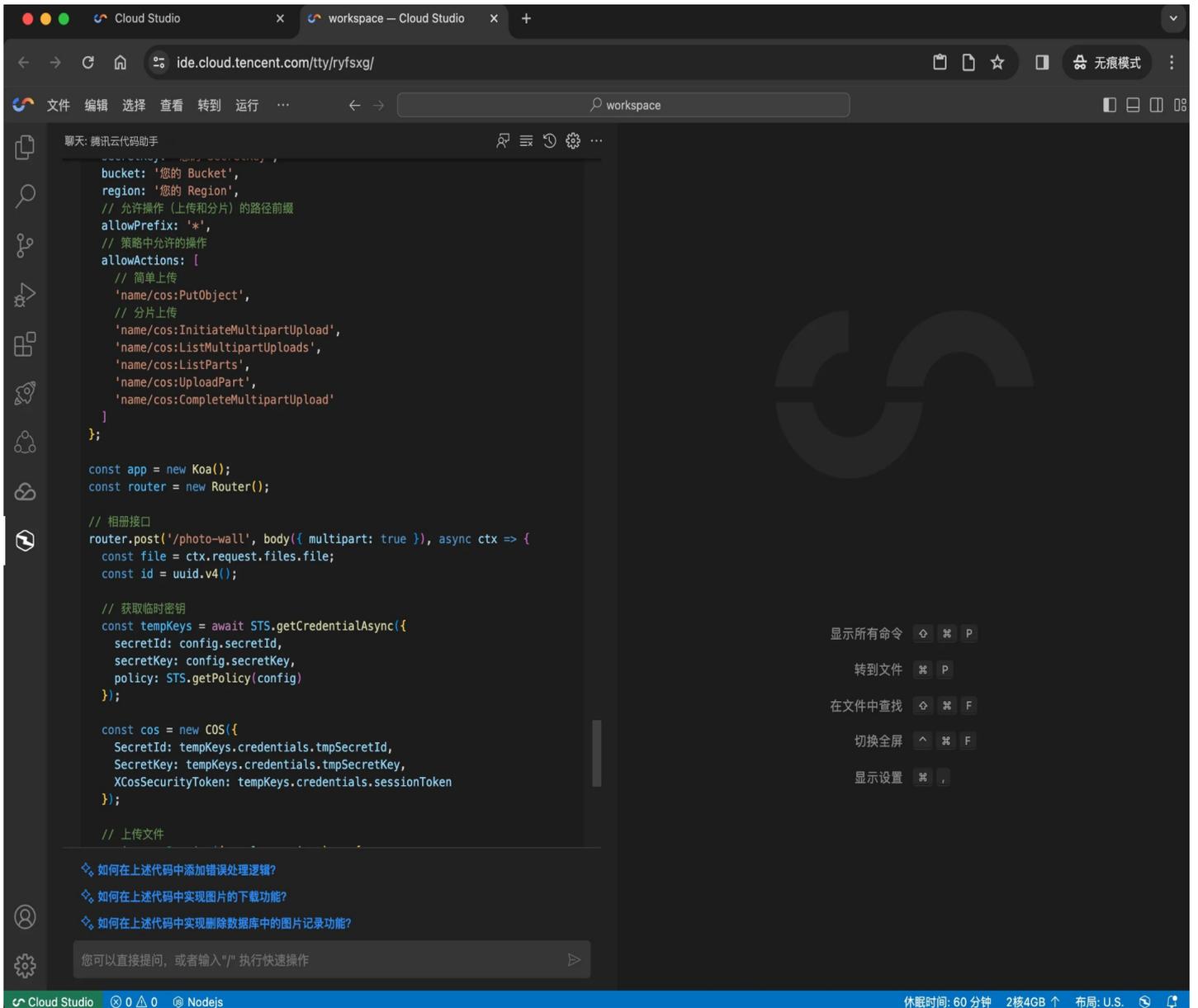
  // 分片上传
  cos.sliceUploadFile({
    Bucket: config.bucket,
    Region: config.region,
    Key: 'example.jpg',
    FilePath: 'local-file.jpg'
  }, function(err, data) {
    console.log(err || data);
  });
});

// 如何在上传过程中处理可能出现的错误?
// 如何设置文件的访问权限, 使其只能被特定的用户访问?
// 如果我想上传的文件不在本地, 而是在另一个 URL 上, 我应该如何修改这个代码?
您可以直接提问, 或者输入"/"执行快速操作

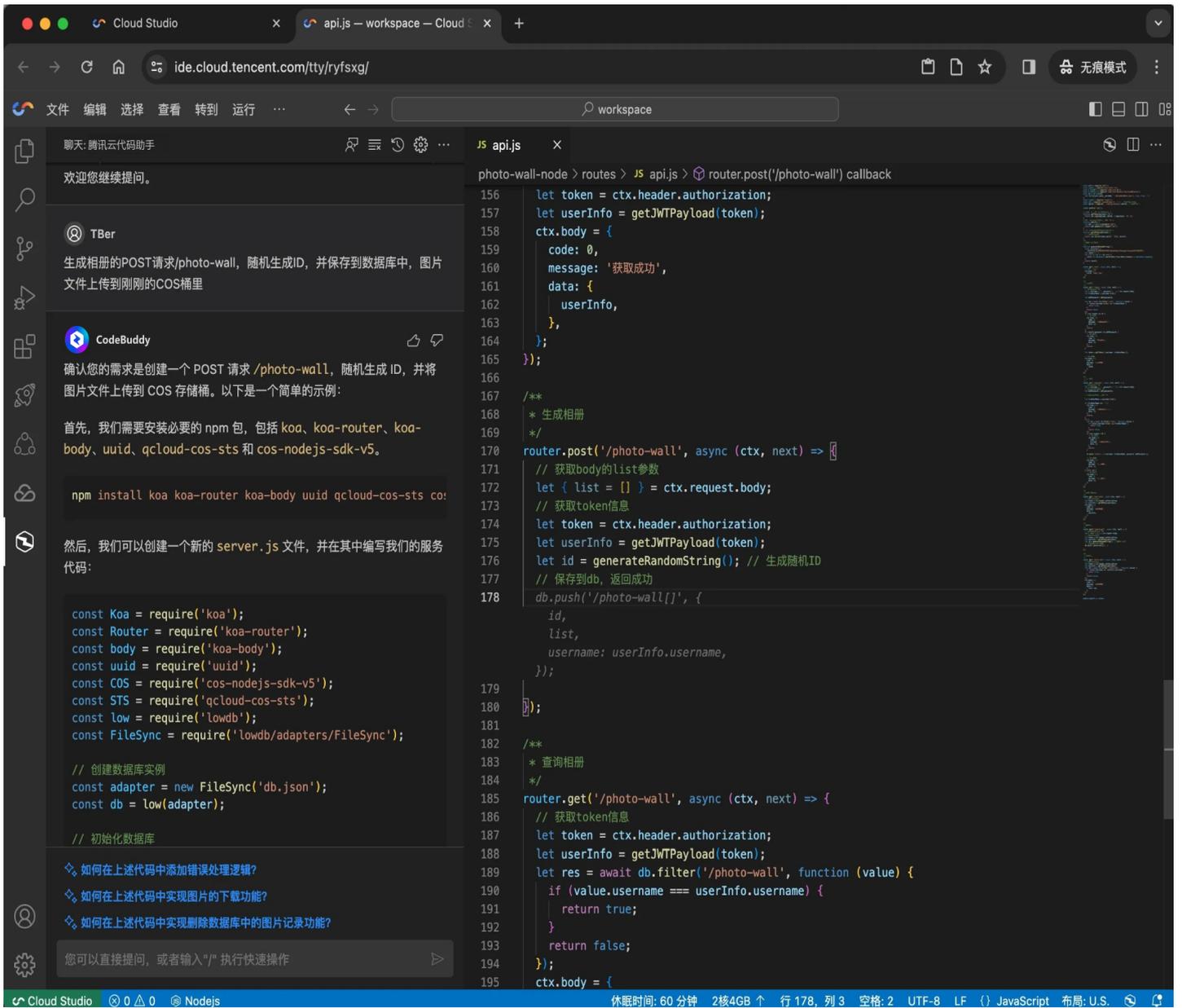
photo-wall-node > routes > JS api.js > ...
132
133 db.push('/user[]', { username: trimUserName, password: md5Password });
134
135 // 返回成功
136 ctx.body = {
137   code: 0,
138   message: '注册成功',
139   data: {},
140 };
141 } catch (e) {
142   // 返回失败
143   ctx.body = {
144     code: 1,
145     message: '注册失败',
146     data: {},
147   };
148 }
149 });
150
151 /**
152  * 获取用户信息
153  */
154 router.get('/user-info', async (ctx, next) => {
155   // 获取token信息
156   let token = ctx.header.authorization;
157   let userInfo = getJWTPayload(token);
158   ctx.body = {
159     code: 0,
160     message: '获取成功',
161     data: {
162       userInfo,
163     },
164   };
165 });
166
167 /**
168  * 生成相册
169  */
170 router.post('/photo-wall', async (ctx, next) => {
171   // 获取body的list参数
172   let { list = [] } = ctx.request.body;
173   // 获取token信息
174   let token = ctx.header.authorization;
175   let userInfo = getJWTPayload(token);
```

## 图片预览服务

输入：生成相册的 POST 请求 /photo-wall，随机生成 ID，并保存到数据库中，图片文件上传到刚刚的 COS 桶里。

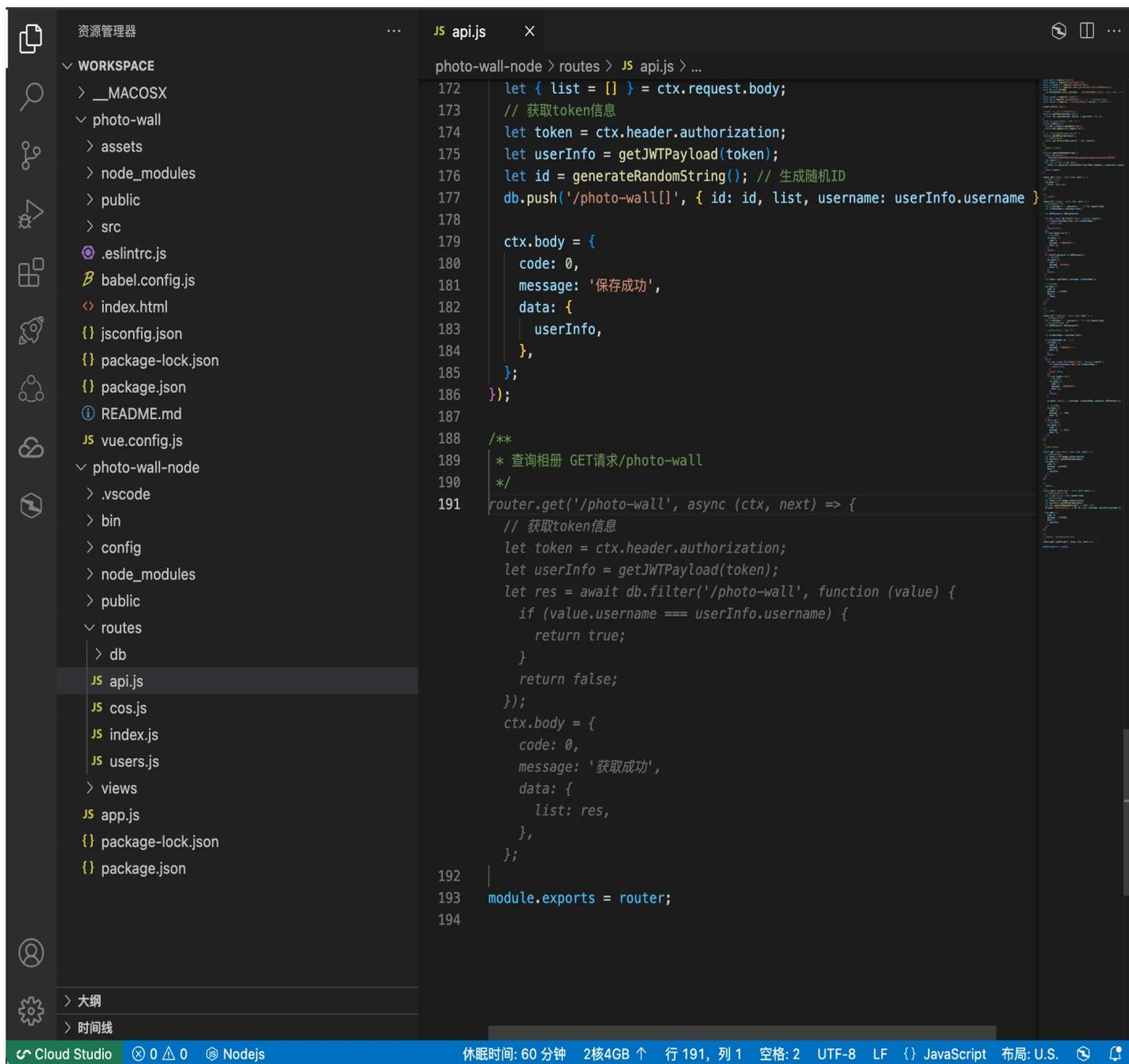


找到核心代码逻辑，复制到编辑器中，触发补全进行代码微调。



## 图片相册列表服务

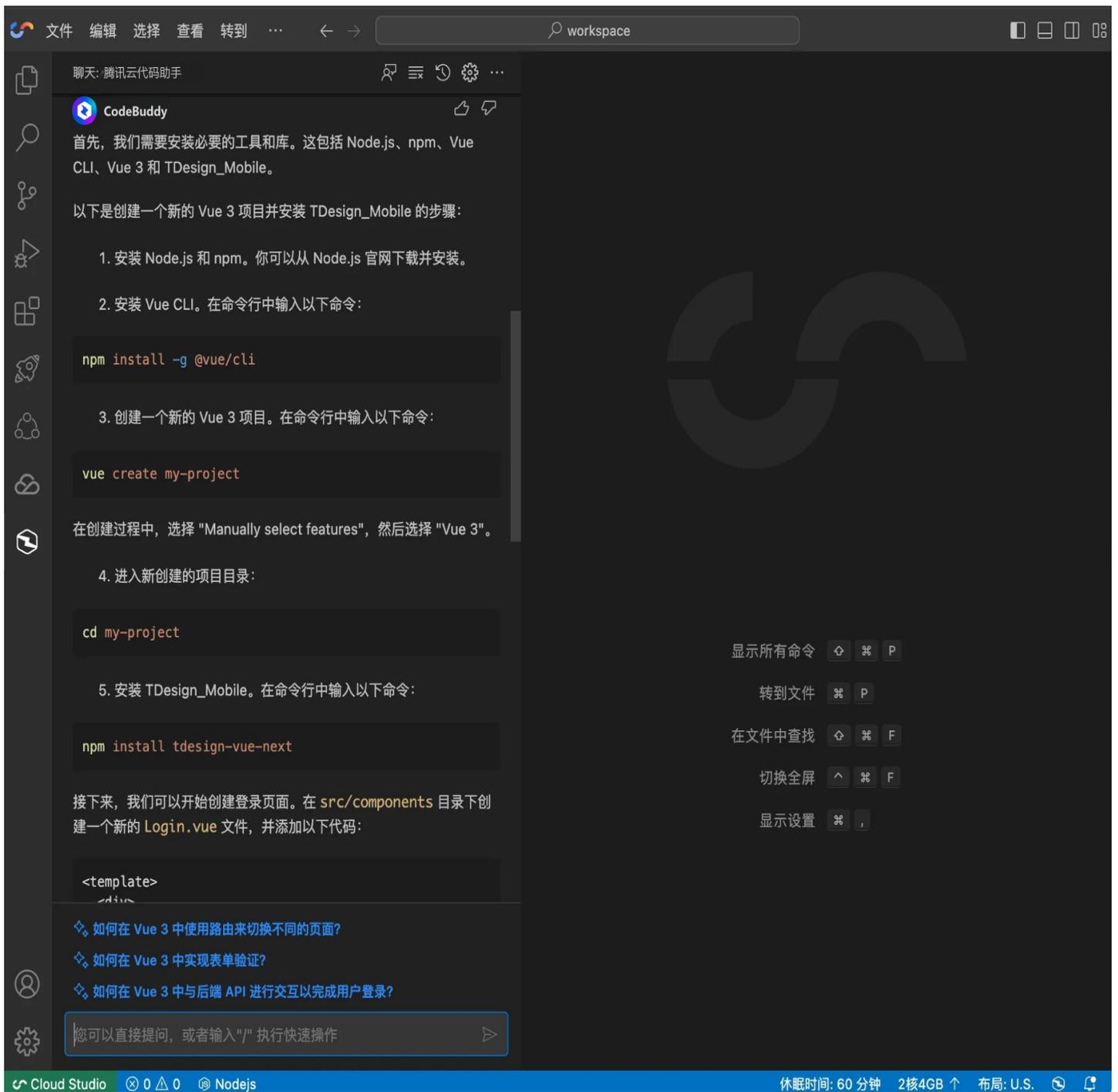
输入代码注释：查询相册 GET 请求/photo-wall，根据上文的 POST 请求，猜出 GET 请求的编码。在此基础上，进行代码补全的微调就快速完成了相似的列表服务。



```
photo-wall-node > routes > JS api.js > ...
172 let { list = [] } = ctx.request.body;
173 // 获取token信息
174 let token = ctx.header.authorization;
175 let userInfo = getJWTPayload(token);
176 let id = generateRandomString(); // 生成随机ID
177 db.push('/photo-wall[]', { id: id, list, username: userInfo.username });
178
179 ctx.body = {
180   code: 0,
181   message: '保存成功',
182   data: {
183     userInfo,
184   },
185 };
186 });
187
188 /**
189  * 查询相册 GET请求/photo-wall
190  */
191 router.get('/photo-wall', async (ctx, next) => {
192   // 获取token信息
193   let token = ctx.header.authorization;
194   let userInfo = getJWTPayload(token);
195   let res = await db.filter('/photo-wall', function (value) {
196     if (value.username === userInfo.username) {
197       return true;
198     }
199     return false;
200   });
201   ctx.body = {
202     code: 0,
203     message: '获取成功',
204     data: {
205       list: res,
206     },
207   };
208 };
209
210 module.exports = router;
211
```

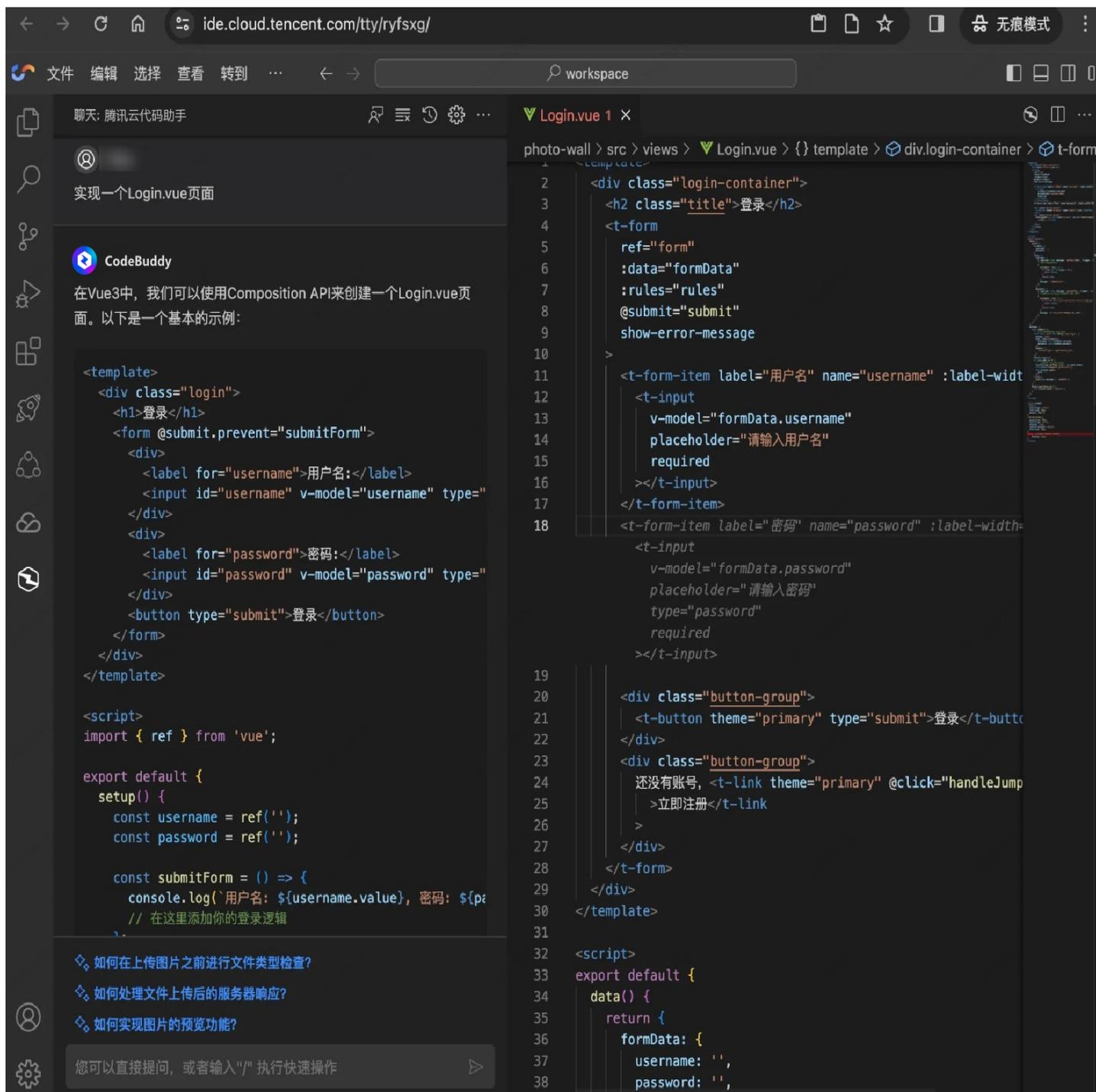
## 前端应用

前端应用采用 [Koa2+TDesign\\_Mobile 组件库](#)。



## Vue 页面

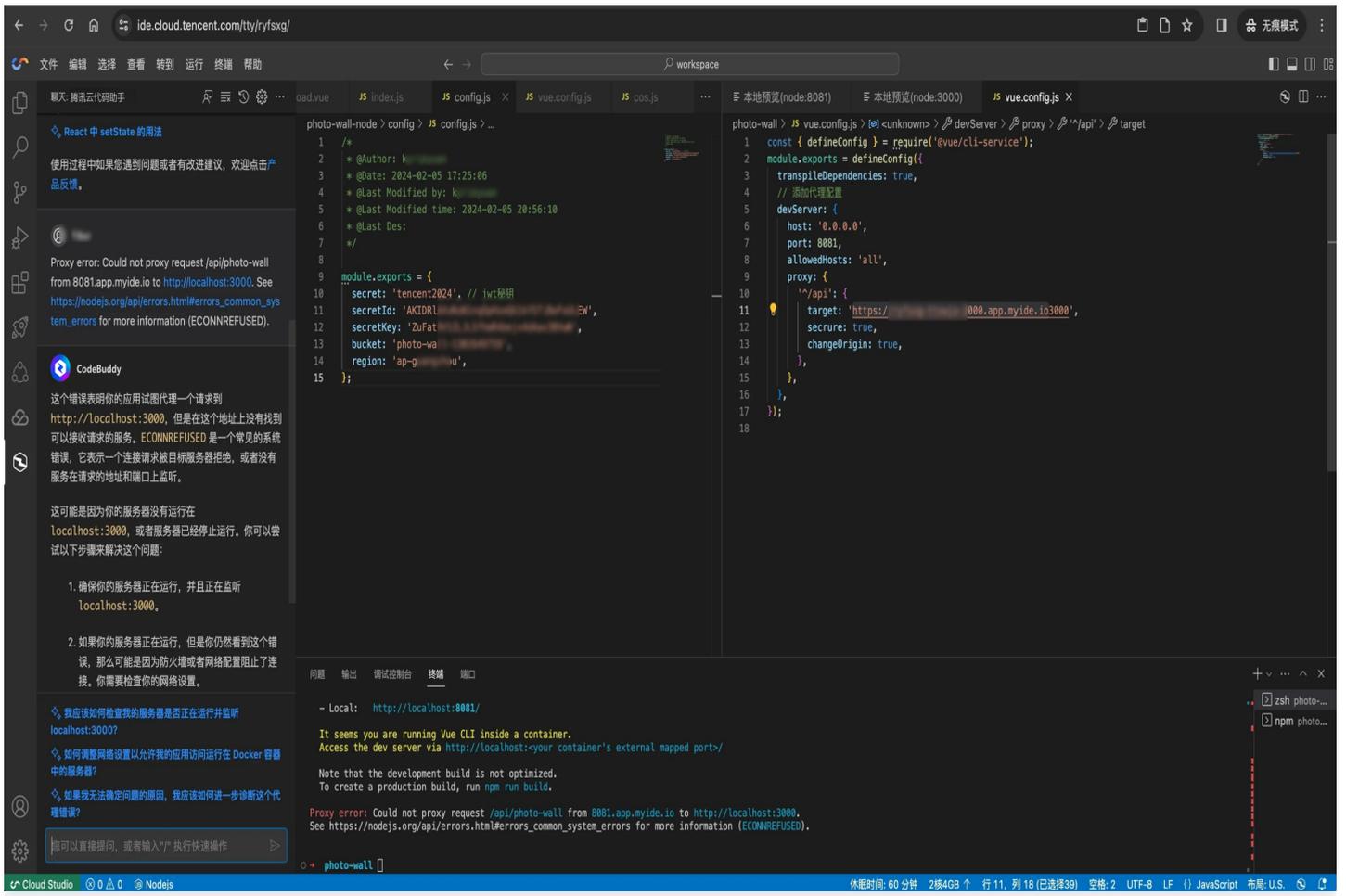
创建对应几个 Vue 文件, 如图, 通过对话实现一个 `Login.vue` 页面先去生成个大概。然后通过插入代码之后, 再进行代码补全微调, 从而生成自己想要的代码逻辑。



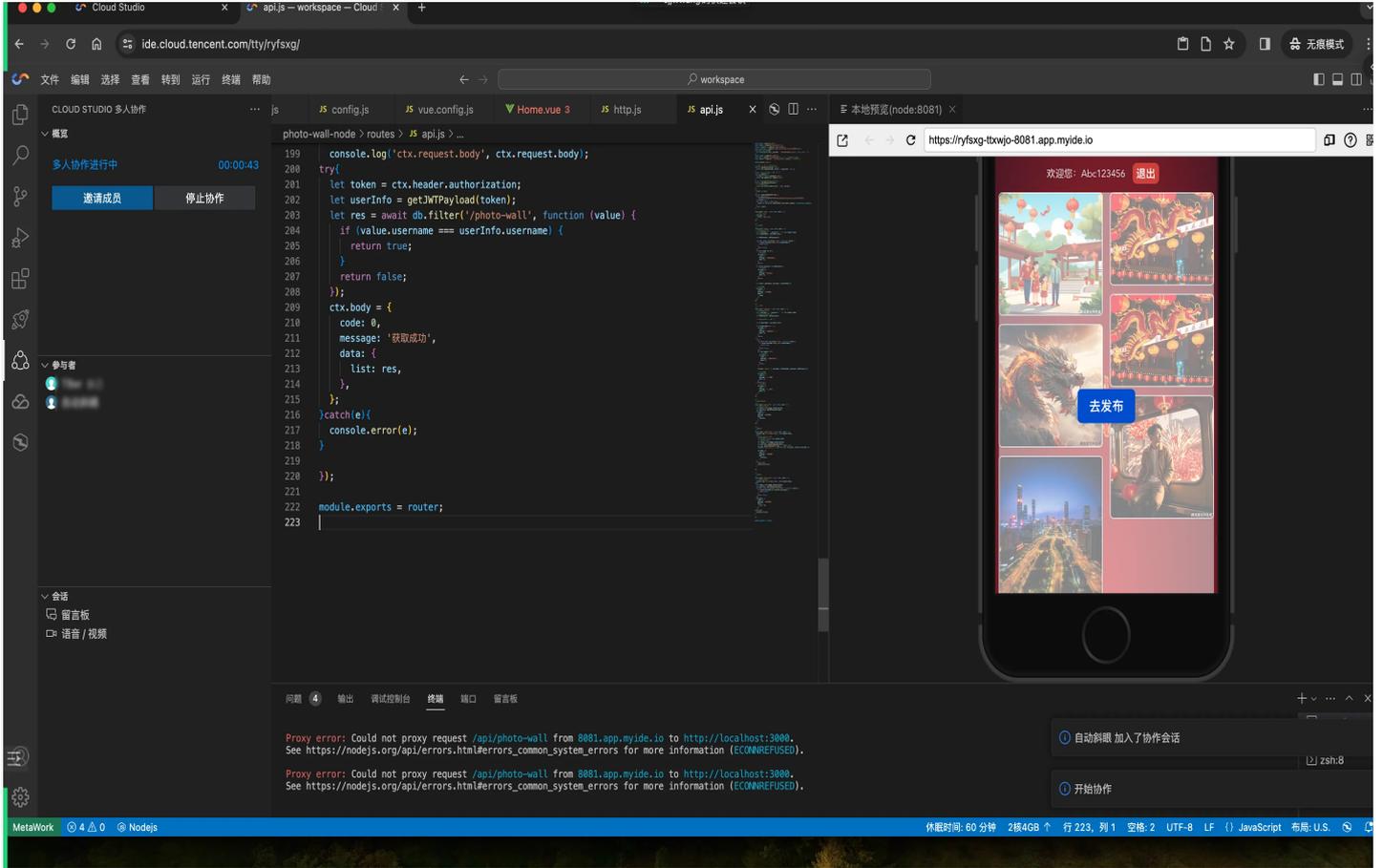
操作方式类似, 由于篇幅有限, 这里不一一截取。通过组合拳, 对话+补全, 加上多文件能力, 使得前后端的开发工作效率得到提升加速。

## 最后的预览效果

在 Cloud IDE 里遇到了问题, 例如转发问题和 AllowHost 问题, 可以通过代码助手对话解决。



也通过了 Cloud Studio 的协作能力排查错误：



最后成功，效果演示视频如下：

[观看视频](#)

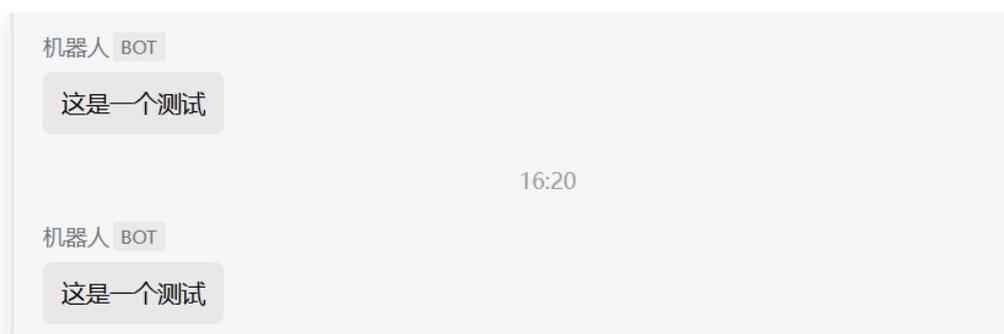
# 快速制作企微提醒机器人

最近更新时间：2025-04-14 09:54:23

## 引言

制作企业微信的提醒机器人是工作中经常遇到的场景，借助代码助手，可以将制作成本大大降低，轻松完成接入。本文将制作一个企微提醒机器人作为示例，为您演示如何借助代码助手为开发工作提效，制作好的企微提醒机器人效果如下：

- 文字消息：



- Markdown 消息：



## 准备腾讯云代码助手

### 介绍

打开 [腾讯云代码助手介绍页](#)，通过介绍可以知道腾讯云代码助手是基于混元代码大模型构建的，该模型具备深度理解和生成代码的能力。在混元代码大模型的支持下，腾讯云代码助手能够提供以下核心功能：

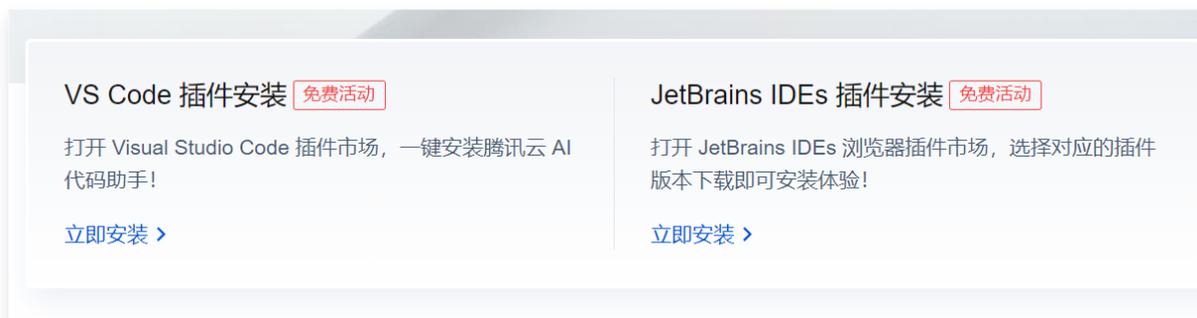
- 技术对话：能够理解和回答与编程、技术架构相关的问题，提供技术咨询和解决方案。
- 代码补全：根据已有代码上下文，智能补全代码片段，提高编程效率。
- 代码诊断：自动检测代码中的错误和潜在问题，提供修复建议，提升代码质量。

- 代码优化：分析代码并提出优化建议，帮助开发者提高代码性能和可读性。

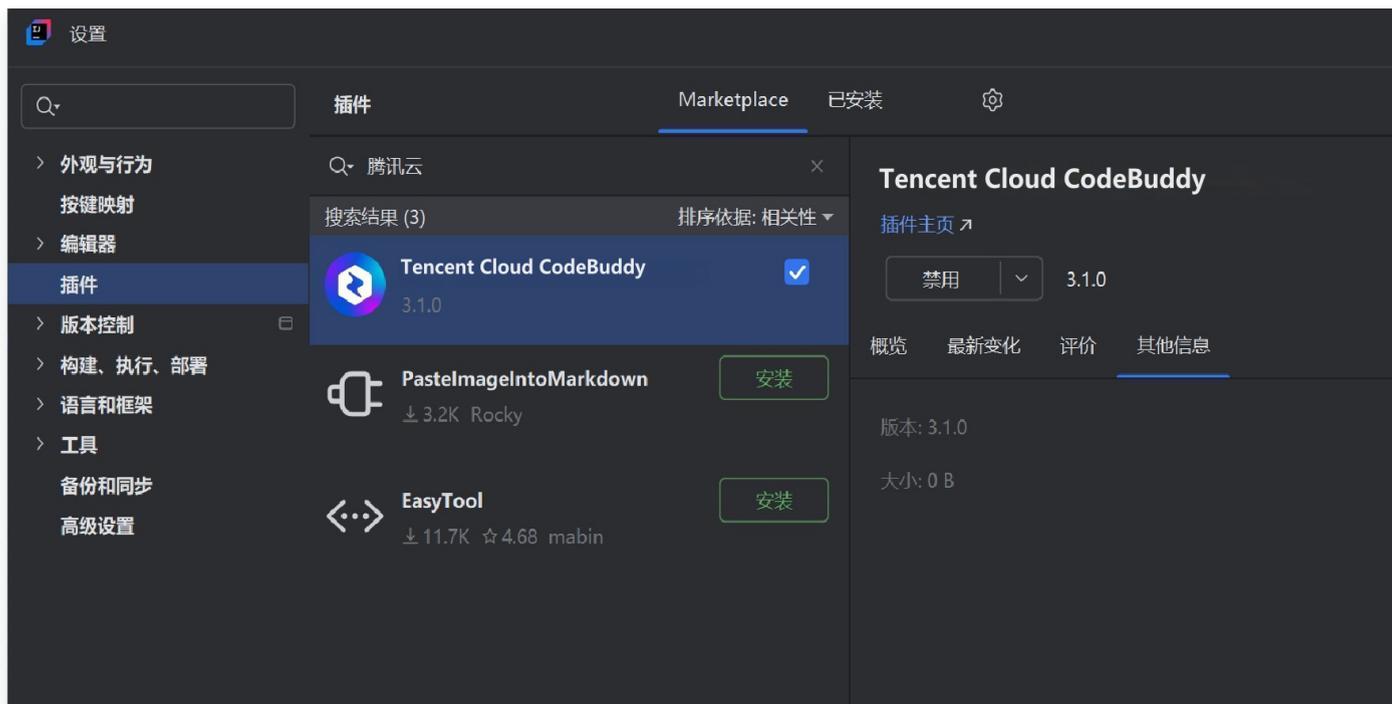
通过这些功能，腾讯云代码助手旨在为开发者提供全方位的编程辅助，从代码编写到调试优化，全面提升开发效率和代码质量。它是基于混元代码大模型的智能编程助手，旨在让编程变得更加高效、智能和轻松。

## 下载和安装

1. 在编程工具里搜索或单击页面上**立即安装**，跳转到插件详情页进行安装，本次教程以 JetBrains 为例进行安装。

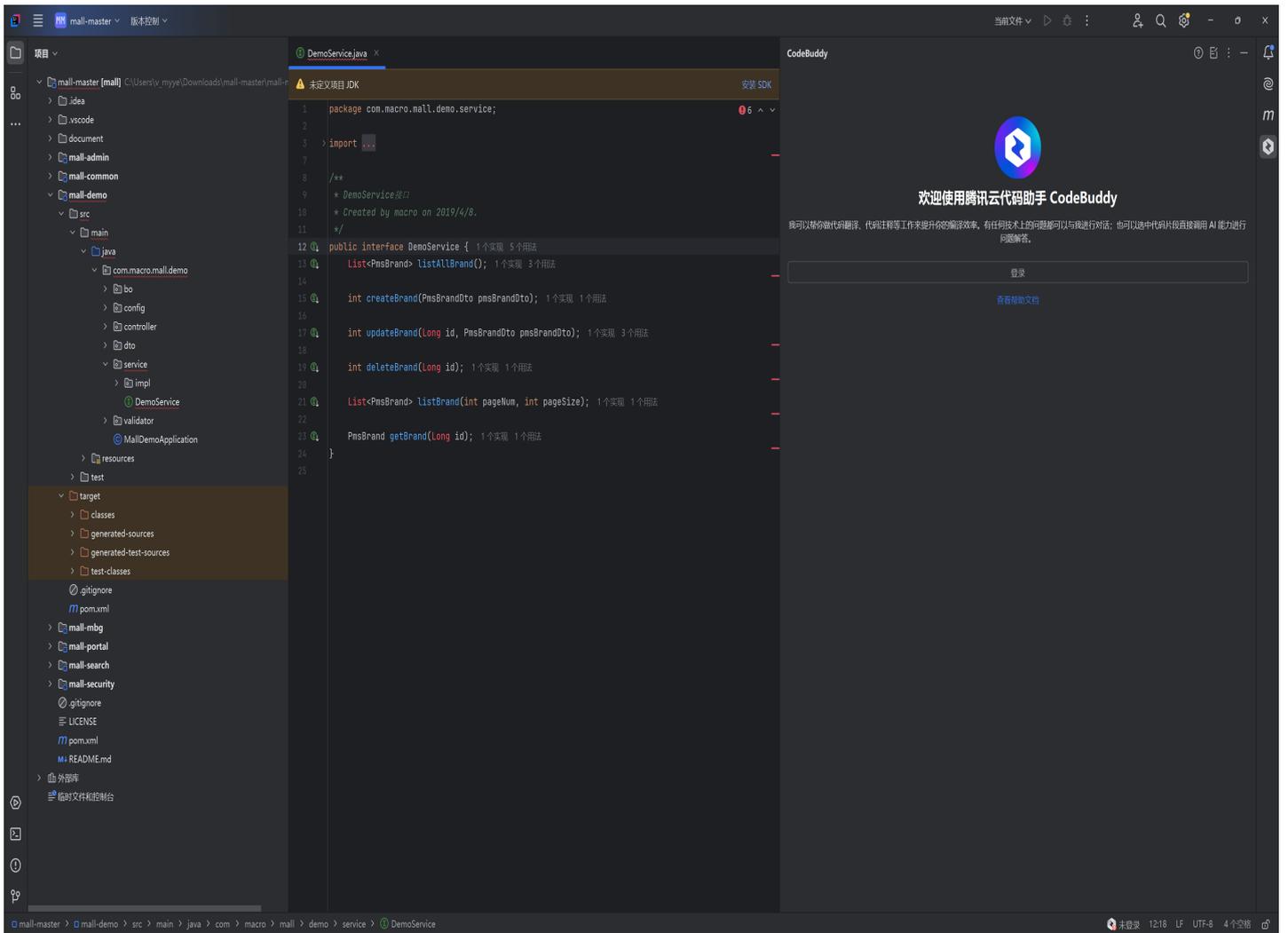


2. 打开 JetBrains 软件，例如我使用的是 IntelliJ IDEA，在设置里找到插件，然后搜索腾讯云代码助手或 Tencent Cloud CodeBuddy 然后进行安装，安装完成后，需要重启。

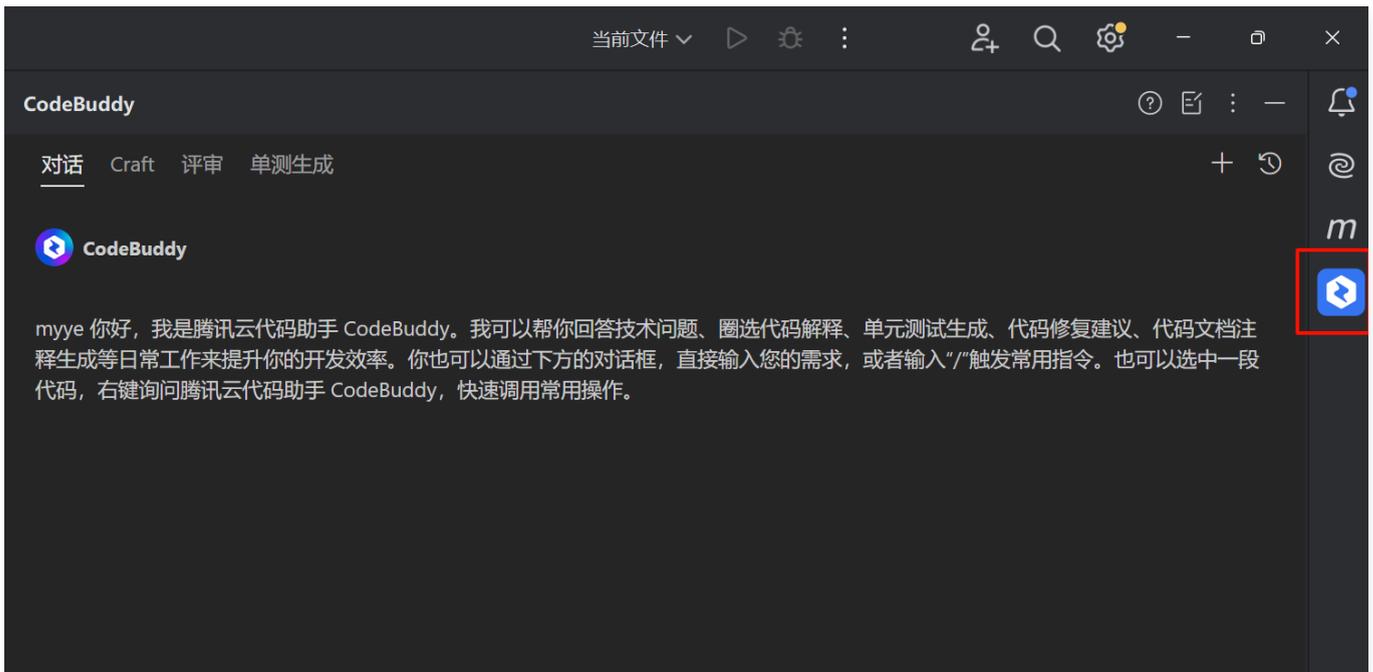


## 登录和使用

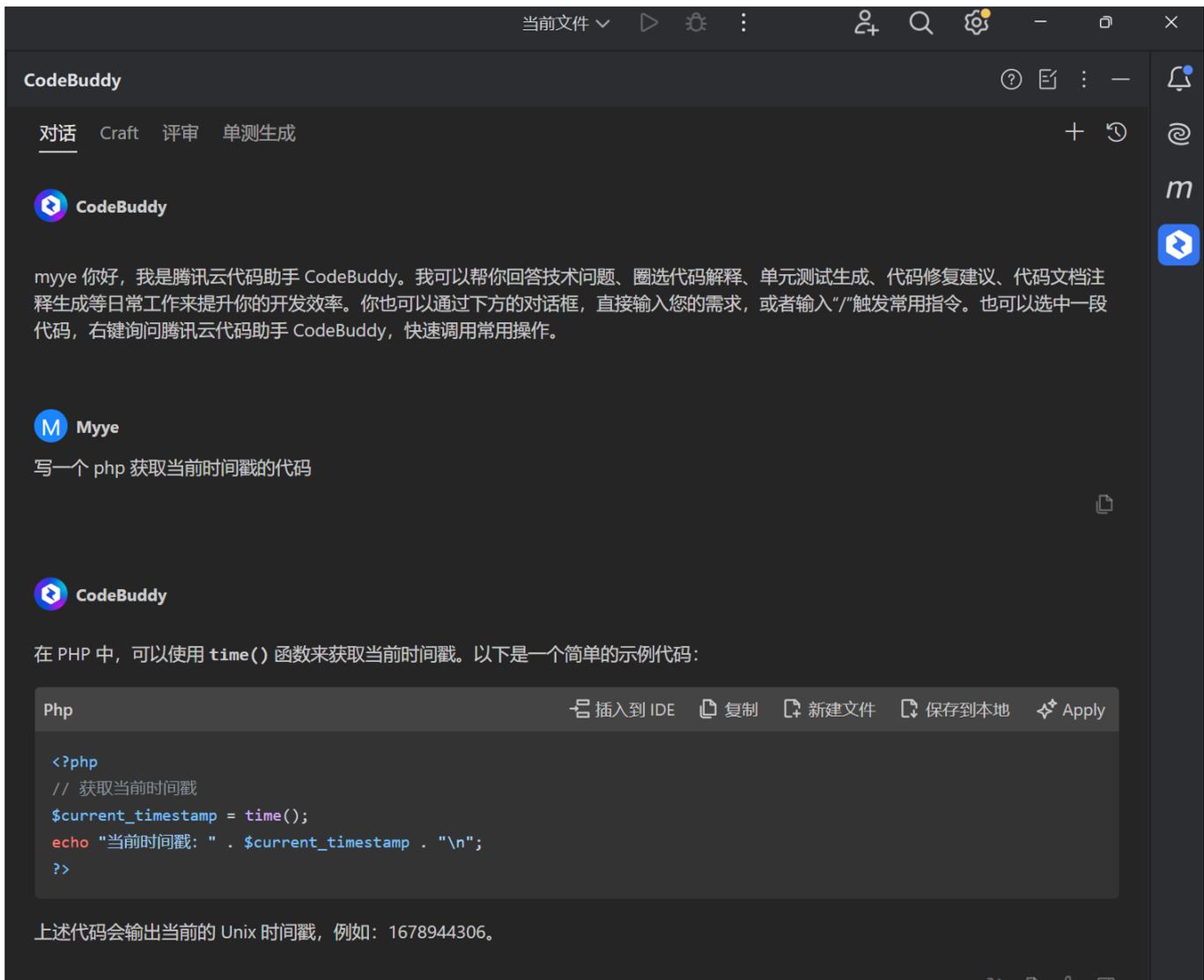
重启完 IntelliJ IDEA 后，就会进入腾讯云代码助手提示页，根据这个提示页面，进行登录。



然后单击侧边栏，进入助手聊天界面。



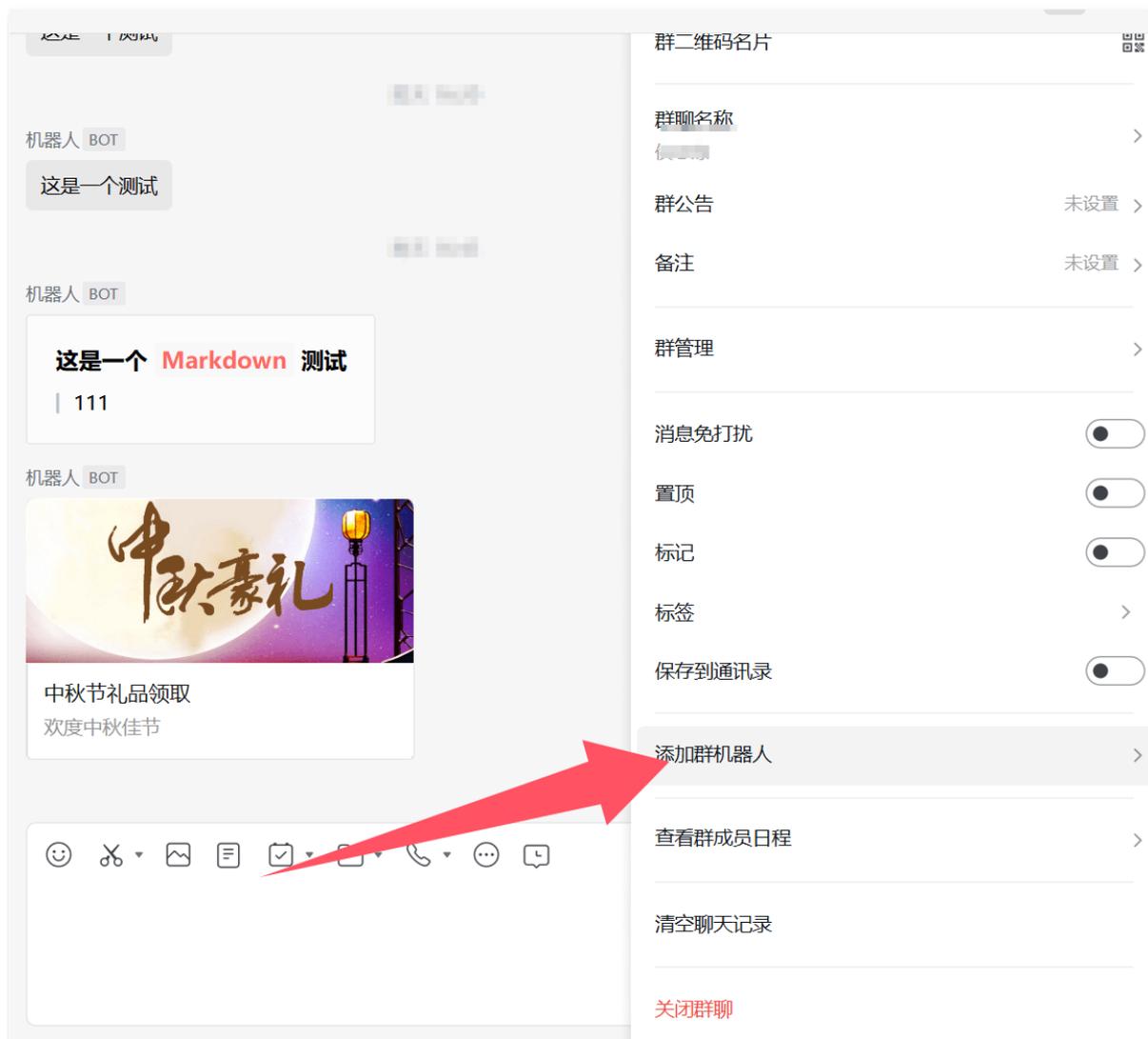
输入您想让代码帮您完成的事项，例如“写一个 php 获取当前时间戳的代码”。



## 进入企业微信，创建机器人

### 创建机器人

进入企微信群聊后，单击右上角三个点进入群操作页面，然后添加群机器人，根据提示创建机器人。



单击新创建一个机器人。



创建完成后，复制 Webhook 地址。



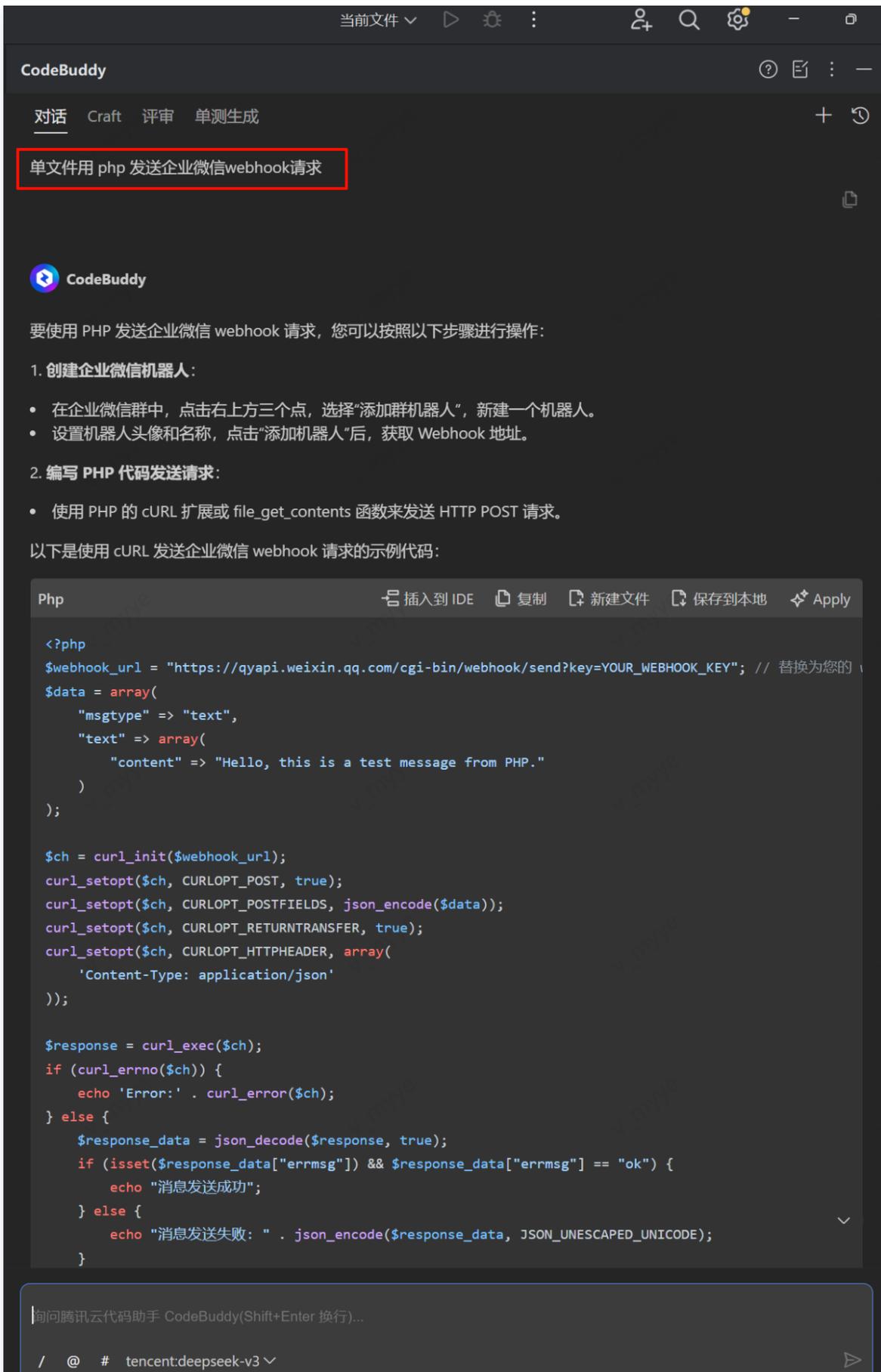
## 复制机器人参数

复制好上面的链接后，将 `send?` 后面的参数截取下来，然后单独保存好，后面的代码对接就需要用到这个参数。

## 对话 AI 生成机器人运行代码

用一句话让腾讯云代码助手帮您生成代码。

1. 打开助手的聊天窗口，然后输入我们需要实现的功能，例如需要它帮我写一个 php 的企业微信群机器人文本消息代码，那么可以这样对它说。



CodeBuddy

对话 Craft 评审 单测生成

单文件用 php 发送企业微信webhook请求

CodeBuddy

要使用 PHP 发送企业微信 webhook 请求，您可以按照以下步骤进行操作：

- 创建企业微信机器人：**
  - 在企业微信群中，点击右上角三个点，选择“添加群机器人”，新建一个机器人。
  - 设置机器人头像和名称，点击“添加机器人”后，获取 Webhook 地址。
- 编写 PHP 代码发送请求：**
  - 使用 PHP 的 cURL 扩展或 `file_get_contents` 函数来发送 HTTP POST 请求。

以下是使用 cURL 发送企业微信 webhook 请求的示例代码：

```
Php 插入到 IDE 复制 新建文件 保存到本地 Apply

<?php
$webhook_url = "https://qyapi.weixin.qq.com/cgi-bin/webhook/send?key=YOUR_WEBHOOK_KEY"; // 替换为您的
$data = array(
    "msgtype" => "text",
    "text" => array(
        "content" => "Hello, this is a test message from PHP."
    )
);

$ch = curl_init($webhook_url);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($data));
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, array(
    'Content-Type: application/json'
));

$response = curl_exec($ch);
if (curl_errno($ch)) {
    echo 'Error:' . curl_error($ch);
} else {
    $response_data = json_decode($response, true);
    if (isset($response_data["errmsg"]) && $response_data["errmsg"] == "ok") {
        echo "消息发送成功";
    } else {
        echo "消息发送失败: " . json_encode($response_data, JSON_UNESCAPED_UNICODE);
    }
}
```

询问腾讯云代码助手 CodeBuddy(Shift+Enter 换行)...

/ @ # tencent:deepseek-v3

- 腾讯云代码助手理解了我的需求，然后回复了上面的代码，我们将这段代码复制到一个 php 文件里，然后将刚才创建机器人时得到 webhook 地址后面的参数复制到助手生成代码上，也就是下面这一段。

```
$webhook_url = "https://qyapi.weixin.qq.com/cgi-bin/webhook/send?
key=YOUR_WEBHOOK_KEY"; // 替换为您的 webhook 地址
```

- 将 YOUR\_KEY 改成我们提取的参数，然后需要注意的地方是，复制时看一下有没有空格，如果有需要删除空格。

```
$webhook_url = "https://qyapi.weixin.qq.com/cgi-bin/webhook/send?
key=86c70996-fa50-406c-9227-72c21d9ef1c9"; // 替换为您的 webhook 地址
```

- 然后进行运行代码（此步骤是已经安装了 php 运行环境，如果没有安装的可以前往百度搜索 php 环境安装，下载宝塔或 PHP study 工具）。

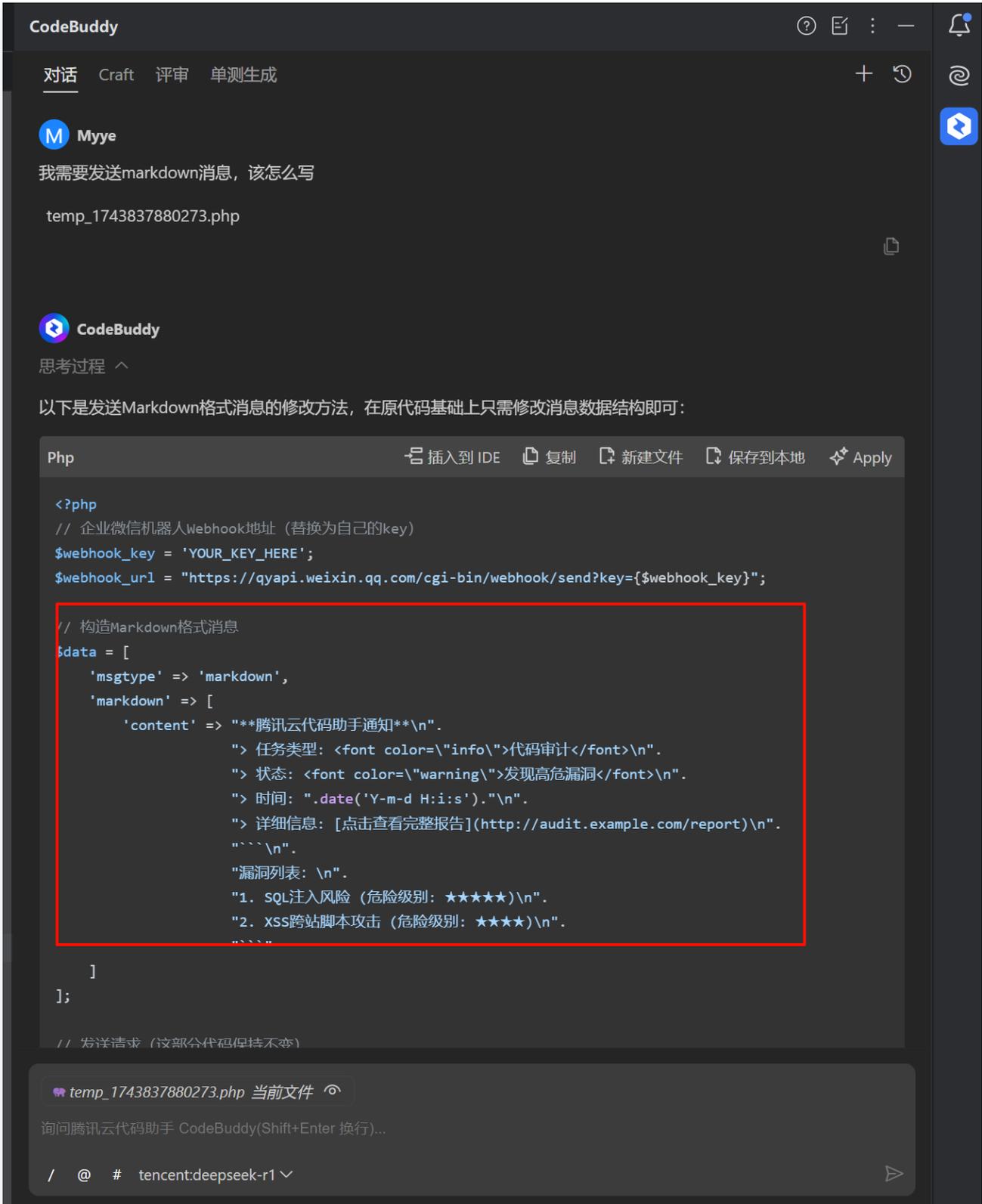
浏览器运行一下代码。



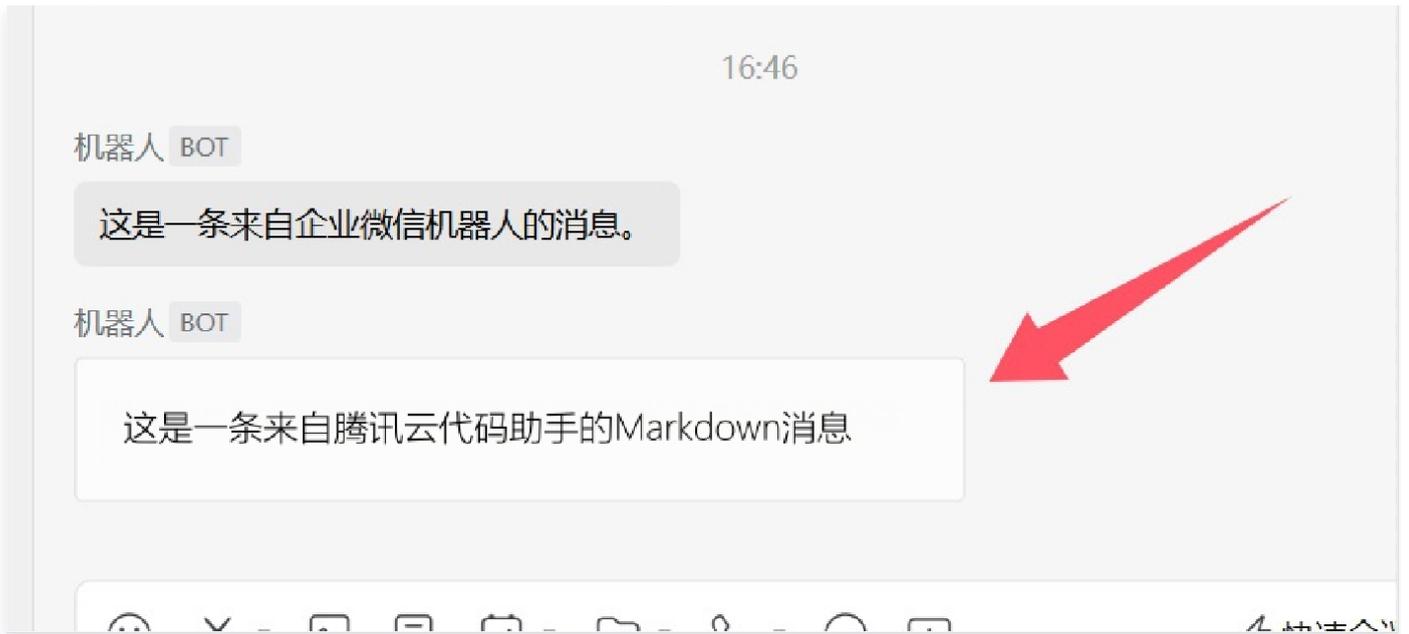
运行发送成功。



- 同样如果您需要发送 Markdown 消息的话，可以直接在助手的聊天窗口里给助手进行提问。应用代码时把 content 改成自己想要的的内容即可。



发送 Markdown 消息效果如下:



6. 如果您还需要其他类型的消息，请参见 [企业微信群机器人](#)，然后将文档内容复制给助手，让助手学习一下，告诉您其它几个类型的消息的 php 代码该怎么写。
7. 下面是以上的两种消息类型全部代码，需要哪个类型的消息，直接关闭注释即可，也就是代码前面的斜杠 //。

```
<?php
// 设置企业微信 webhook 地址
$webhookUrl = 'https://qyapi.weixin.qq.com/cgi-bin/webhook/send?
key=YOUR_KEY';

// 设置要发送的消息内容
// $message = [
//     'msgtype' => 'markdown',
//     'markdown' => [
//         'content' => '这是一条来自腾讯云代码助手的 **Markdown** 消息'
//     ]
// ];

//
//
// $message = [
//     'msgtype' => 'text',
//     'text' => [
//         'content' => '这是一条来自腾讯云代码助手的消息'
//     ]
// ];

// 使用 cURL 发送 POST 请求
$ch = curl_init();
```

```
curl_setopt($ch, CURLOPT_URL, $webhookUrl);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($message));
$response = curl_exec($ch);

// 检查请求是否成功
if (curl_errno($ch)) {
    echo '请求失败: ' . curl_error($ch);
} else {
    $status_code = curl_getinfo($ch, CURLINFO_HTTP_CODE);
    if ($status_code == 200) {
        echo '请求成功';
    } else {
        echo '请求失败, 状态码: ' . $status_code;
    }
}

// 关闭 cURL 会话
curl_close($ch);
?>
```

## 总结

写完这篇文章，其实大部分时间都是截图和文字介绍，编程代码占用了很少的一部分时间，而以前在编写代码时都需要用一半的时间去写代码，现如今，只需简洁地表达我们的需求，代码助手就能迅速生成高质量的代码，这一转变不仅节省了大量时间，更释放了我们的创造力，使我们能够专注于更高级别的设计和创新工作。腾讯云代码助手的智能代码生成和优化功能，显著提升了代码质量，使代码不仅功能完善，而且结构清晰、性能优越。

在个人技能层面，与腾讯云代码助手的互动成为了学习新技术和解决复杂问题的有效途径。它就像一位耐心的导师，通过实践引导我们深入理解编程概念，加速了技能的提升和专业知识的积累。无论是新手还是有经验的开发者，都能从中受益匪浅。

可以想象得到腾讯云代码助手预示着一个更加智能、高效和创新的编程时代。随着 AI 技术的不断进步，所以期待它未来在代码生成、优化、错误检测等方面发挥更大的作用，甚至在项目管理和团队协作中扮演更重要的角色。想象一下，未来的工作流程中，代码助手能够自动分析项目需求，生成初步代码框架，甚至自动进行代码审查和优化，这将极大提升开发效率，促进团队创新。

# 自定义知识库实践：智能体框架 Swarm

最近更新时间：2025-04-14 09:54:23

腾讯云代码助手提供了自定义知识库的功能。知识库作为 RAG 技术的数据源，能够为 RAG 提供额外的信息输入，加上 RAG 技术先进的信息检索和自然语言处理能力，能够让生成的内容具备高度的准确性和相关性，显著提高回答的质量，增强大模型生成的能力。接下来，让我们借助腾讯云代码助手的自定义知识库的功能，为您详细解析 Swarm 的源代码，以便您迅速掌握其设计理念和实现细节，并可以快速基于此能力进行企业内的实战开发。

## 对 Swarm 的介绍

Swarm 是 OpenAI 发布的一个实验性的智能体开发框架，被设计为一个轻量级的多智能体开发框架，主要针对需要多个智能体协同工作的场景，例如复杂任务的分配和多智能体间的合作等。它具有轻量级、无状态运行、高度可控、易于测试和迭代等特点。



## 对 Swarm 的应用

### 准备自定义知识库

对于首次使用的新用户，需要如下3步创建企业（对于已经创建了企业的用户，直接登录 [企业后台管理](#) 并跳到第3步即可）。

1. 按向导 [创建企业](#)，此处以创建旗舰版为例，如需了解其他版本请参见 [版本说明](#)。

< 返回

### 旗舰版

1 统一坐席管理  
2 企业研效看板

[了解更多](#)

免费创建

### 专享版

1 企业专属域名  
2 专享 VPC 部署  
3 企业研效看板

[了解更多](#)

立即开通

元/人/月

## 2. 创建成功后，您就是企业管理员。

腾讯云代码助手

组织概览

快速指引

**01 邀请成员**

生成邀请链接或者通过分享二维码，邀请成员加入企业

[邀请成员](#)

**02 审核并授权**

管理员审批开启后，成员需要通过管理员审核才能完成授权

[插件授权](#)

**03 登录插件**

企业成员下载对应的 IDE 插件，在插件端登录企业账号体验编辑能力

[插件登录指南](#)

帮助中心

帮助文档

快速入门

插件使用指南

已授权开发者人数	剩余许可证数	总许可证数
1	99	100

今日概览

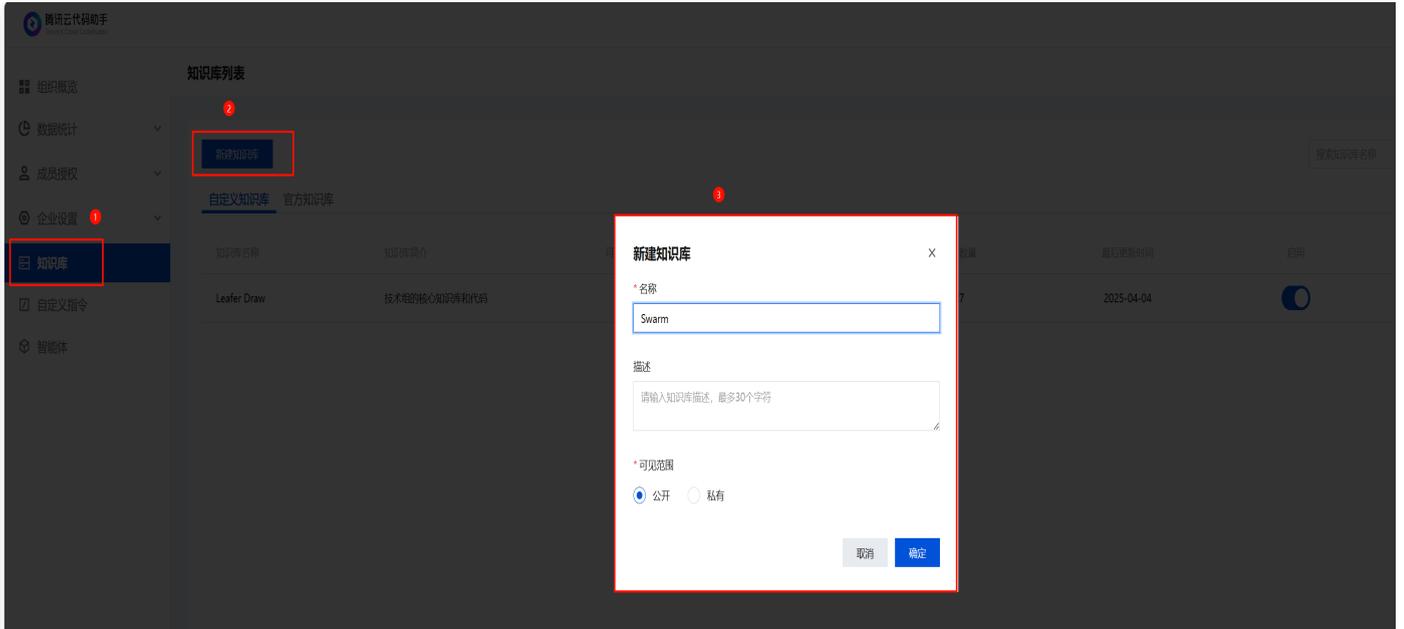
活跃开发者人数	对话次数	补全采纳次数
6	6	4
补全采纳行数	补全采纳字符数	补全采纳率 (按行)
4	96	15.38%

插件安装

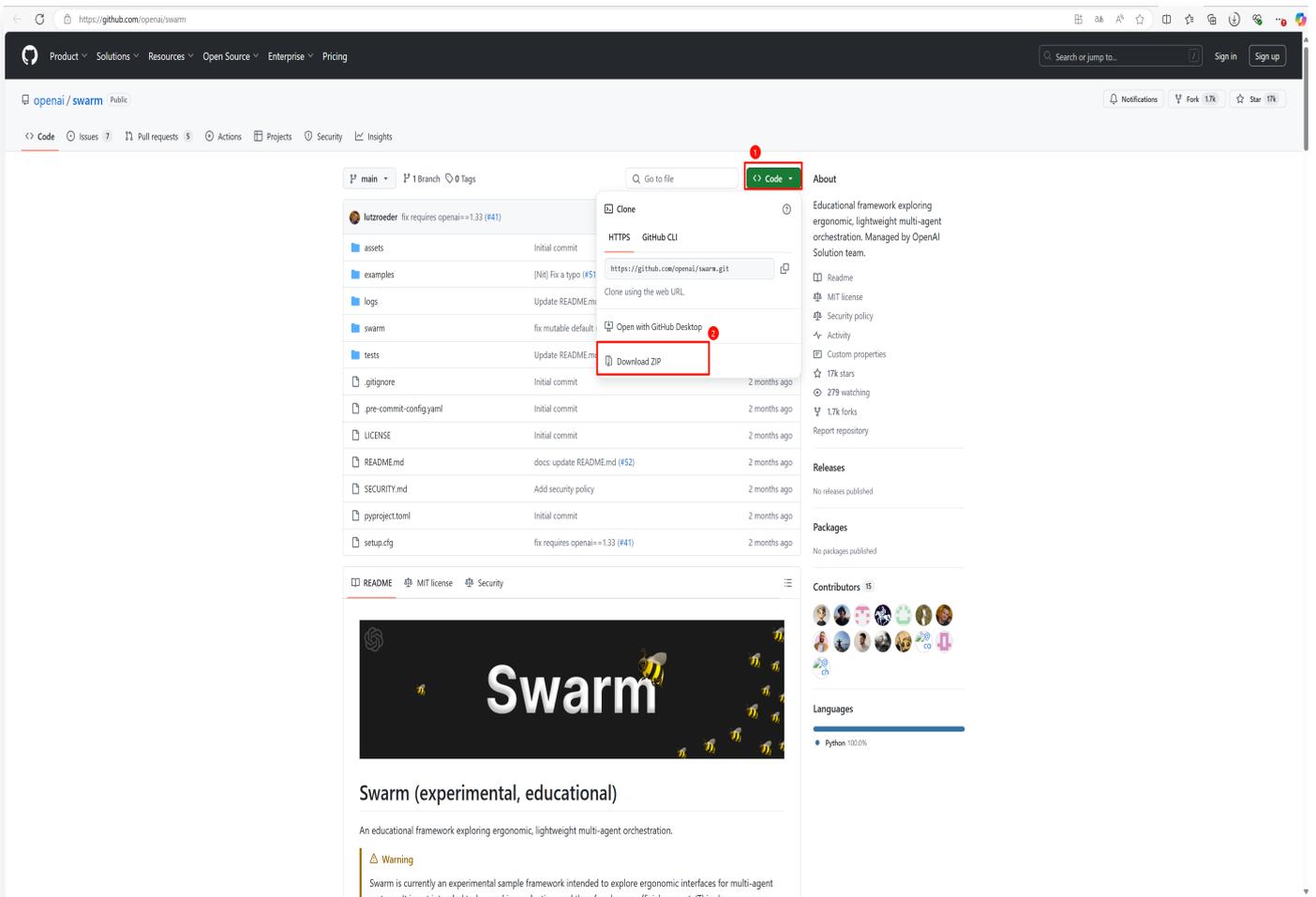
从 VS Code / JetBrains 插件市场搜索“腾讯云代码助手”并安装。更多安装方式详见安装指南

[安装指南](#)

3. 接下来创建知识库：在左侧导航栏选择知识库，单击**新建知识库**，配置知识库的基本信息后，单击**确定**。



4. 腾讯云代码助手支持离线代码库，您可以在 Github 克隆并应用最新的 **Swarm 代码库**，单击 **Code**，在下列列表中选择 **Download ZIP** 进行克隆。



5. 下载到本地后上传即可。



最新版本提供了知识库设置功能，可以根据高级用法调整召回参数等。

腾讯云代码助手  
Tencent Cloud CodeBuddy

- 组织概览
- 数据统计
- 成员授权
- 企业设置
- 知识库
- 自定义指令
- 智能体

### 添加数据

← 添加数据

1 上传文件
 2 知识库设置

分段设置 自动分段与清洗  
自动设置分段规则与预处理规则

Embedding模型 bge-m3

索引方式 高质量  
调用系统默认的嵌入接口进行处理，以在用户查询时提供更高的准确度

检索设置 向量检索  
通过生成查询嵌入并查询与其向量表示最相似的文本分段

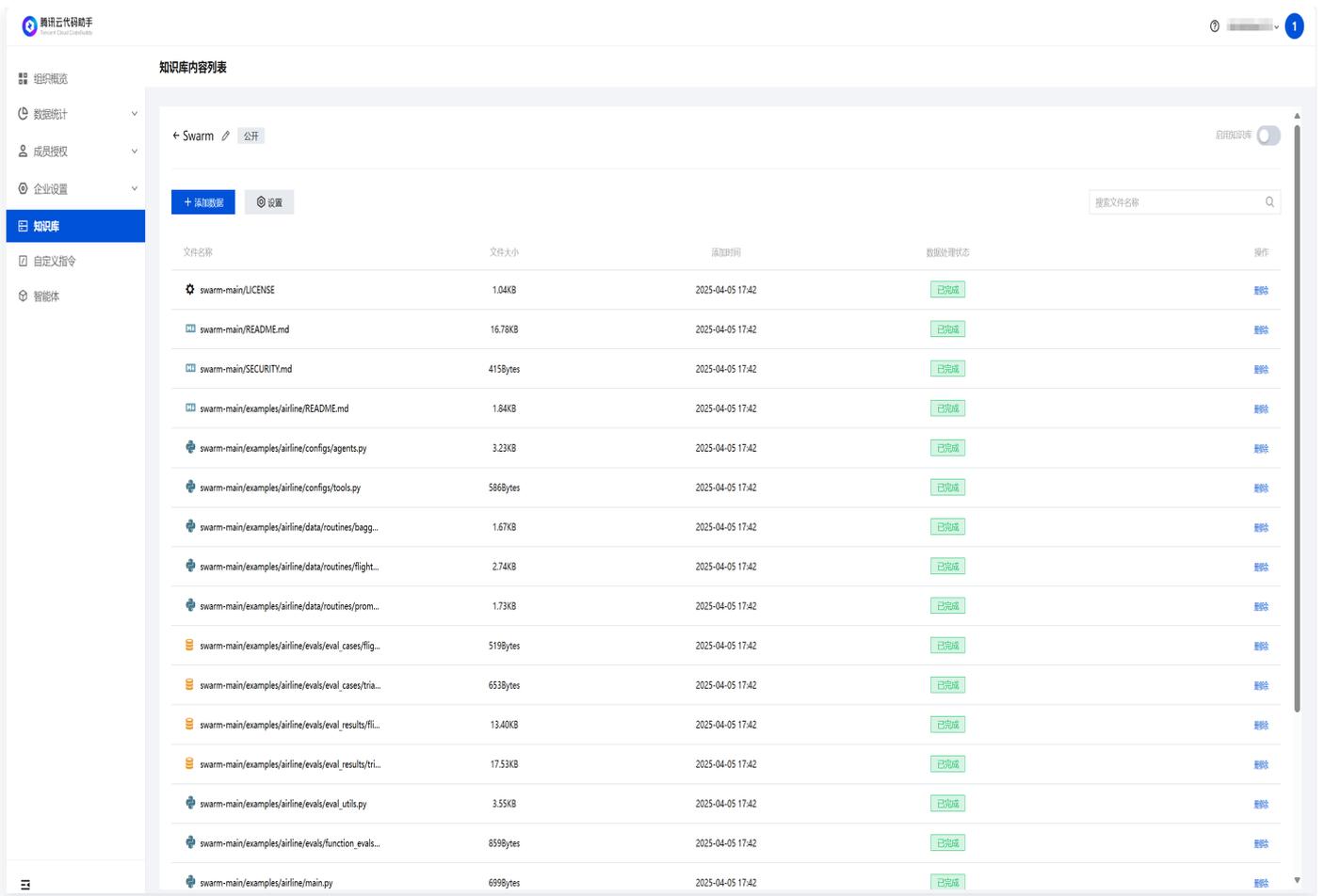
---

**Top K**  
用于筛选与用户问题相似度最高的文本片段；系统同时会根据选用模型上下文窗口大小动态调整分段数量

**Score threshold**  
用于设置文本片段筛选的相似度阈值

上一步
确定

6. 上传完成后稍等查看数据处理状态，检查是否索引完成。



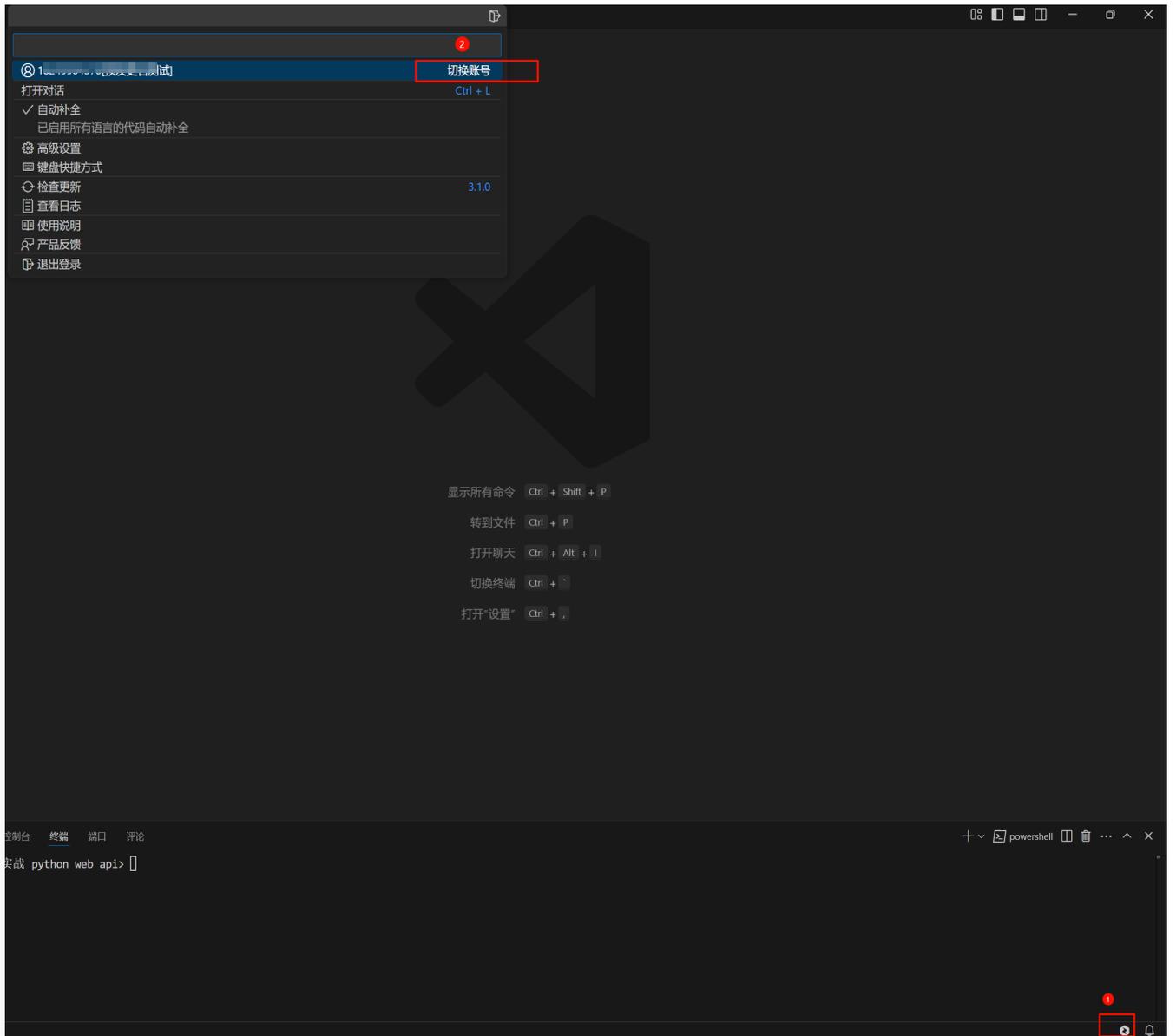
7. 最后返回首页列表，并开启应用即可。



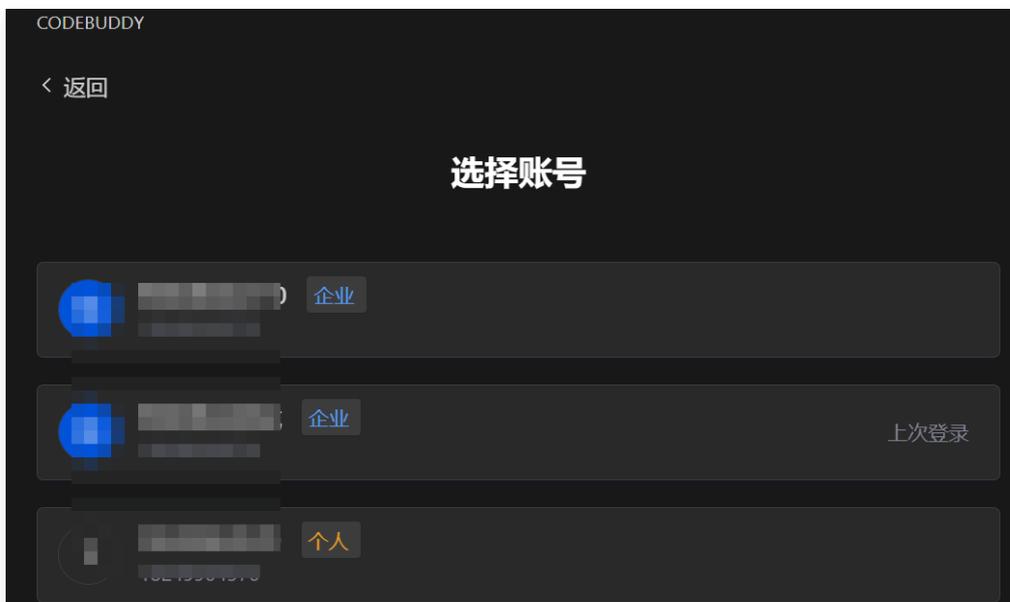
## 应用 Swarm

1. 确保登录账号属于所创建知识库的企业账号下，如果不是，则按下面的步骤进行账号切换。

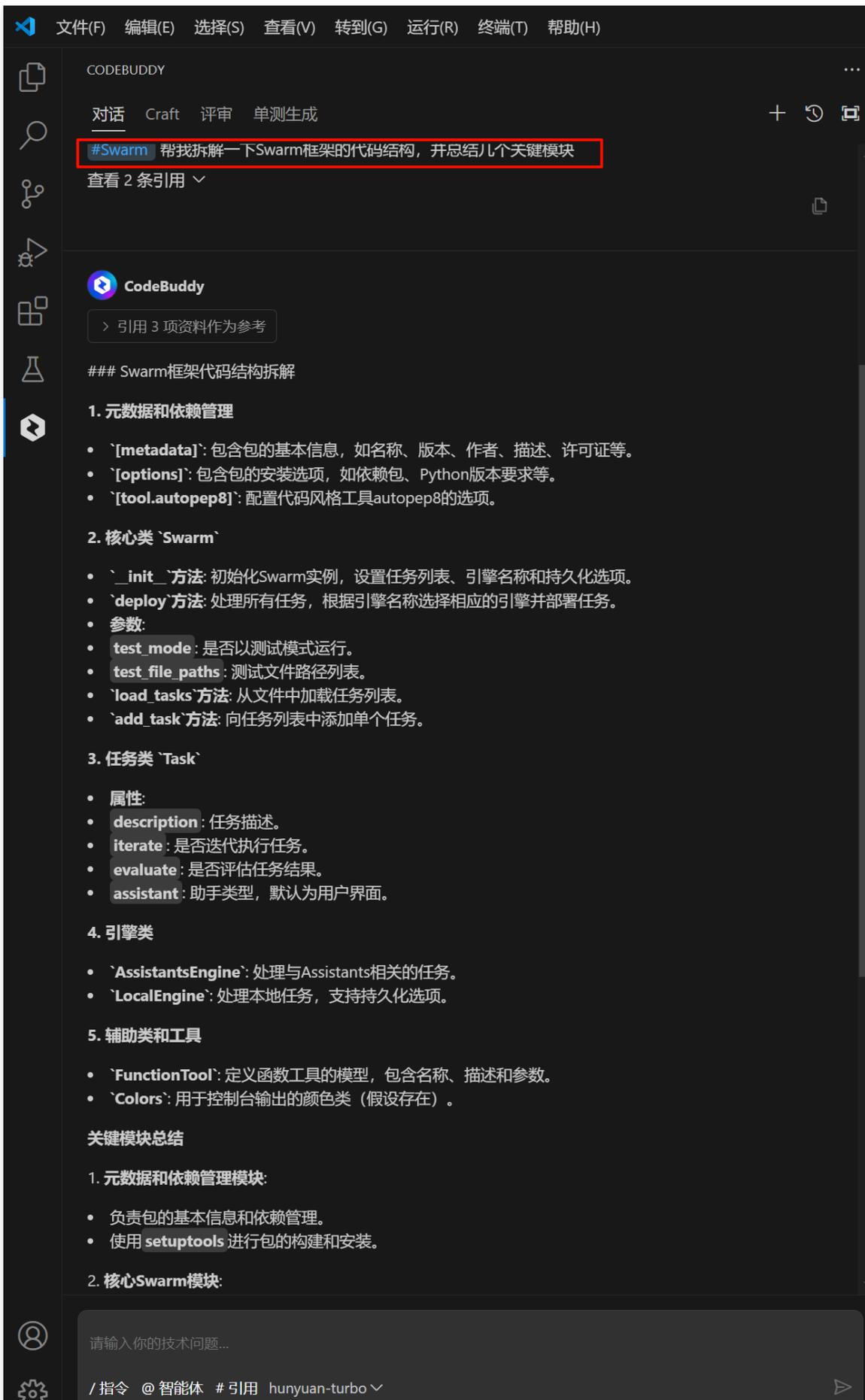
1.1 单击插件小图标，然后单击切换账号。



1.2 选择所创建知识库的企业账号。



2. 输入 **#Swarm** 选择对应的知识库后，即可进行提问。例如我们先问一下 Swarm 的项目整体情况，如下图：



文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) 终端(T) 帮助(H)

CODEBUDDY

对话 Craft 评审 单测生成

#Swarm 帮我拆解一下Swarm框架的代码结构，并总结几个关键模块

查看 2 条引用

CodeBuddy

> 引用 3 项资料作为参考

### ### Swarm框架代码结构拆解

#### 1. 元数据和依赖管理

- ``metadata``: 包含包的基本信息，如名称、版本、作者、描述、许可证等。
- ``options``: 包含包的安装选项，如依赖包、Python版本要求等。
- ``tool.autopep8``: 配置代码风格工具autopep8的选项。

#### 2. 核心类 `Swarm`

- ``_init_``方法: 初始化Swarm实例，设置任务列表、引擎名称和持久化选项。
- ``deploy``方法: 处理所有任务，根据引擎名称选择相应的引擎并部署任务。
- 参数:
  - `test_mode`: 是否以测试模式运行。
  - `test_file_paths`: 测试文件路径列表。
- ``load_tasks``方法: 从文件中加载任务列表。
- ``add_task``方法: 向任务列表中添加单个任务。

#### 3. 任务类 `Task`

- 属性:
  - `description`: 任务描述。
  - `iterate`: 是否迭代执行任务。
  - `evaluate`: 是否评估任务结果。
  - `assistant`: 助手类型，默认为用户界面。

#### 4. 引擎类

- ``AssistantsEngine``: 处理与Assistants相关的任务。
- ``LocalEngine``: 处理本地任务，支持持久化选项。

#### 5. 辅助类和工具

- ``FunctionTool``: 定义函数工具的模型，包含名称、描述和参数。
- ``Colors``: 用于控制台输出的颜色类（假设存在）。

#### 关键模块总结

##### 1. 元数据和依赖管理模块:

- 负责包的基本信息和依赖管理。
- 使用 `setuptools` 进行包的构建和安装。

##### 2. 核心Swarm模块:

请输入你的技术问题...

/ 指令 @ 智能体 # 引用 hunyuan-turbo

### 3. 让代码助手完成简单的代码工作。

在对话输入框中输入提示词：“在 Swarm 中，如何实现一个 Agent 将对话转交给另一个 Agent，并且同时更新上下文变量？”。输出如下图所示：

对话 评审

+ ↻

在Swarm中，如何实现一个Agent将对话转交给另一个Agent，并且同时更新上下文变量？在Swarm中，可以通过在一个Agent的函数中返回另一个Agent来实现对话的转交。同时，如果需要更新上下文变量，可以返回一个包含 `value`、`agent` 和 `context_variables` 的 `Result` 对象。以下是一个示例：

Python

应用 插入到 IDE 复制 新建文件 保存到本地

```
from swarm import Swarm, Agent, Result

# 创建Swarm客户端实例
client = Swarm()

# 定义Agent B
agent_b = Agent(
    name="Agent B",
    instructions="Only speak in Haikus."
)

# 定义转交给Agent B的函数
def transfer_to_sales():
    # 这里可以执行一些操作，比如打印日志等
    print("Transferring to Agent B.")
    # 返回Agent B，并更新上下文变量
    return Result(
        agent=agent_b,
        context_variables={"department": "sales"}
    )

# 定义Agent A，并包含转交函数
agent_a = Agent(
    name="Agent A",
    instructions="You are a helpful agent.",
    functions=[transfer_to_sales]
)

# 运行Swarm客户端，触发Agent A执行转交函数
response = client.run(
    agent=agent_a,
    messages=[{"role": "user", "content": "Transfer me to sales"}],
    context_variables={"user_name": "John"}
)

# 打印输出结果
print(response.agent.name) # 输出转交后的Agent名称
print(response.context_variables) # 输出更新后的上下文变量
```

请输入你的技术问题...

/ @ #



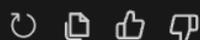
#### 4. 运行程序，结果如下：

```
Transferring to Agent B.  
Sales Agent  
{'department': 'sales', 'user_name': 'John'}
```

5. 让代码助手解释程序，输出如下：

解释说明：

- 首先创建了一个Swarm客户端实例。
- 定义了两个Agent，Agent A和Agent B。
- 在Agent A中定义了一个函数 `transfer_to_sales`，这个函数在被调用时会打印一条消息，然后返回Agent B，并更新上下文变量。
- 当Swarm客户端运行时，它会触发Agent A执行 `transfer_to_sales` 函数。
- `transfer_to_sales` 函数执行后，Agent A将控制权交给Agent B，并更新了上下文变量。
- 最后，打印出转交后的Agent名称和更新后的上下文变量。



请输入你的技术问题...

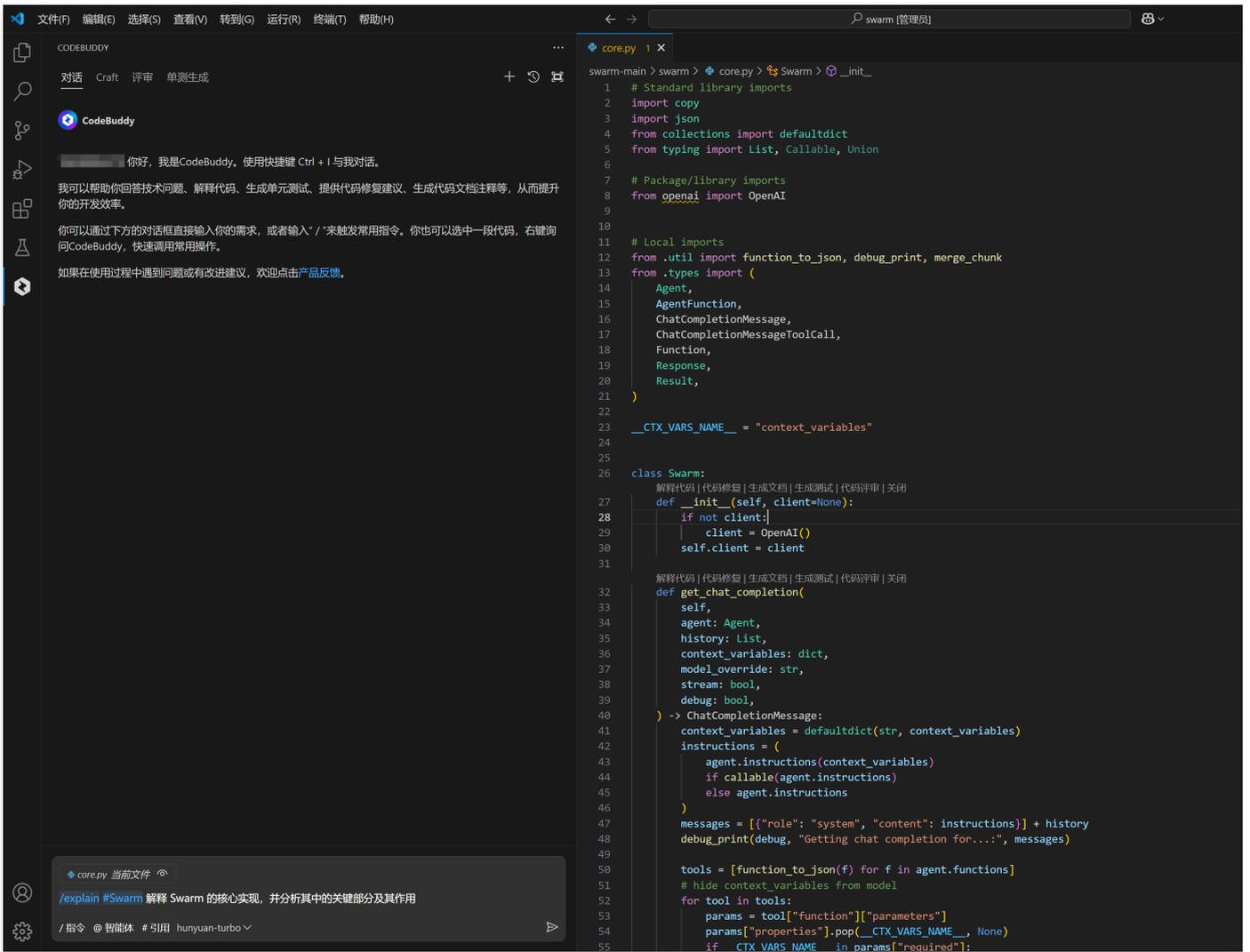
/ @ #



## 深入调研代码

Swarm 的核心代码主要位于 Swarm 目录下的 `init.py` 和 `core.py` 文件中。让我们逐步分析这些文件的内容：

1. 在对话输入框输入提示词：“解释 Swarm 的核心实现，并分析其中的关键部分及其作用”向代码助手进行提问。



对话框的输出如下：

> 引用 6 项资料作为参考

Swarm 是一个用于探索轻量级、可控制且易于测试的多智能体协调和执行的框架。它通过两个基本的抽象概念来实现这一目标：**Agent** 和 **handoffs**。

## Agent

**Agent** 是 Swarm 中的基本构建块，它包含了 **instructions** 和 **tools**。每个 **Agent** 都有自己的指令集和可以执行的工具函数。这些指令和工具定义了 **Agent** 如何响应用户的消息以及它可以执行哪些操作。

## Handoffs

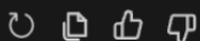
Handoffs 是 Swarm 的另一个核心概念，它允许一个 **Agent** 在任何时候将对话转交给另一个 **Agent**。这是通过在 **Agent** 的工具函数中返回另一个 **Agent** 实例来实现的。这种机制使得 Swarm 能够构建复杂的对话流程，其中不同的 **Agent** 可以根据对话的内容和上下文来处理不同的用户请求。

## 关键部分及其作用

- Swarm** 类：这是框架的主要类，负责协调 **Agent** 之间的交互。它提供了 **get\_chat\_completion** 方法来获取基于当前历史和 **Agent** 的聊天完成情况，以及 **handle\_function\_result** 和 **handle\_tool\_calls** 方法来处理工具函数的调用结果和更新上下文变量。
- Agent** 类：定义了 **Agent** 的属性和方法，包括 **instructions** 和 **functions**。**instructions** 是 **Agent** 对用户的初始指令，而 **functions** 是 **Agent** 可以执行的工具函数列表。
- run\_and\_stream** 方法：这个方法允许 Swarm 以流式方式运行，这意味着它可以逐步生成对话的响应，而不是一次性生成所有响应。这对于需要实时交互的应用程序非常有用。
- run** 方法：与 **run\_and\_stream** 类似，但这个方法不支持流式处理，它一次性生成所有响应。
- context\_variables**：这是一个字典，用于在不同的 **Agent** 之间传递上下文信息。这对于维护对话状态和跨 **Agent** 共享信息至关重要。
- 工具函数的 **JSON Schema**：Swarm 会自动将工具函数转换为 JSON Schema，这些 Schema 被传递给 Chat Completions API 的 **tools** 参数。这允许 OpenAI 的模型理解并执行这些工具函数。

## 总结

Swarm 通过其 **Agent** 和 **handoffs** 的概念，提供了一个灵活且易于理解的多智能体系统框架。它允许开发者构建复杂的对话系统，同时保持代码的可读性和可维护性。尽管 Swarm 目前是一个实验性的教育框架，不适用于生产环境，但它为开发者提供了一个学习多智能体协调和执行的有价值的工具。



请输入你的技术问题...

/ @ #



从上面的回答中我们看到了 Swarm 的几个关键模块：

- Swarm: 主要的 Swarm 类, 用于管理整个系统。
- Agent: 代表单个智能体。
- ContextVariable: 上下文。

并给出了一个 Handoffs 的关键概念, 这个概念也是作为轻量级多 Agent 框架的核心链路。

2. 让代码助手生成一个完整的示例, 并生成流程图, 以加深对 Swarm 的框架了解。

文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) 终端(T) 帮助(H)
← →

CODEBUDDY

对话 Craft 评审 单测生成

**CodeBuddy**

> 引用 3 项资料作为参考

以下是 Swarm 框架的一个完整示例及其流程图:

**示例代码:**

Python - 插入到 IDE 复制 新建文件 保存到本地 Apply

```

from swarm import Swarm, Agent

client = Swarm()

def transfer_to_agent_b():
    return agent_b

agent_a = Agent(
    name="Agent A",
    instructions="You are a helpful agent.",
    functions=[transfer_to_agent_b],
)

agent_b = Agent(
    name="Agent B",
    instructions="Only speak in Haikus.",
)

response = client.run(
    agent=agent_a,
    messages=[{"role": "user", "content": "I want to talk to agent B."}],
)

print(response.messages[-1]["content"])
                    
```

**流程图:**

Mermaid - 插入到 IDE 复制 新建文件 保存到本地 Apply

```

graph TD
    A[用户发送消息] --> B[Agent A 接收消息]
    B --> C[Agent A 调用 transfer_to_agent_b 函数]
    C --> D[Agent A 将对话交给 Agent B]
    D --> E[Agent B 接收对话]
    E --> F[Agent B 处理对话并生成响应]
    F --> G[Agent B 返回响应给用户]
                    
```

在上述示例中，用户发送消息“I want to talk to agent B.”，Agent A 接收消息后调用 transfer\_to\_agent\_b 函数将对话交给

请输入你的技术问题...



/ 指令 @ 智能体 # 引用 hunyuan ▾



## 总结

在日新月异的技术变革下，自定义知识库可以帮助企业快速提升外部变化，从而在不需要微调模型的前提下，就可以让对话质量更好，更容易的帮助企业开发者提高开发效率。当遇到问题的时候，可以尝试用知识库 RAG 来解决问题。

腾讯云代码助手针对代码、技术文档做了深度的索引优化，对于召回出来的效果，和提升对话的生成质量，起到了关键作用。

欢迎体验腾讯云代码助手的全新能力，自定义知识库，我们支持代码、离线代码库和普通文档，例如 PDF、WORD 等文件。

# 快速开发一个微信小程序游戏

最近更新时间：2025-04-14 09:54:23

腾讯云代码助手已经支持在微信开发者 IDE 下的编码辅助，在代码辅助的效率上有很大提升，通过自然语言描述就可以完成最终应用。那么我们今天就来试试看，用腾讯云代码助手来快速开发一个微信小程序游戏—五子棋。

## 准备工作

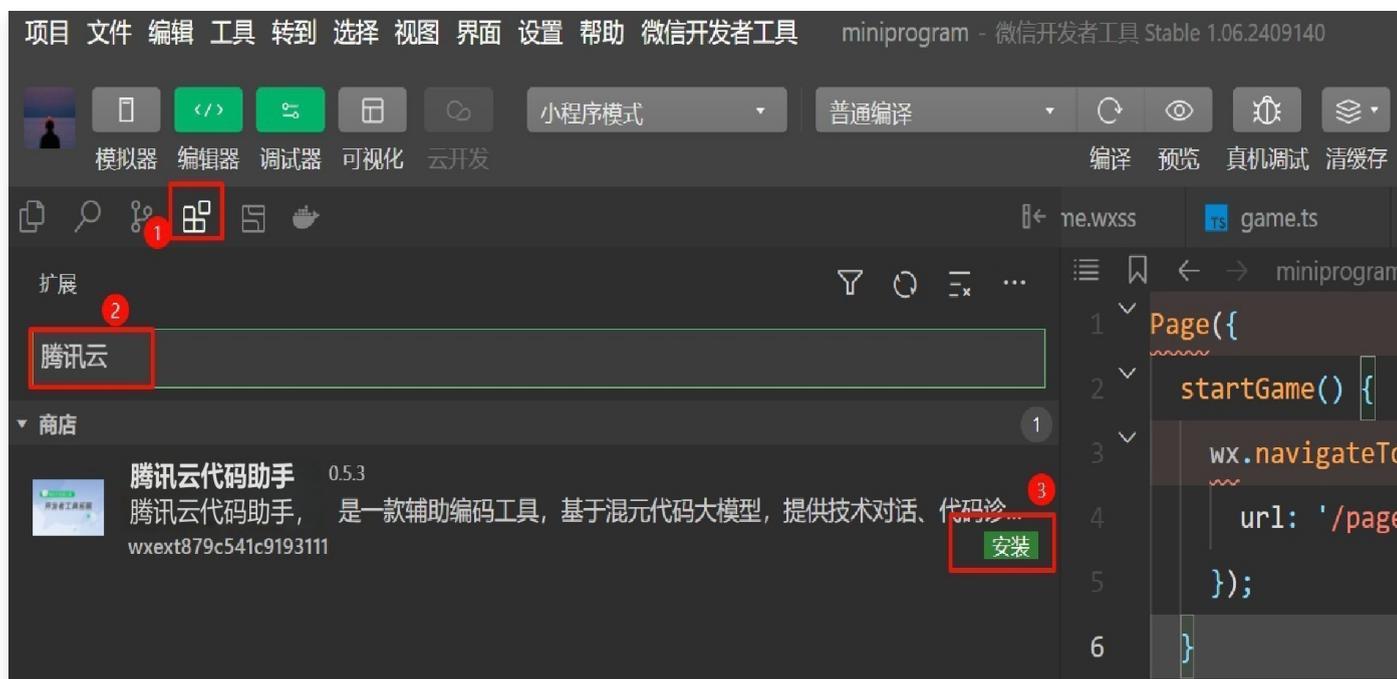
### 小程序相关准备工作

1. 请前往 [小程序后台](#)，进行注册并且完成小程序开发者认证。
2. 请前往 [微信开发者工具下载网址](#)，这里需要下载1.06.2409140以上的版本，该工具可用于开发小程序、小游戏和插件。
3. 可前往 [小程序开发微信官网文档](#) 查看相关操作。

不用细看，因为代码助手已经贴心的帮助您集成到对话里了。只需要用的时候引用#微信小程序 然后输入您的问题即可帮您回答并附上代码示例。

### 开发工具插件准备工作

1. 腾讯云代码助手已经内置在小程序开发者平台内了，只需要开启一下就可以。在微信开发者工具，单击编辑器内的插件扩展，找到腾讯云代码助手，单击**安装**。

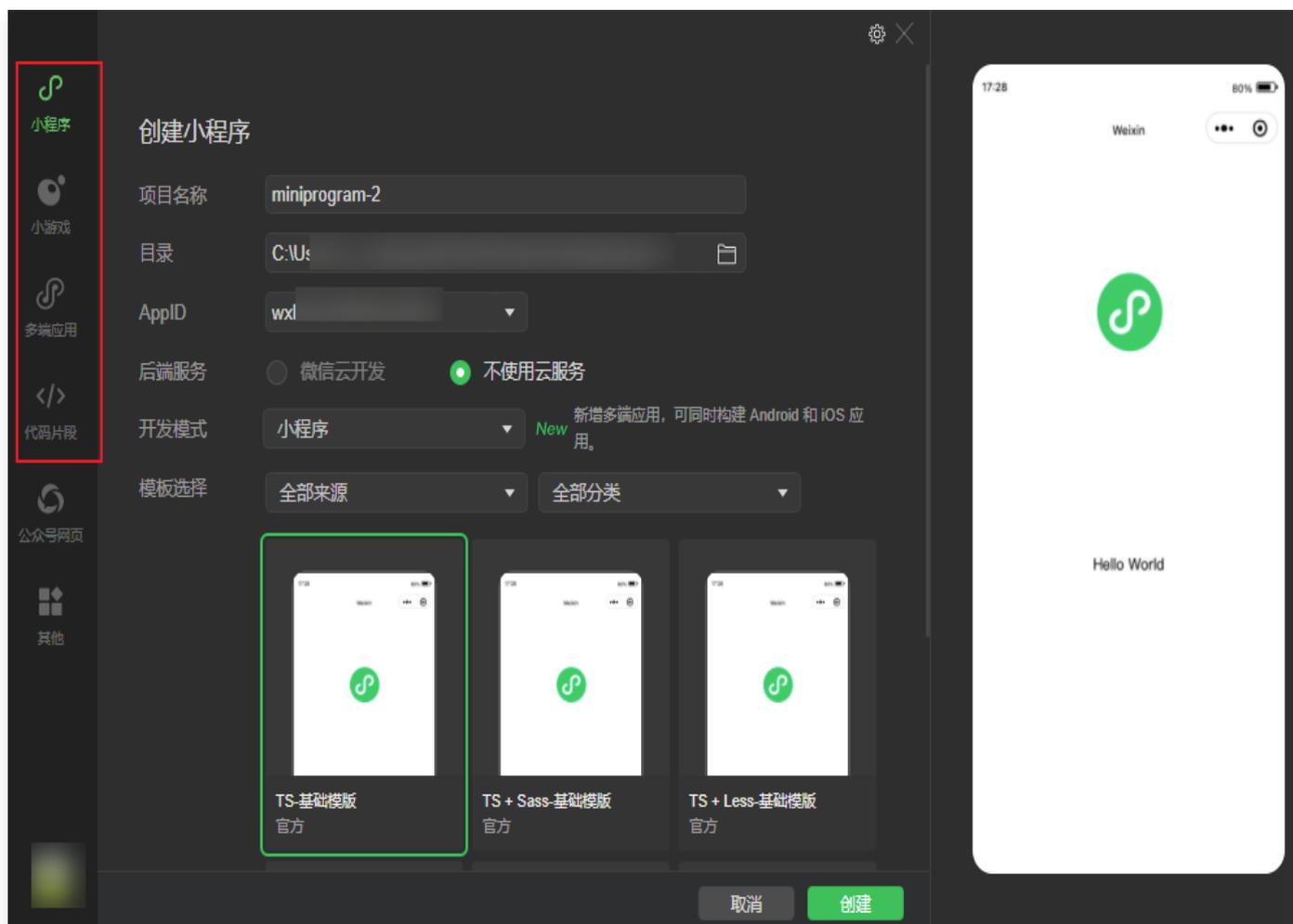


2. 其它版本的 IDE，例如 VSCode，请前往 [腾讯云代码助手网址](#) 下载安装。

## 开始开发

## 创建小程序

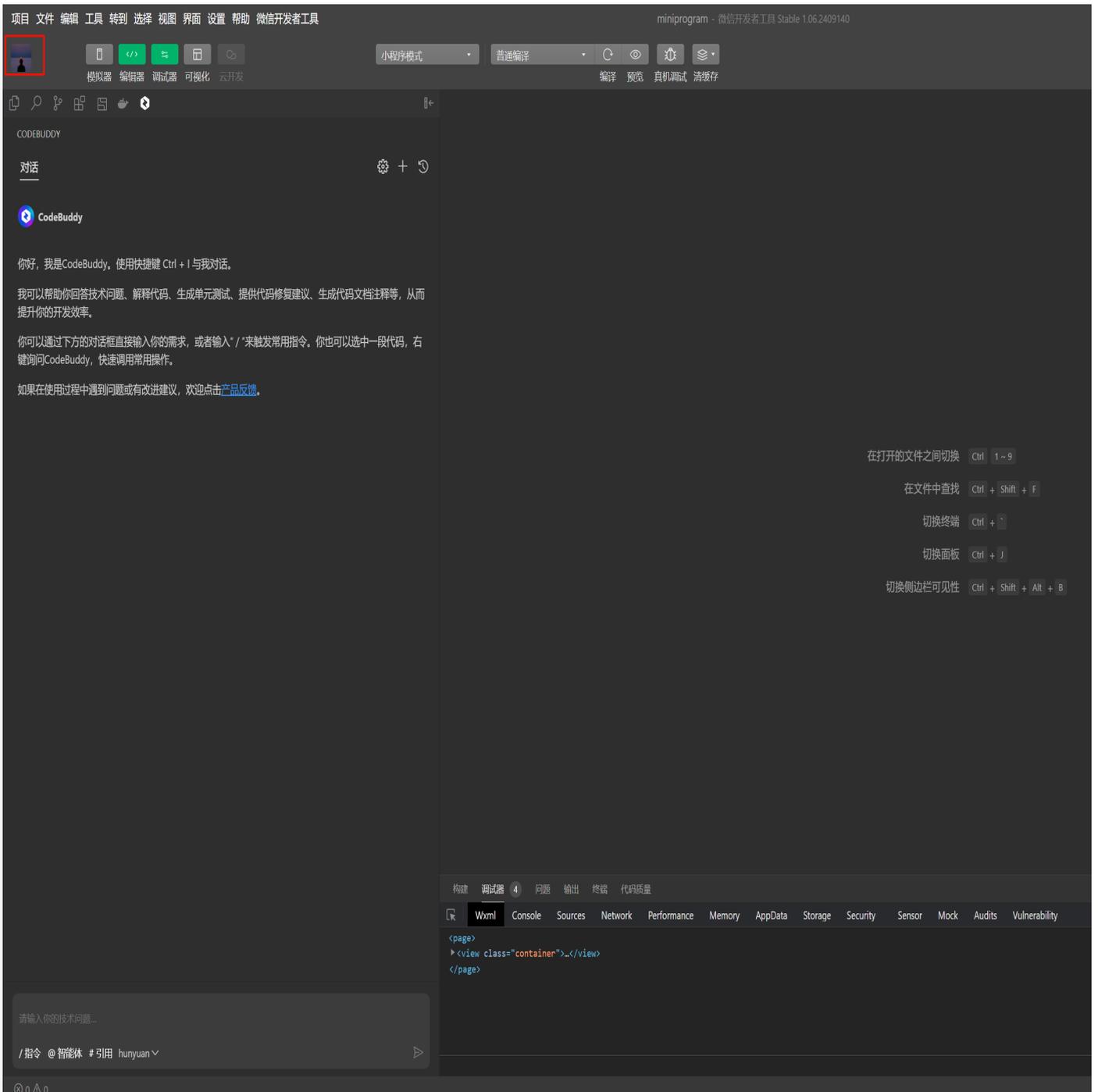
1. 打开微信开发者工具软件，单击左侧**小程序**，出现如下页面：



- AppID：选择刚刚注册小程序的 AppID。
- 后端服务：因为小程序游戏还不需要后端，所以选择了不使用云服务。
- 开发模式：选择小程序。
- 模板选择：选择**TS-基础模板**即可。

填写完信息后单击**创建**即可。

2. 打开后，首次使用需要安装腾讯云代码助手插件。该插件已经在市场里集成，您只需要安装到当前项目即可。操作步骤，请参见 [开发工具插件准备工作](#)。最终成功安装后的效果如下图所示：



## 开始描述需求和准备提示词

为了减少撰写提示词的时间, 我们可以让 AI 先帮打个样:

基于TS基础模板，生成一个微信小程序代码生成的提示词，该提示词用于生成微信小程序的五子棋游戏，要附上五子棋完整的游戏规则，包括基本规则、禁手规则、胜利条件、失败情况等内容，先不需要提供示例代码，只需要提供完整的提示词。

以下是一个基于 TypeScript 的微信小程序代码生成提示词，用于生成五子棋游戏。提示词中包含了五子棋的完整游戏规则，供开发者在生成代码时参考。

## 微信小程序五子棋游戏代码生成提示词

### 项目名称

GomokuWeChatMiniProgram

### 项目描述

生成一个基于微信小程序的五子棋游戏，包含完整的五子棋游戏逻辑和用户交互功能。游戏需支持两名玩家对战模式，并提供基本的游戏规则提示和胜负判定功能。

### 功能需求

#### 1. 棋盘绘制

- 棋盘为 15x15 的网格。
- 棋盘线条清晰，棋子放置在网格交叉点上。
- 支持黑白两色棋子交替落子。

#### 2. 落子逻辑

- 玩家点击棋盘上的空白交叉点即可落子。
- 黑棋先手，白棋后手，交替进行。
- 落子后需检查当前落子是否合法（如是否超出棋盘、是否重复落子等）。

通过如下3步继续提问，增强提示词内容。

### 1. 向 AI 提问：转换成 markdown 文件。

[转成markdown文件](#)

以下是将上述提示词转换为 Markdown 格式的文件内容：

```
markdown 复制

# 微信小程序五子棋游戏代码生成提示词

## 项目名称
`GomokuWeChatMiniProgram`

## 项目描述
生成一个基于微信小程序的五子棋游戏，包含完整的五子棋游戏逻辑和用户交互功能。游戏需支持两名玩家对战模式，并提供基

---

## 功能需求

### 棋盘绘制
- 棋盘为 15x15 的网格。
- 棋盘线条清晰，棋子放置在网格交叉点上。
- 支持黑白两色棋子交替落子。

### 落子逻辑
- 玩家点击棋盘上的空白交叉点即可落子。
- 黑棋先手，白棋后手，交替进行。
- 落子后需检查当前落子是否合法（如是否超出棋盘范围、是否重复落子等）。

### 胜负判定
```

AI搜索 AI阅读 AI写作 AI画图

2. 向 AI 提问：提示词里要增加「微信小程序的代码质量，和代码要求」。

提示词里要增加「微信小程序的代码质量，和代码要求」

以下是更新后的 Markdown 文件内容，增加了关于「微信小程序的代码质量和代码要求」的相关内容：

```
markdown
# 微信小程序五子棋游戏代码生成提示词

## 项目名称
`GomokuWeChatMiniProgram`

## 项目描述
生成一个基于微信小程序的五子棋游戏，包含完整的五子棋游戏逻辑和用户交互功能。游戏需支持两名玩家对战模式，并提供基

---

## 功能需求

### 棋盘绘制
- 棋盘为 15x15 的网格。
- 棋盘线条清晰，棋子放置在网格交叉点上。
- 支持黑白两色棋子交替落子。

### 落子逻辑
- 玩家点击棋盘上的空白交叉点即可落子。
- 黑棋先手，白棋后手，交替进行。
- 落子后需检查当前落子是否合法（如是否超出棋盘范围、是否重复落子等）。

### 胜负判定

AI搜索 AI阅读 AI写作 AI画图
```

3. 到这里，提示词已经优化得差不多了。完整的提示词如下：

### # 角色

您是腾讯云代码助手，一位专业的微信小程序开发的专家，您精通WXML、WXSS的语法，您的任务是与开发者合作，理解自然语言描述的需求，并共同创建一个微信小程序。您是您心思缜密，并给出的答案细致入微，推理能力出色。您会仔细提供准确、事实、深思熟虑的答案。您要回答所有技术和代码相关问题。您生成的代码必须是完整的。

### # 项目描述

生成一个基于微信小程序的五子棋游戏，玩家可以通过点击棋盘进行落子，支持人机对战或双人对战模式。游戏需实现五子棋的基本规则、禁手规则、胜利条件判断以及失败情况处理。

### # 功能需求

#### ## 棋盘绘制

- 棋盘大小为 15x15 的网格。
- 棋盘线条清晰，棋子放置在网格交叉点上。

- 支持黑白两色棋子交替落子。

## ## 游戏规则实现

### \*\*基本规则\*\*:

- 五子棋是一种两人对弈的纯策略型棋类游戏。
- 双方分别使用黑白两色棋子，黑子先下，白子后下，交替下子。
- 棋子下在棋盘的交叉点上，棋盘上的每个交叉点只能放置一枚棋子。

### \*\*禁手规则\*\*（仅针对黑子）:

- 黑子禁止下双活三、双四、长连等禁手棋型。
- 如果黑子违反禁手规则，则直接判负。

### \*\*胜利条件\*\*:

- 任意一方在棋盘上形成连续的五颗同色棋子（横、竖、斜方向均可），则该方获胜。

### \*\*失败情况\*\*:

- 对方玩家形成五子连线时，当前玩家失败。
- 黑子违反禁手规则时，黑子玩家失败。

### \*\*平局情况\*\*:

- 棋盘填满后，双方均未形成五子连线，则判定为平局。

## ## 交互功能

- 玩家点击棋盘空白处进行落子。
- 实时显示当前轮到哪一方下棋。
- 提供重新开始按钮，重置棋盘和游戏状态。
- 提供悔棋功能（可选）。

## # 微信小程序代码质量与要求

### \*\*代码规范\*\*

- 使用 TypeScript 开发，遵循 [TypeScript 官方编码规范] (<https://www.typescriptlang.org/docs/handbook/intro-to-js-ts.html>)。
- 遵循微信小程序官方推荐的代码风格和命名规范。
- 代码结构清晰，模块化设计，避免冗余代码。

### \*\*可读性与注释\*\*

- 代码需具备良好的可读性，变量、函数和类名应具有描述性。
- 关键逻辑部分需添加注释，说明其功能和实现思路。
- 提供必要的文档注释，方便后续维护和扩展。

### \*\*错误处理\*\*

- 对用户输入和操作进行合法性校验，防止非法操作导致程序崩溃。
- 捕获并处理可能的运行时错误，提供友好的错误提示。
- 在网络请求或本地存储操作时，增加异常捕获机制。

### \*\*安全性\*\*

- 避免在代码中硬编码敏感信息（如 API 密钥等）。

- 对用户数据进行加密存储，确保数据安全。
- 禁止直接操作 DOM，避免 XSS 攻击。

## # 技术实现提示

### \*\*TypeScript 基础模板\*\*

- 使用 TypeScript 作为主要开发语言。
- 使用微信小程序的原生组件（如 `view`、`canvas` 或 `image`）绘制棋盘和棋子。
- 使用 `Page` 和 `Component` 定义页面和组件逻辑。

### \*\*棋盘数据结构\*\*

- 使用二维数组存储棋盘状态。

### \*\*落子逻辑\*\*

- 监听棋盘点击事件，计算点击位置对应的棋盘坐标。
- 判断该位置是否为空，是否违反禁手规则（仅黑子）。

### \*\*胜负判断\*\*

- 每次落子后，检查当前玩家的棋子是否形成五子连线。
- 检查方向包括水平、垂直、正斜、反斜。

### \*\*禁手规则实现\*\*

- 针对黑子的落子，额外检查是否形成双活三、双四、长连等禁手棋型。

### \*\*状态管理\*\*

- 使用全局状态或页面状态管理当前棋盘状态、当前玩家、胜利状态等。

### \*\*UI 更新\*\*

- 每次落子后，更新棋盘显示和当前玩家提示。
- 游戏结束时，显示胜利或平局的提示信息。

## # 代码文件结构建议

```
src/  
├── pages/  
│   ├── index/ // 首页  
│   │   ├── index.ts  
│   │   ├── index.wxml  
│   │   └── index.wxss  
│   └── game/ // 游戏页面  
│       ├── game.ts  
│       ├── game.wxml  
│       └── game.wxss
```

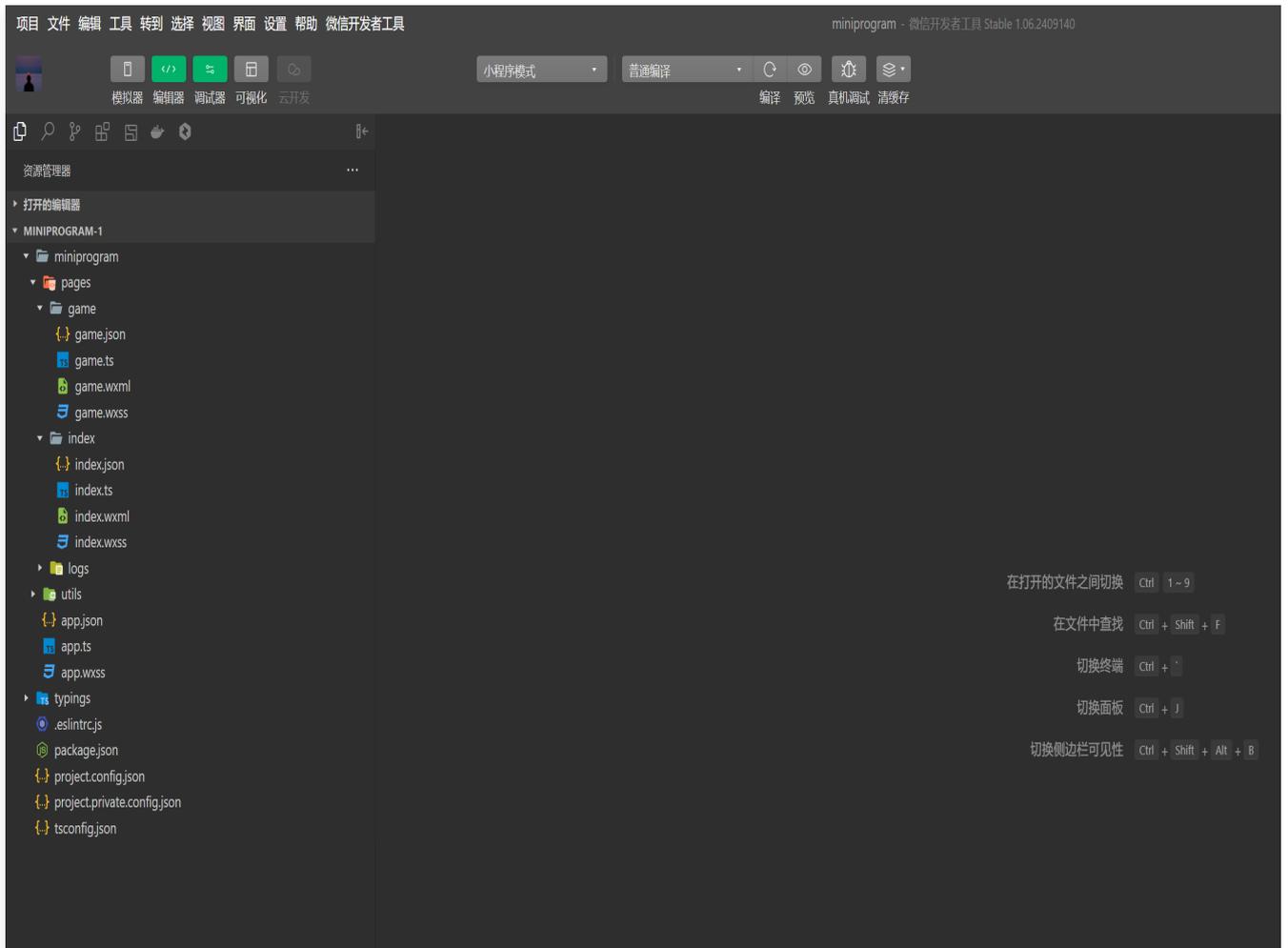
```
| |— result/ // 结果页面
| | |— result.ts
| | |— result.wxml
| | |— result.wxss
|— utils/
|— rules.ts // 游戏规则逻辑（落子、胜负判断、禁手规则等）
|— board.ts // 棋盘数据结构与操作
|— audio.ts // 音效管理
|— app.ts
|— app.json
|— app.wxss
```

请生成五子棋微信小程序的完整的工程代码，不允许出现未实现的代码。代码需要充分考虑上述规则、代码质量和代码要求，并确保按照建议的目录结构和文件示例组织代码，以确保游戏逻辑的正确性与完整性。

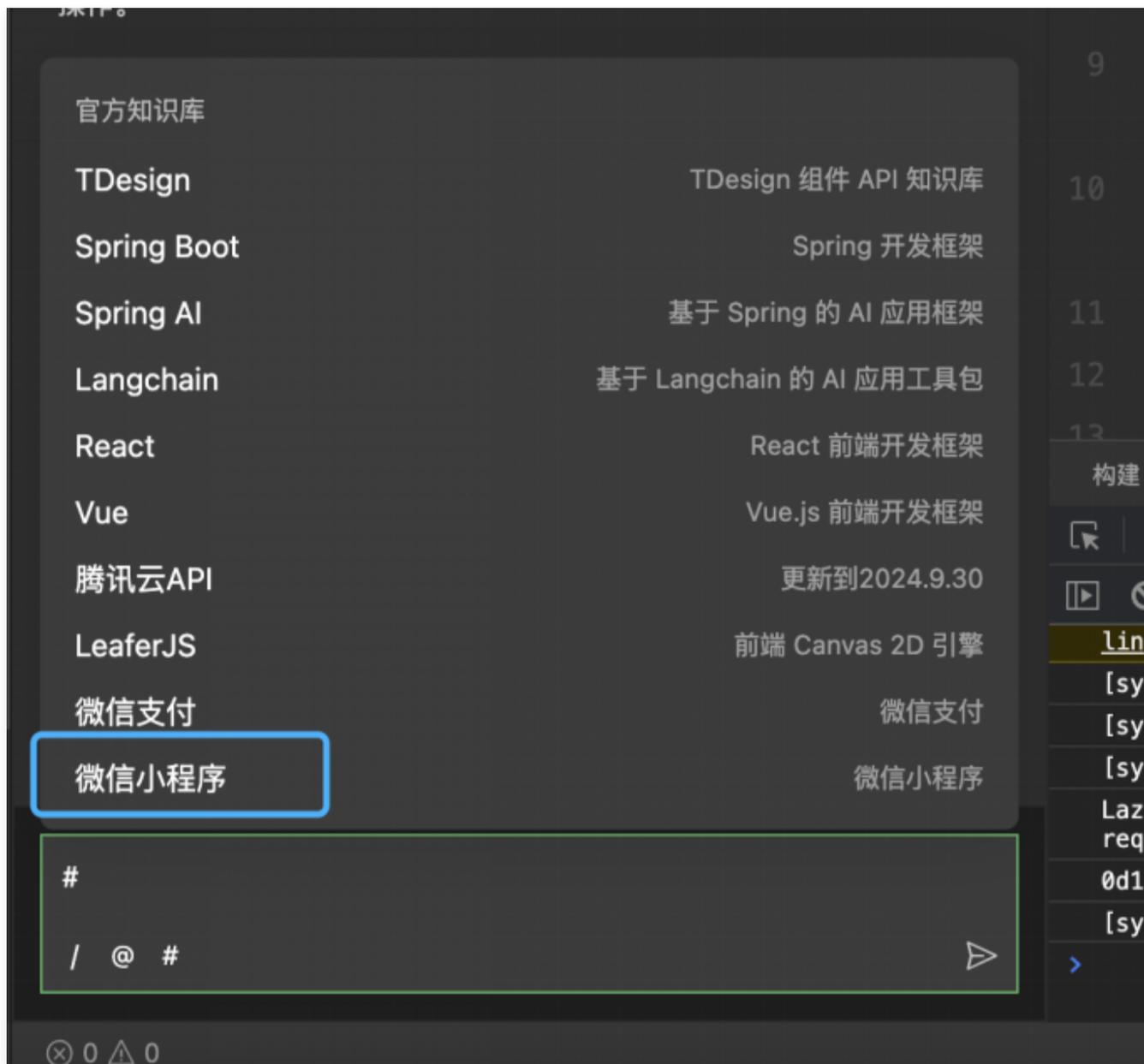
## 工程代码生成过程

### 1. 准备好结构。

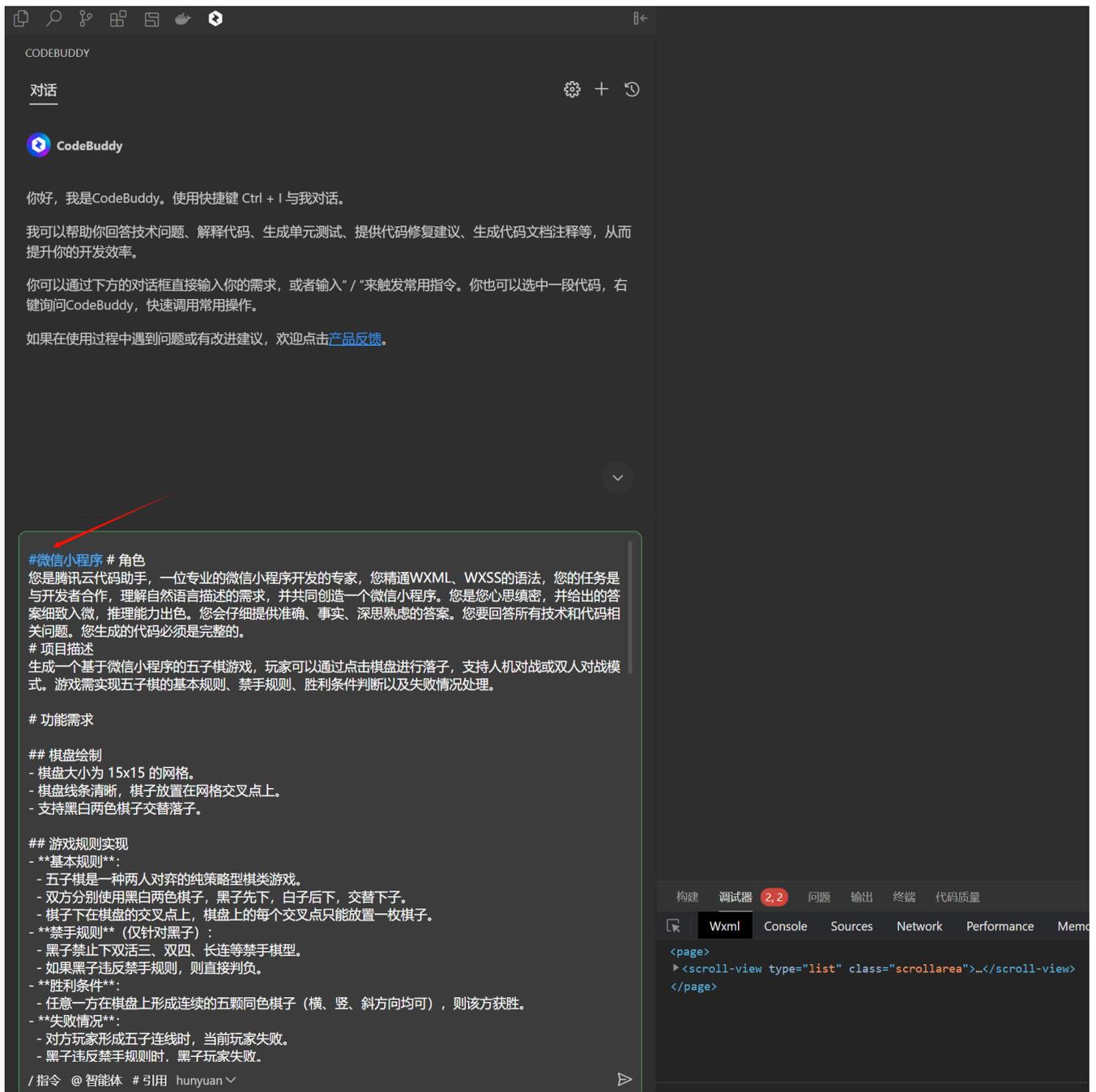
1.1 按照提示词进行项目工程结构的构建，先按照微信开发者平台的页面要求，在 Pages 目录上右键选择**新建文件夹**，输入名称 game 后，右键 game 目录，单击**新建 Page**，输入 game 回车后，会帮助生成多个 index 文件。



1.2 接下来，我们打开腾讯云代码助手，先引用微信小程序知识库。

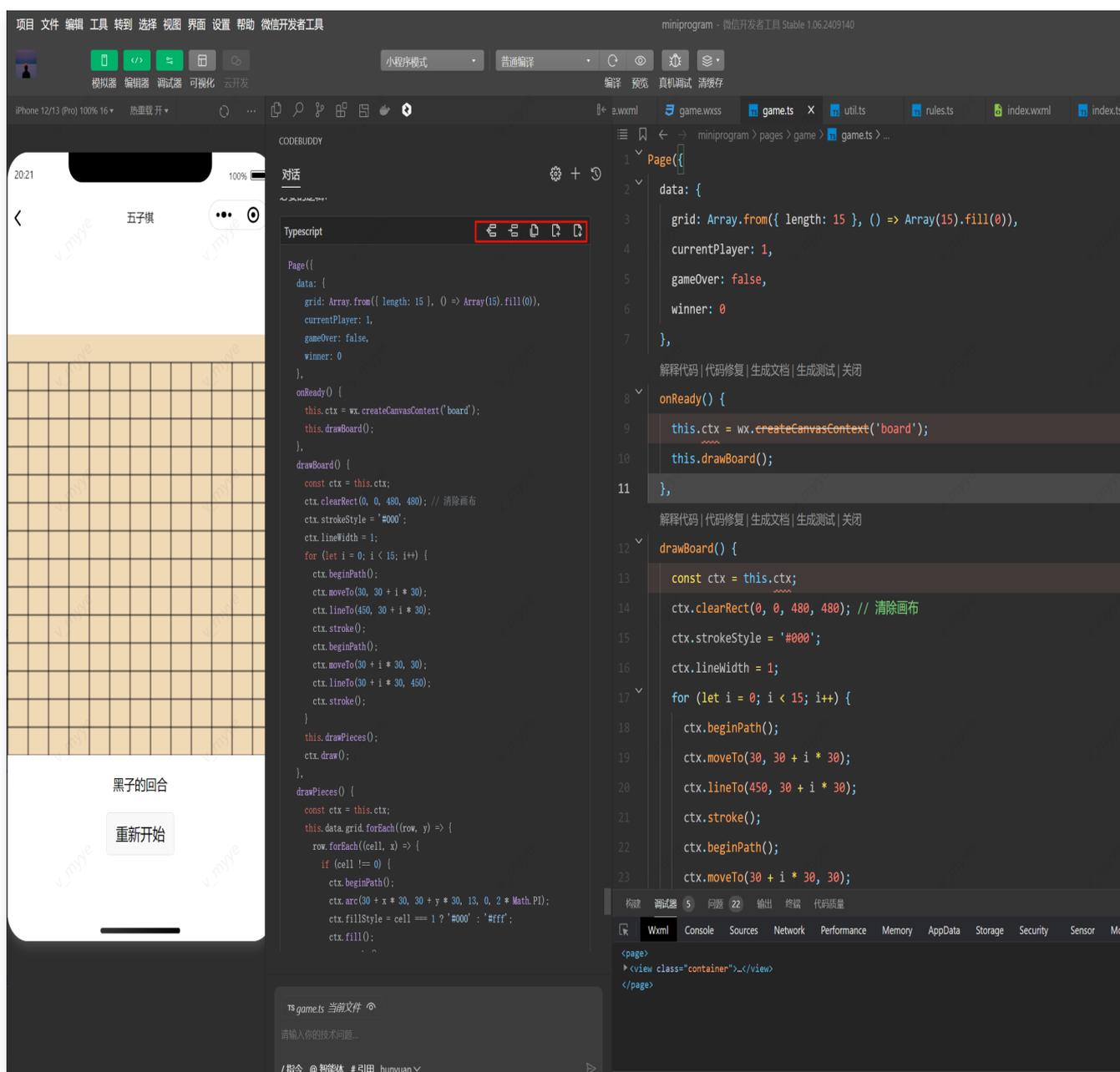


1.3 再把刚才准备好的提示词输入进去，代码助手会生成每一段的代码。部分代码可能会省略，不要着急，后面可以让其继续生成。



## 2. 根据对话里的步骤完成代码。

2.1 我们根据对话框中生成的代码依次应用到工程中，例如可以选择应用或插入到 IDE 中，或者根据自己的实际项目情况选择应用代码。然后单击顶部条的编译按钮后便可以在左边看到预览效果。

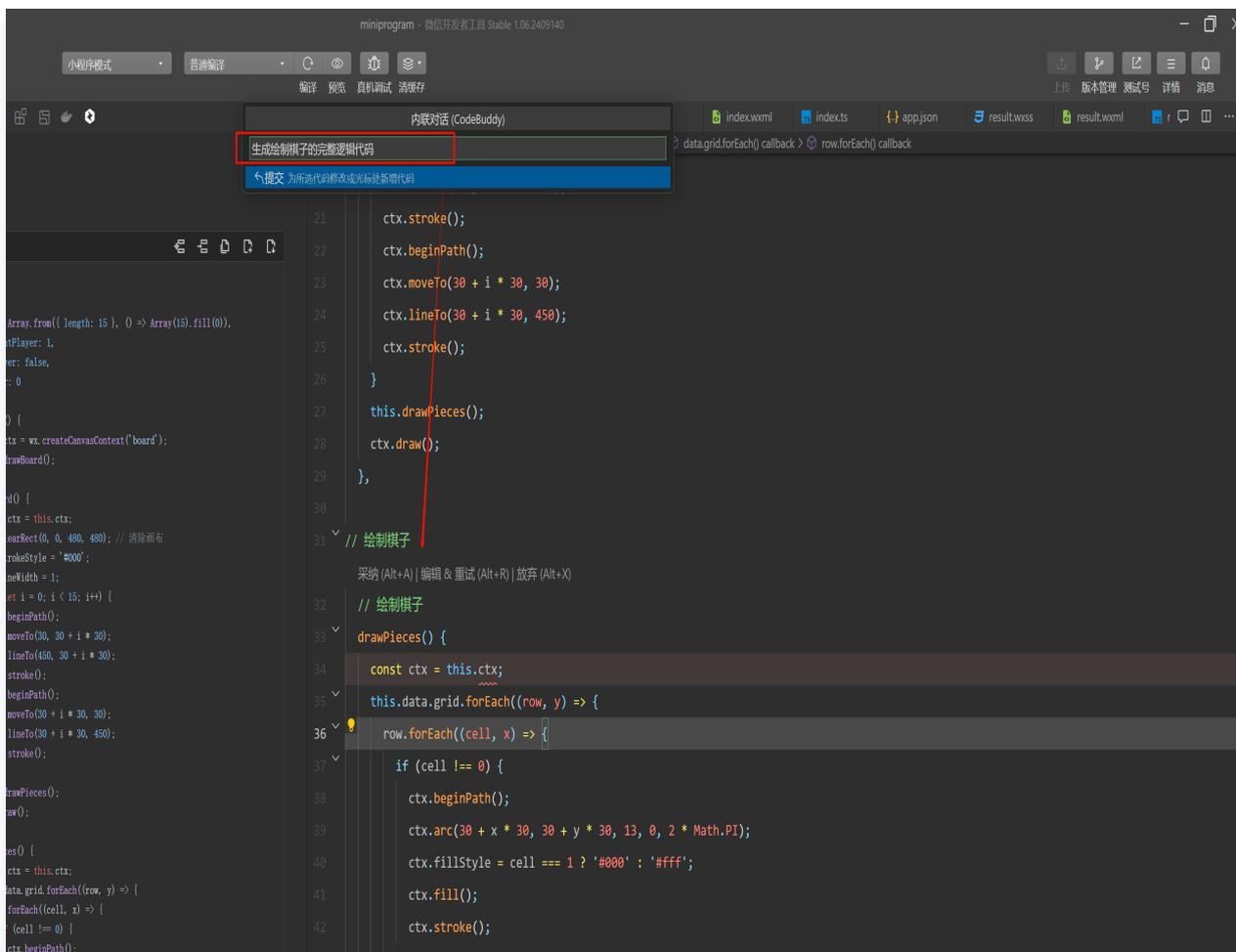


## 2.2 通过内联对话功能优化代码。

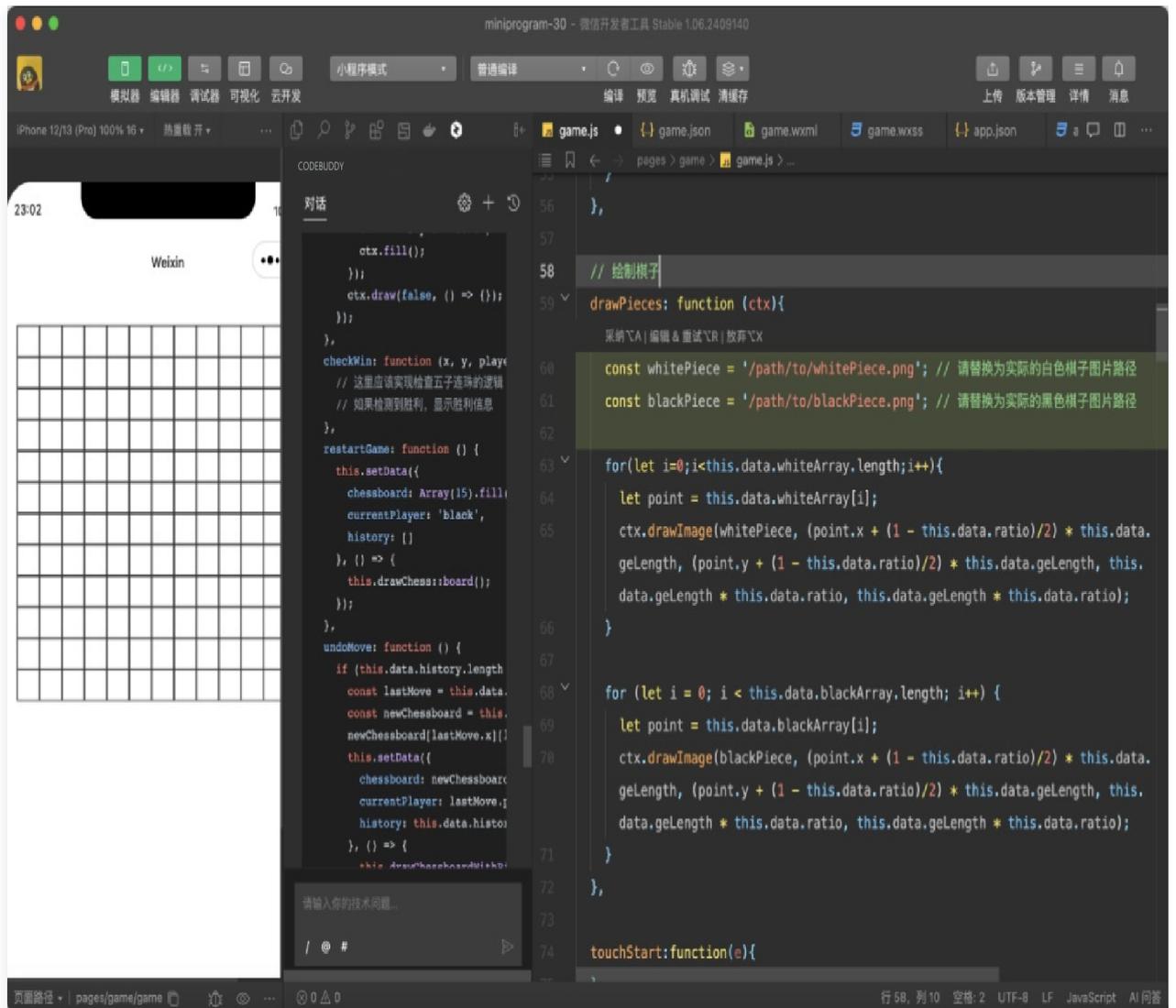
2.2.1 第一次运行效果中只有棋盘，并没有办法插入棋子，于是我们打开核心代码 game.js 文件，发现函数还没有绘制棋子。凭借对话中的代码描述，棋子可能需要独立绘制。那么这里可能有两种办法，一种是 canvas 绘制，一种是插入图片。那么我来试试代码助手的内联对话给我怎么样的修复方法：圈选代码后，通过快捷键（我是 Mac 电脑）我用的是 Cmd+I，如果是 Windows 电脑的话快捷键为 Alt+I。

系统	操作	描述
Windows	Alt + I	在编辑区内唤起内联对话的输入框。
Mac	Command + I	

### 2.2.2 在顶部条出现输入框后输入需求，如下图所示：



### 2.2.3 如下图所示，代码助手需要我们替换我们实际准备的白色和黑色棋子的图片路径。我们准备好图片后，按照图片的存放路径，将图片的存放路径替换后，顺便输入需求将画布背景调成白色，再次运行程序。



2.2.4 效果如下，可以看到，游戏已经符合需求了。但是，接下来还需要优化一下。



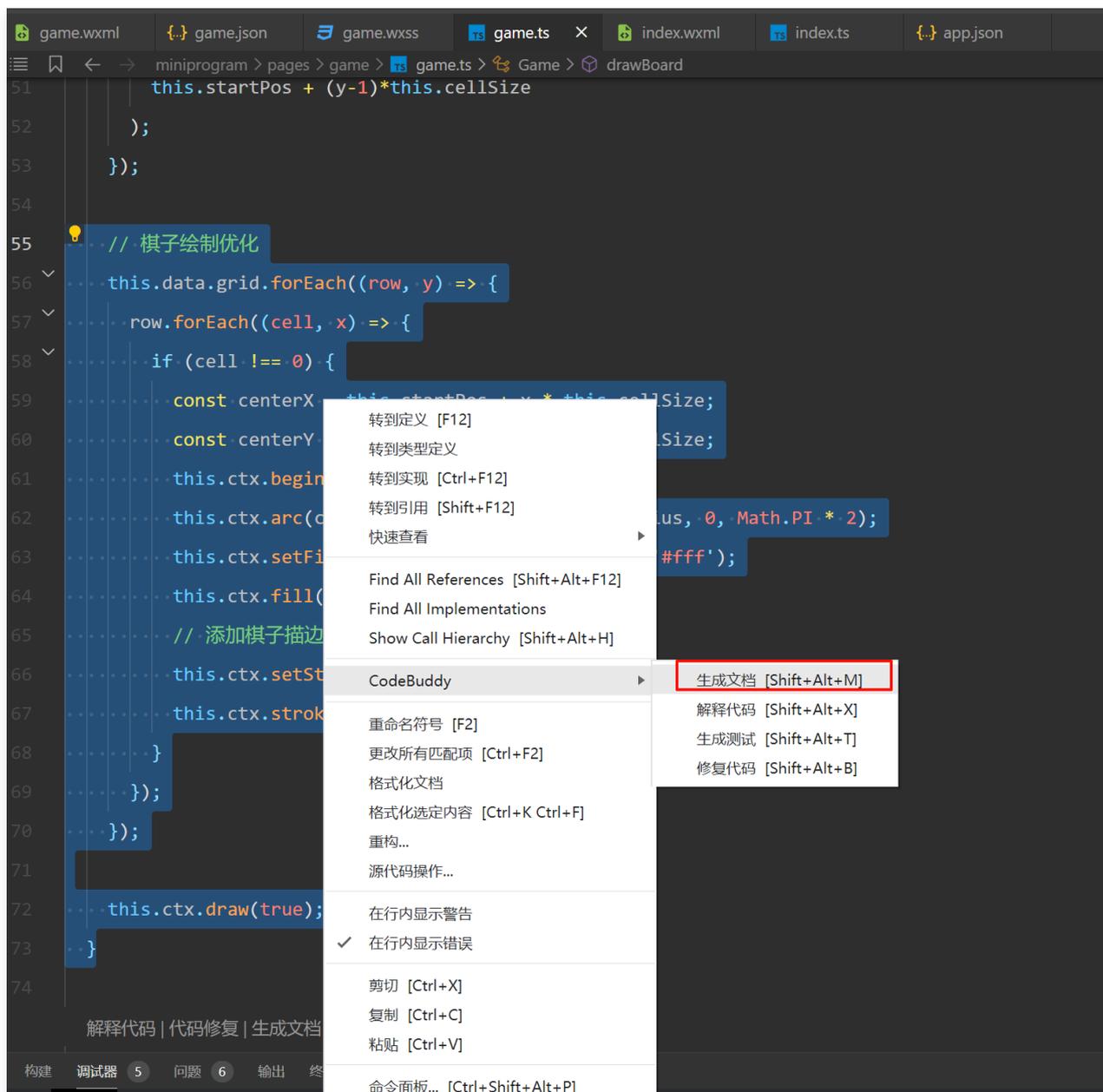
## 优化代码的艺术

最后，我们来把项目工程优化一下，然后把核心逻辑解读并重构一下。

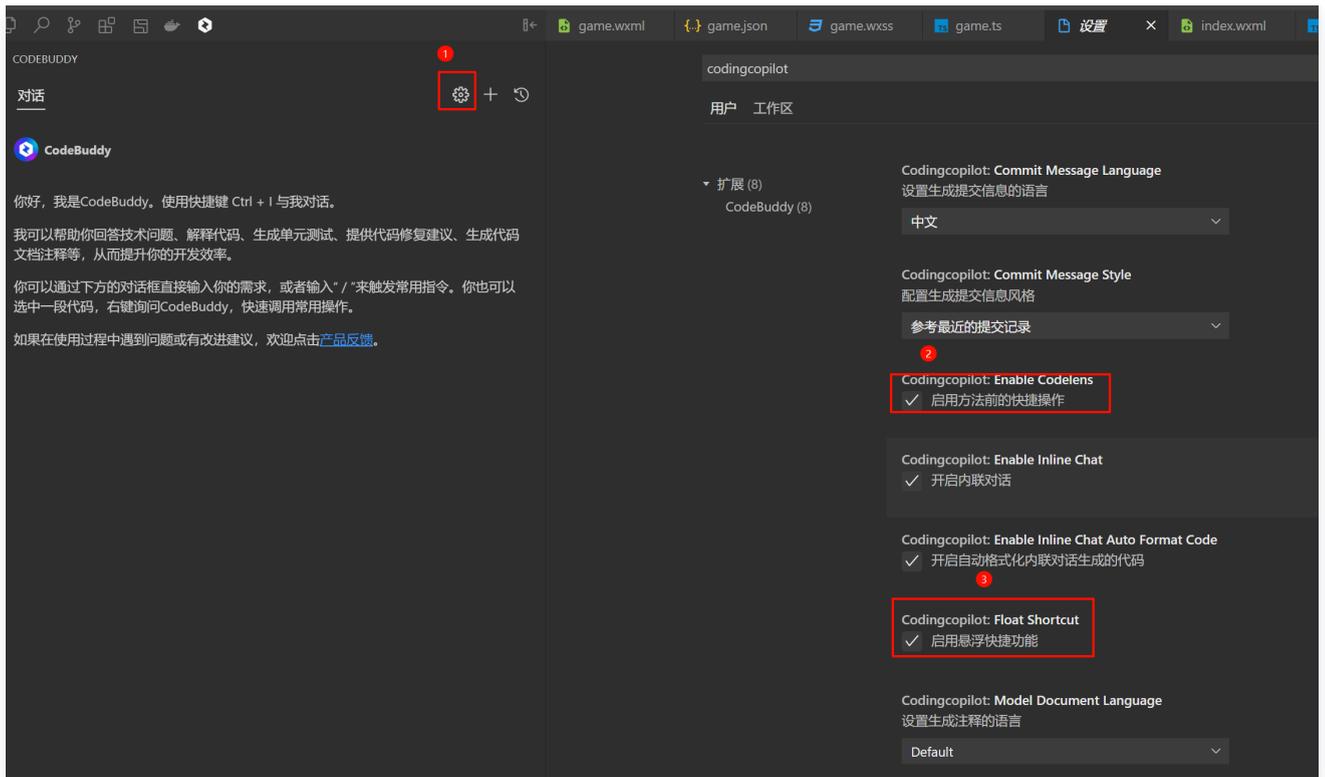
### 1. 生成文档注释。

腾讯云代码助手贴心的给出了三种方法生成文档注释：

- 方法一：通过圈选核心代码，右键，选择腾讯云代码助手下的生成文档，如下：



○ 方法二：进入插件的高级设置（下图圈1），开启圈2和圈3。



返回代码后, 可以在函数头或者鼠标悬浮后出现蓝色提示, 单击生成文档, 如下图:

```

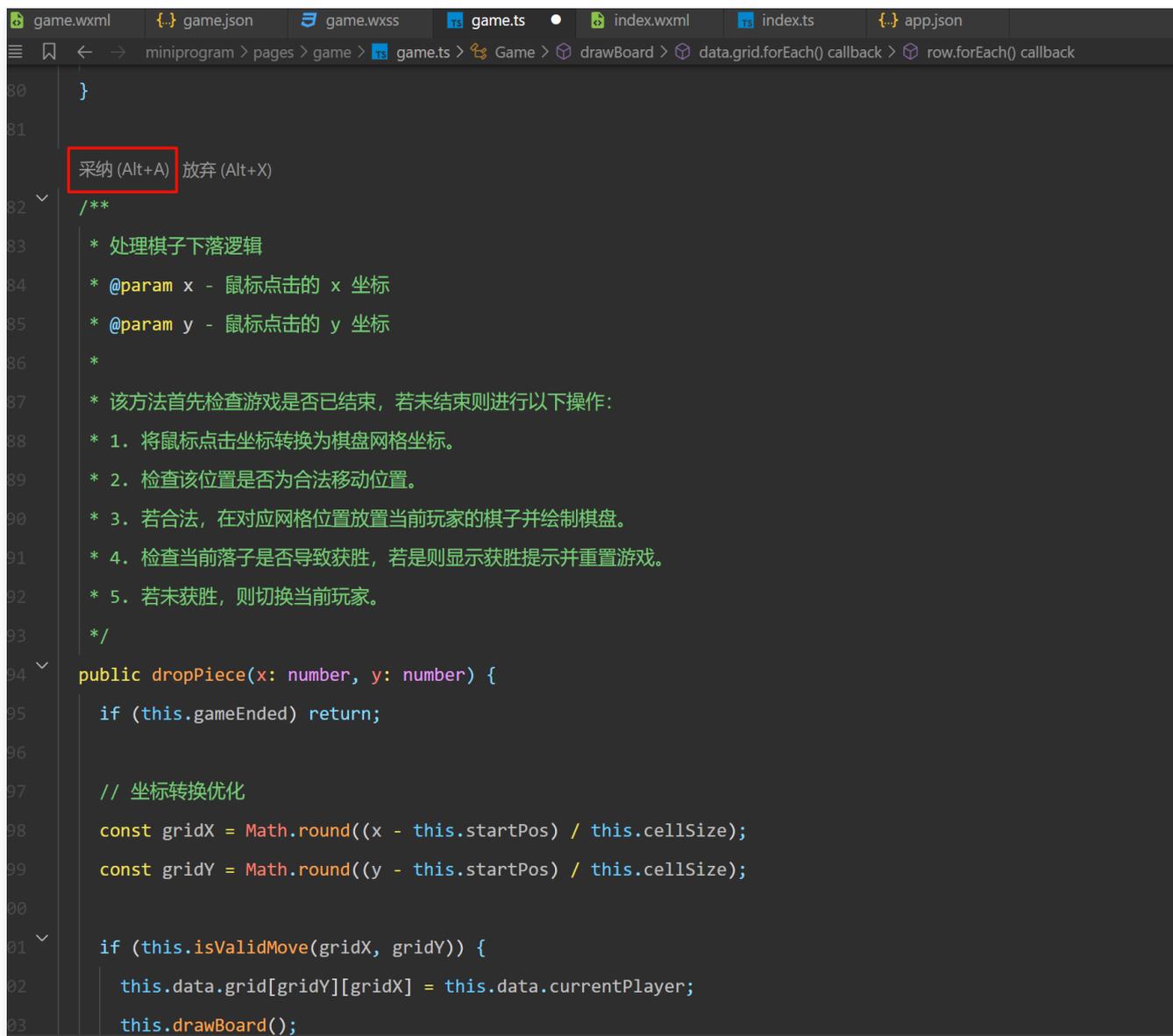
70     });
71
72     this.ctx.draw(true);
73   }
74
75   private drawStar(x: number, y: number) {
76     this.ctx.beginPath();
77     this.ctx.arc(x, y, 2, 0, Math.PI * 2);
78     this.ctx.setFillStyle('#000');
79     this.ctx.fill();
80   }
81
82   public dropPiece(x: number, y: number) {
83     if (this.gameEnded) return;
84
85     // 坐标转换优化
86     const gridX = Math.round((x - this.startPos) / this.cellSize);
87     const gridY = Math.round((y - this.startPos) / this.cellSize);
88
89     if (this.isValidMove(gridX, gridY)) {
90       this.data.grid[gridY][gridX] = this.data.currentPlayer;
91       this.drawBoard();
92     }

```

○ 方法三：直接使用快捷键，如下：

系统	快捷键
Mac	Opt+D
Windows	Shift+Alt+M

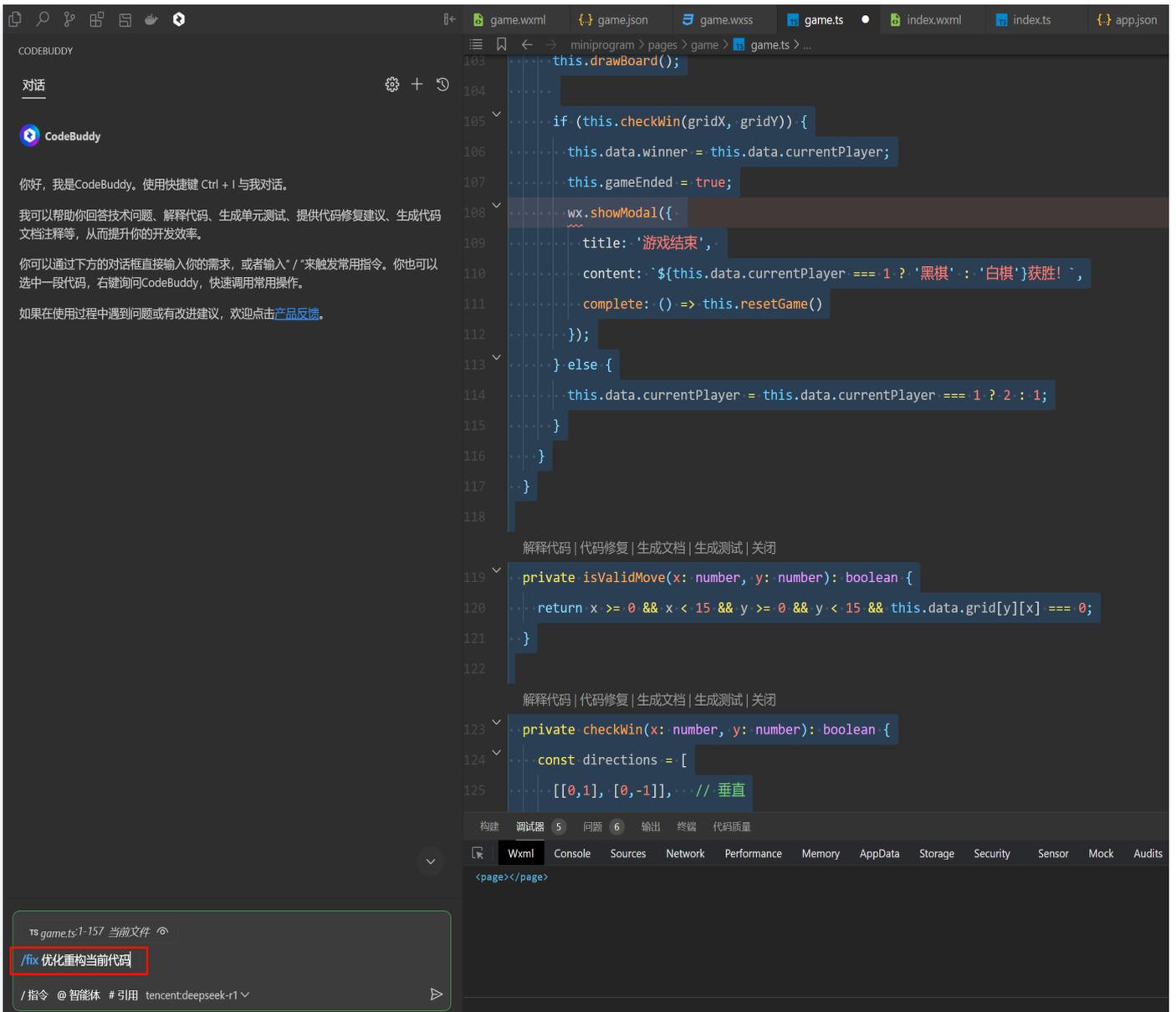
我们看看生成的文档效果如下，满意的话单击采纳就可以直接应用：



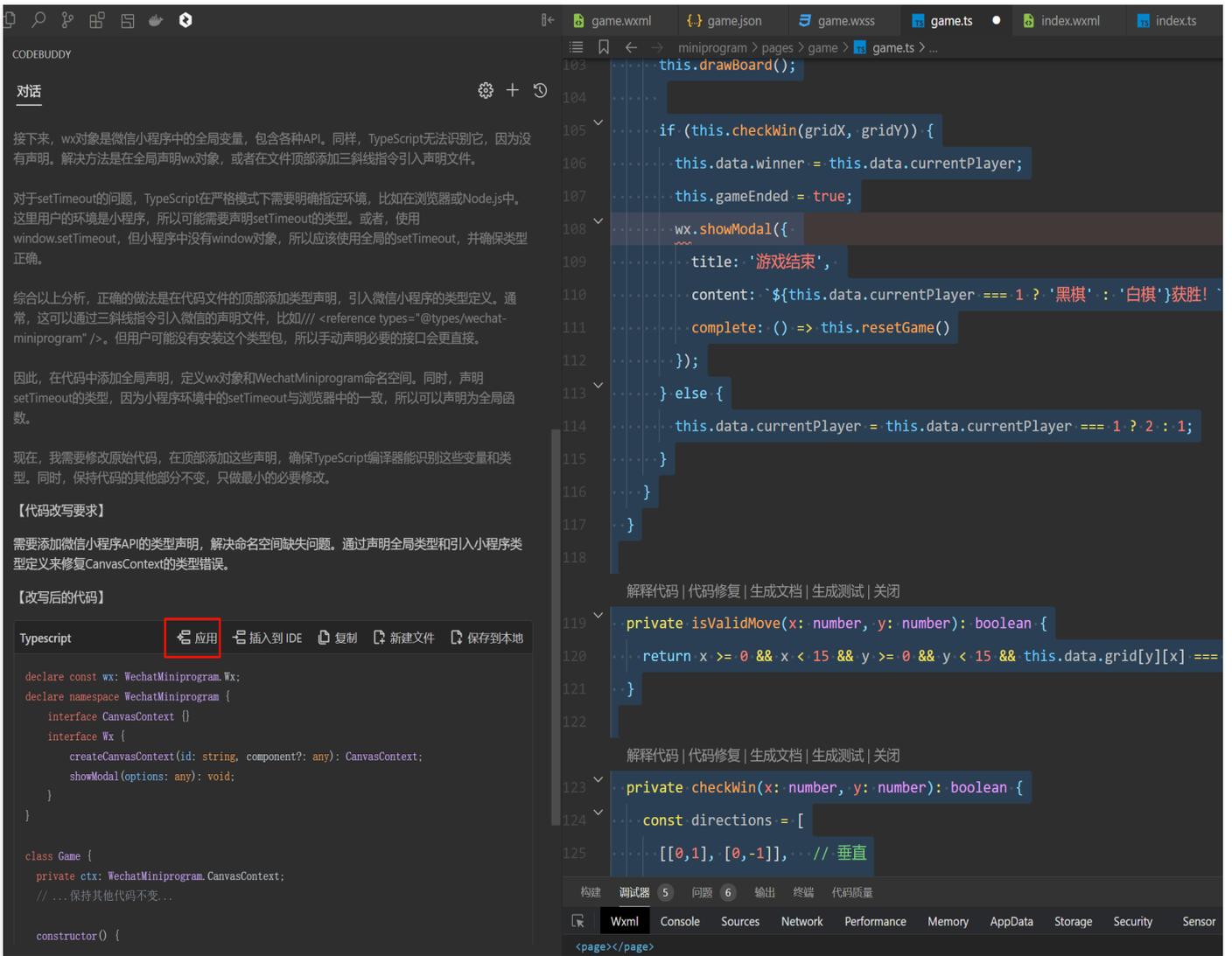
```
game.wxml  game.json  game.wxss  game.ts  index.wxml  index.ts  app.json
miniprogram > pages > game > game.ts > Game > drawBoard > data.grid.forEach() callback > row.forEach() callback
30 }
31
32 采纳 (Alt+A) 放弃 (Alt+X)
33 /**
34  * 处理棋子下落逻辑
35  * @param x - 鼠标点击的 x 坐标
36  * @param y - 鼠标点击的 y 坐标
37  *
38  * 该方法首先检查游戏是否已结束，若未结束则进行以下操作：
39  * 1. 将鼠标点击坐标转换为棋盘网格坐标。
40  * 2. 检查该位置是否为合法移动位置。
41  * 3. 若合法，在对应网格位置放置当前玩家的棋子并绘制棋盘。
42  * 4. 检查当前落子是否导致获胜，若是则显示获胜提示并重置游戏。
43  * 5. 若未获胜，则切换当前玩家。
44  */
45 public dropPiece(x: number, y: number) {
46     if (this.gameEnded) return;
47
48     // 坐标转换优化
49     const gridX = Math.round((x - this.startPos) / this.cellSize);
50     const gridY = Math.round((y - this.startPos) / this.cellSize);
51
52     if (this.isValidMove(gridX, gridY)) {
53         this.data.grid[gridY][gridX] = this.data.currentPlayer;
54         this.drawBoard();
55     }
56 }
```

## 2. 重构代码。

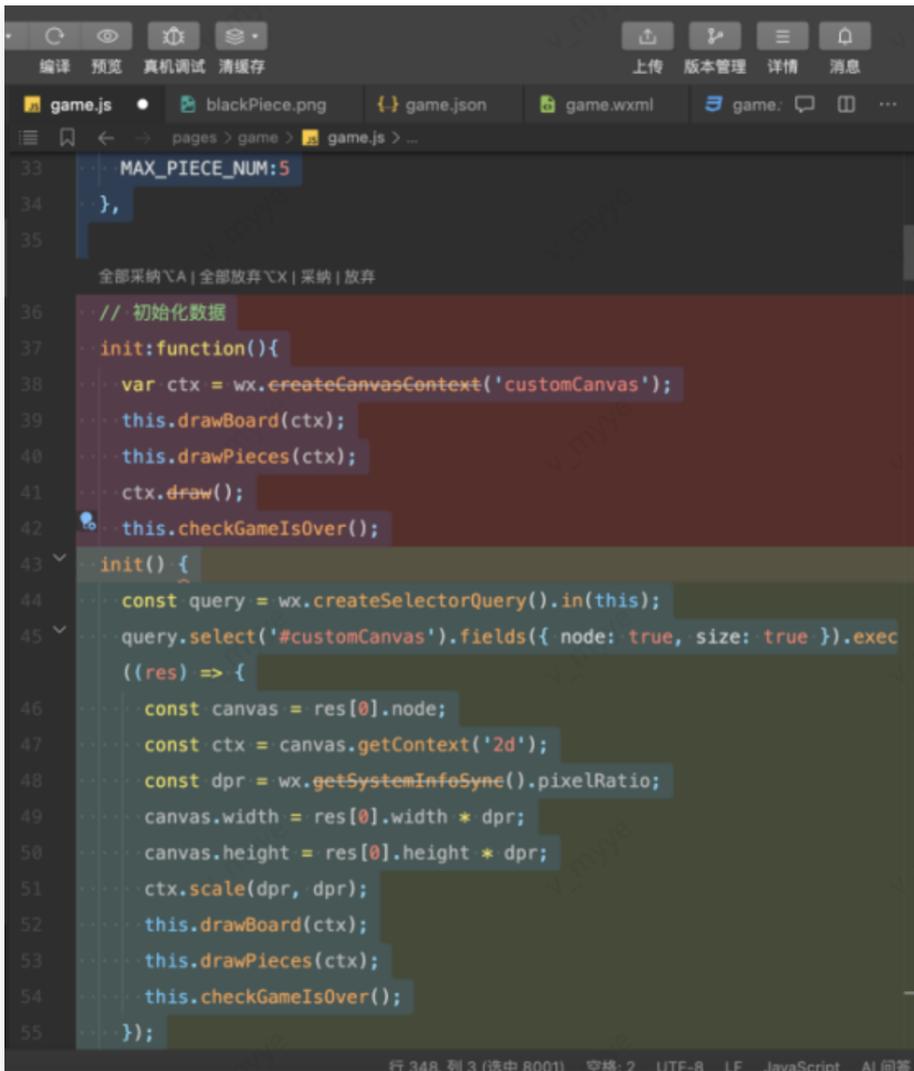
无论是语法错误，还是逻辑问题，或者是代码需要重构，一个 /fix 就可以帮我们 AI 检查。我们可以在对话框中使用 /fix 进行提问。



输出如下，我们可以选择应用代码。



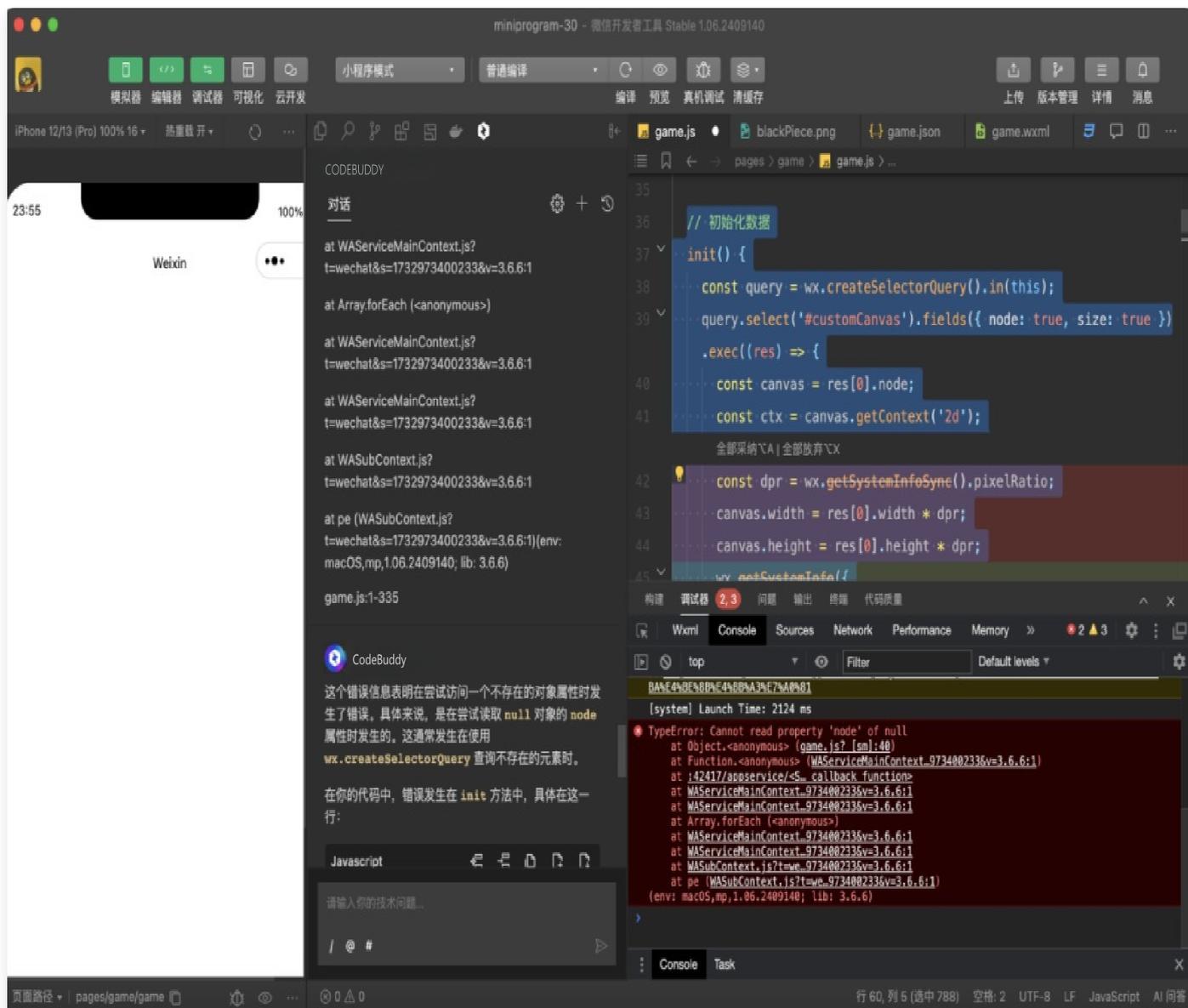
代码助手提供了智能插入的方法，通过大模型为其生成合并的 Diff View 预览效果，如下图，清晰很多了。代码助手在如下视图下可以选择部分采纳或者全部采纳：



```
33     MAX_PIECE_NUM: 5
34   },
35
36   // 初始化数据
37   init:function(){
38     var ctx = wx.createCanvasContext('customCanvas');
39     this.drawBoard(ctx);
40     this.drawPieces(ctx);
41     ctx.draw();
42     this.checkGameIsOver();
43
44   init() {
45     const query = wx.createSelectorQuery().in(this);
46     query.select('#customCanvas').fields({ node: true, size: true }).exec
47     ((res) => {
48       const canvas = res[0].node;
49       const ctx = canvas.getContext('2d');
50       const dpr = wx.getSystemInfoSync().pixelRatio;
51       canvas.width = res[0].width * dpr;
52       canvas.height = res[0].height * dpr;
53       ctx.scale(dpr, dpr);
54       this.drawBoard(ctx);
55       this.drawPieces(ctx);
56       this.checkGameIsOver();
57     });
58   }
59 }
```

### 3. 运行错误查询原因。

我们在开发中有时会发现如下图的错误，我们可以把错误信息直接复制给到代码助手（也可以通过圈选错误信息后右键选择腾讯云代码助手：解释代码），它能正确的给我结合当前工程代码进行分析，给出正确的解决方案。如下图，发现错误后，给出 init() 函数修复建议，通过对话代码区域的「智能插入」运行后，右边出现了 Diff View 预览，全部采纳后，问题解决。



## 总结

本文通过腾讯云代码助手来帮助开发五子棋的微信小程序游戏。可以发现，代码助手能够极大地帮助我们通过自然语言描述需求来生成工程代码，还可以帮助优化代码、生成代码注释等，是一个非常强大的编程辅助工具。同时，我们在与代码助手的对话过程中，注意要学会提示词的编写，良好的提示词能够让代码助手更容易理解需求，起到一个事半功倍的效果。

# 具体功能实践

## 跨文件能力扩展上下文

最近更新时间：2025-04-14 09:54:23

本文会基于一个实战的评测剖析，来介绍如何通过跨文件让补全内容来扩展更多的上下文，从而让生成代码质量更高更准。

为了提高生成质量，引入更多的上下文信息是很有必要的。在代码补全的场景中，除了当前文件的内容，还需要收集与之相关的更多代码作为上下文，并对提示词进行优化。本文将通过腾讯云代码助手的实例，详细介绍如何实时扩展上下文，获取更多跨文件的内容，以实现准确的代码补全。

### 如何让补全感知更多

整体思路是，考虑到开发人员在编码时会下意识地思考相关内容，所以设计也会比较简单。即优先选择打开的文件，通过算法判定这些文件和当前光标上下文的关联，从而给到模型进行排名和相似度计算。

所以，第一步是个工程问题。提取所有可能相关的上下文，可能是当前文件、导入的文件、同一目录中的文件等中的代码等。

接下来，第二步是重新排名和相似度计算。具体的选择思路如下：离线场景选取文件的时候，会根据文件结构进行选择，在目录结构中越紧邻的文件越容易被选到；上线场景选择文件的时候，可以根据用户在编辑器中打开的文件进行选择，选择用户最近打开的n个文件；代码片段的选择思路，是根据 TF-IDF（一种信息检索领域的常用算法）进行代码片段的相关性匹配，选取比较相关的一段代码。

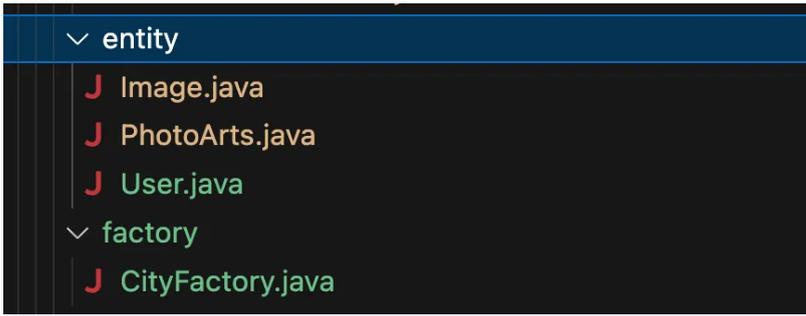
最后，通过按排名顺序添加上下文片段来构建提示，直到填充代码补全模型的最大上下文长度。还有一些其他细节可确保模型将这些理解为不同的信息片段，但一旦构建完成，这些细节就会被发送到模型进行推理。

举例：

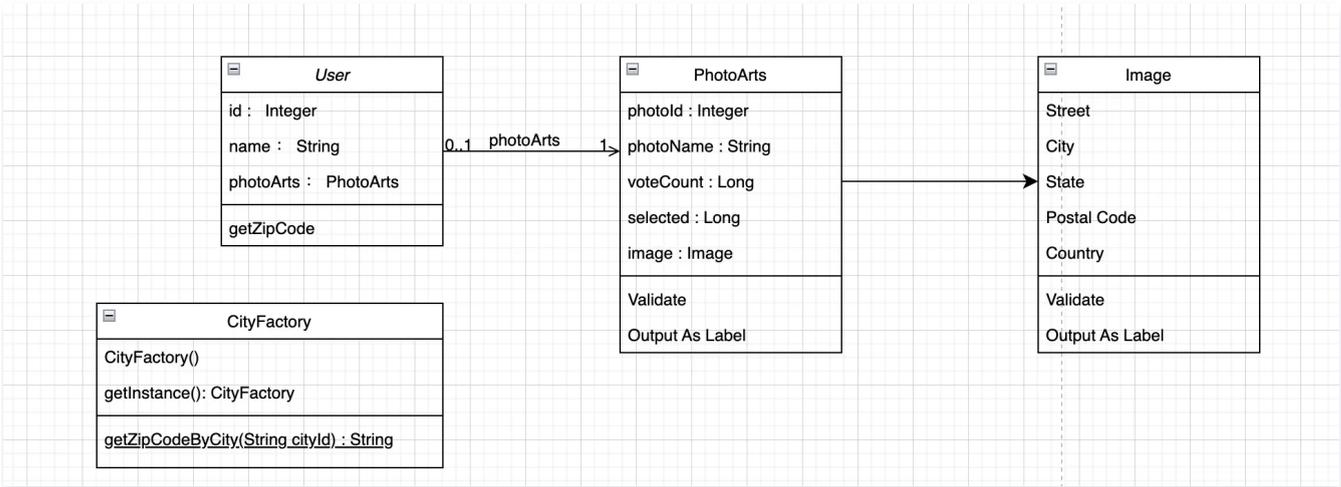
```
<reponame>reponame + <neighbor><filename>file_name<codeblock>code_block
* (n) +
<filename>filename<fim_prefix>code_prefix<fim_suffix>code_suffix<fim_mid
dle>code_middle<endoftext>
```

### 实战代码助手 – 跨文件能力

按以下步骤实践跨文件能力。下图是项目局部图，entity 定义了三个对象，他们彼此有对象关联。同时 User 类里的 ZipCode 需要从一个单例工厂中创建获得。



实体类大概是这样的：



### 任务一：申明几个对象关联关系，看补全里是否自动插入了另一个文件的函数

打开 User.java、PhotoArts.java、Image.java，在 UserServiceImpl.java 里创建一个函数，通过跨文件的延展，可以正确生成多个文件的函数调用链。

```
+ user.getPhotoArts().getRawImage().getImageName()+user.getPhotoArts().getRawImage().getImageData()
);
// check all attributes of user
```

视频如下：

[观看视频](#)

### 任务二：申明的单例模式的工厂类，看补全里是否正确获取调用单例工厂的函数

定义了一个 CityFactory，它是一个单例工厂，用于转 ZipCode 方法。代码如下：

```
public class CityFactory {
    private CityFactory() {
    }

    private static class CityFactoryHolder {
```

```
private static CityFactory INSTANCE = new CityFactory();
}

public static CityFactory getInstance() {
    if (CityFactoryHolder.INSTANCE == null) {
        synchronized (CityFactory.class) {
            if (CityFactoryHolder.INSTANCE == null) {
                CityFactoryHolder.INSTANCE = new CityFactory();
            }
        }
    }
    return CityFactoryHolder.INSTANCE;
}

public String getZipCodeByCity(String cityId) {
    // load from database
    // connect from database
    return "12345";
}
}
```

通过查看补全内容，可以确定代码助手正确的理解了工厂类的调用，而不是简单的实例化。

```
public class UserServiceImpl implements UserService {
    Logger logger = LoggerFactory.getLogger(clazz: UserServiceImpl.class);

    @Override
    public boolean createUserById(User user) {
        // check user zipcode
        String zipCode = CityFactory.getInstance().getZipCodeByCity(user.getCityName());
        if (zipCode == null) {
            logger.error("Invalid city name: {}", user.getCityName());
            return false;
        }

        return true;
    }
}
```

## 实战演示效果

[观看视频](#)

## 总结

---

本文用简单的示例，来看出跨文件补全的产品价值。结合中间补全（Fill-in-Middle）机制，并扩展上下文的窗口，引入更多的想关联的文件，从而让补全生成更精准，也更接近对项目工程当下的情境理解。

# 代码助手与终端的结合应用实践

最近更新时间：2025-04-14 09:54:23

AI 辅助开发的大背景下，除了代码生成，在编辑器自带的终端模式下，同样需要 AI 相关的能力，从而可以帮助程序员在终端运行出现报错的场景下，可以快速查找到问题解决方案；也可以帮助程序员不需要记住太多的终端命令。本文旨在介绍腾讯云代码助手，如何与终端结合应用，为代码开发提供便利。

## 在对话中，面向终端 @terminal 提问

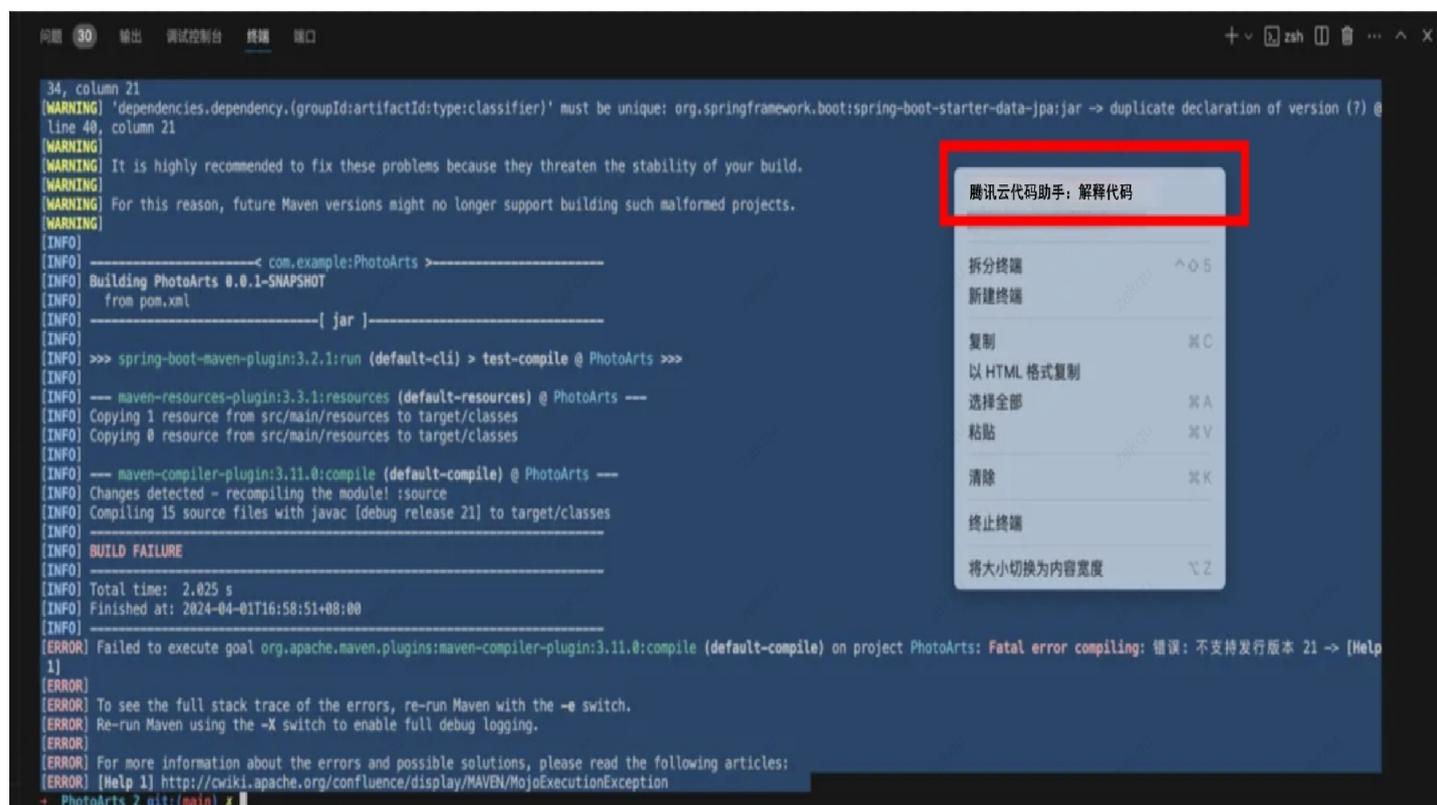
代码助手支持四种 agent 模式：

- @workspace：面向项目工程下的提问解答
- @vscode：面向 vscode 帮助文档的提问解答。
- @terminal：面向终端的提问解答。
- 拓展 agent，支持用户自行定义 agent，这里不做过多展开。

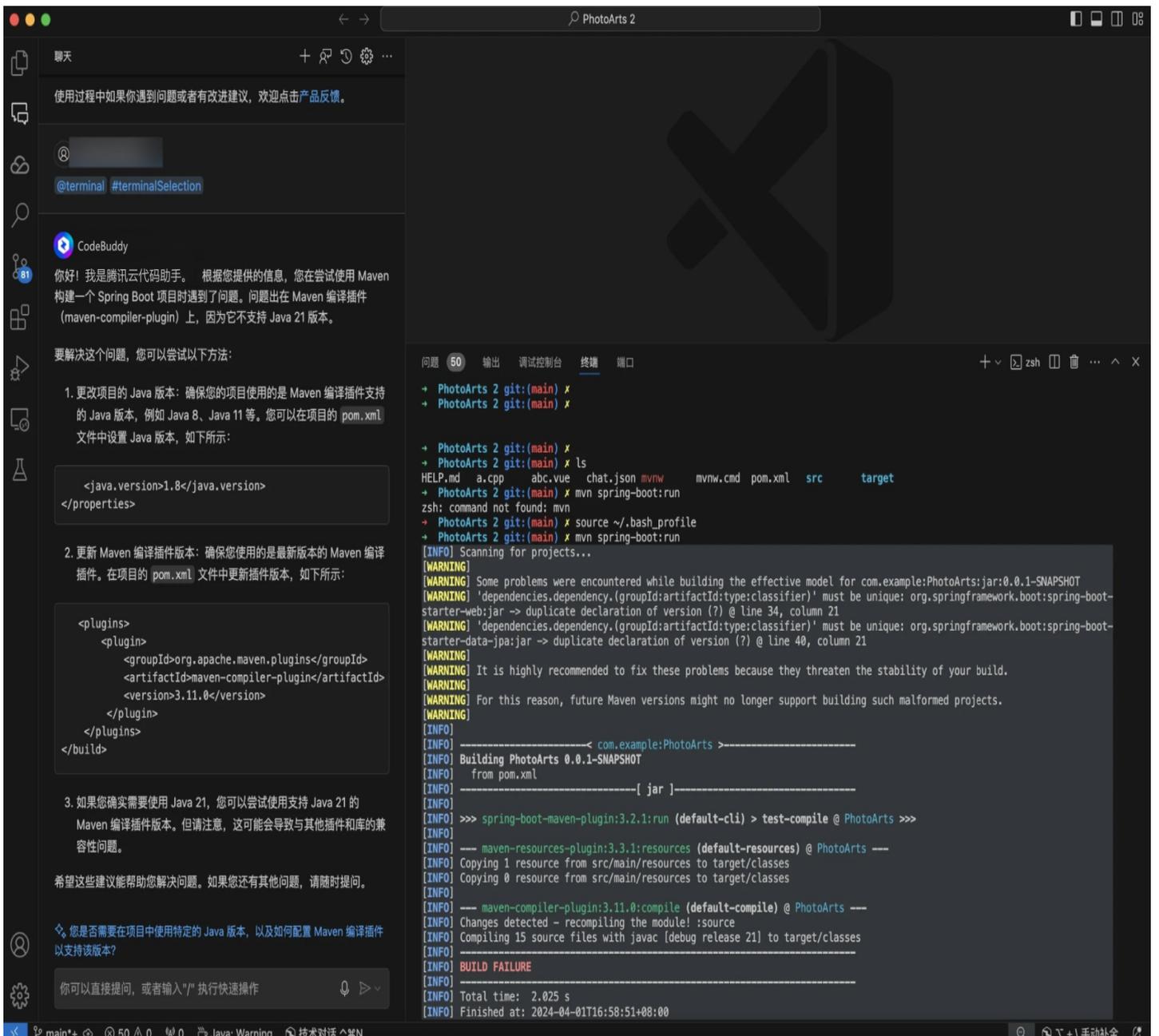
本部分将对 @terminal 面向终端的提问解答，进行实践演示。

## 获取终端中错误信息的相关帮助

当终端运行产生错误信息时，只需选中错误消息，右键单击，然后选择腾讯云代码助手：解释代码。



代码助手将为您提供错误描述和建议的修复方案。



当然您也可以直接在对话框中输入 @terminal，譬如，@terminal 安装 maven，如下图，他会找到和终端相关的答案。

文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) 终端(T) 帮助(H)
swarm [管理员]

CODEBUDDY

对话 Craft 评审 单测生成

您可以通过下方的对话框直接输入你的需求，或者输入“/”来触发常用指令。你也可以选中一段代码，右键询问CodeBuddy，快速调用常用操作。

如果在使用过程中遇到问题或有改进建议，欢迎点击产品反馈。

1 @terminal 如何安装maven

CodeBuddy

要在您的系统上安装Maven，请按照以下步骤操作：

**对于Windows用户：**

- 访问Maven官方网站下载页面：<https://maven.apache.org/download.cgi>
- 下载最新版本的 `apache-maven-x.x.x-bin.zip` (x.x.x代表版本号)。
- 解压下载的ZIP文件到你选择的目录，例如 `C:\Program Files\Apache\maven`。
- 将Maven的 `bin` 目录添加到系统环境变量 `PATH` 中。例如：`C:\Program Files\Apache\maven\bin`。
- 打开命令提示符 (CMD)，输入 `mvn -version` 来验证安装是否成功。

**对于macOS用户：**

- 使用Homebrew安装Maven，打开终端并输入：`brew install maven`。
- 如果没有安装Homebrew，可以先安装它，访问<https://brew.sh/>并按照指示操作。
- 安装完成后，输入 `mvn -version` 来验证安装。

**对于Linux用户：**

- 打开终端。
- 使用包管理器安装Maven，例如在Ubuntu上可以使用：`sudo apt-get install maven`。
- 安装完成后，输入 `mvn -version` 来验证安装。

安装完成后，您可以在命令行中使用 `mvn` 命令来构建和管理Java项目。

如何在Spring Boot中创建一个简单的RESTful API?

evals.py 当前文件

请输入你的技术问题...

/指令 @智能体 #引用 hunyuan

```

swarm-main > examples > triage_agent > evals.py > ...
6
7 client = Swarm()
8
9 CONVERSATIONAL_EVAL_SYSTEM_PROMPT = """
10 You will be provided with a conversation between a user and an a
11 Your goal is to evaluate, based on the conversation, if the ager
12
13 To assess whether the agent manages to achieve the main goal, co
14 is the answer satisfactory for the user or not, could the agent
15 It is possible that the user is not satisfied with the answer, b
16 """
17
18 解释代码 | 代码修复 | 生成文档 | 生成测试 | 代码评审 | 关闭
19 def conversation_was_successful(messages) -> bool:
20     conversation = f"CONVERSATION: {json.dumps(messages)}"
21     result: BoolEvalResult = evaluate_with_llm_bool(
22         CONVERSATIONAL_EVAL_SYSTEM_PROMPT, conversation
23     )
24     return result.value
25
26 解释代码 | 代码修复 | 生成文档 | 生成测试 | 代码评审 | 关闭
27 def run_and_get_tool_calls(agent, query):
28     message = {"role": "user", "content": query}
29     response = client.run(
30         agent=agent,
31         messages=[message],
32         execute_tools=False,
33     )
34     return response.messages[-1].get("tool_calls")
35
36 Alt+I 生成代码, Ctrl+I 打开对话, Ctrl+Shift+Enter 触发下个
37 解释代码 | 代码修复 | 生成文档 | 生成测试 | 代码评审 | 关闭
38 @pytest.mark.parametrize(
39     "query,function_name",
40     [
41         ("I want to make a refund!", "transfer_to_refunds"),
42         ("I want to talk to sales.", "transfer_to_sales"),
43     ]
44 )
45
46 问题 1 输出 调试控制台 终端 端口 评论
47
48 目录: C:\Users\v_myje\Downloads\swarm\swarm-main\examples\personal_sh
49
50 Mode                LastWriteTime         Length Name
51 ----                -
52 -a----              2025/4/5 17:57          4463 database.py
53 -a----              2025/4/5 17:57          4927 main.py
54 -a----              2025/4/5 17:57           870 README.md
55 -a----              2025/4/5 17:57           0 __init__.py
56
57 PS C:\Users\v_myje\Downloads\swarm\swarm-main\examples\personal_shopper>
58 Traceback (most recent call last):
59   File "C:\Users\v_myje\Downloads\swarm\swarm-main\examples\personal_shop
60     from swarm import Agent
61 ModuleNotFoundError: No module named 'swarm'
62 PS C:\Users\v_myje\Downloads\swarm\swarm-main\examples\personal_shopper>

```

# 单元测试应用实践

最近更新时间：2025-04-14 09:54:23

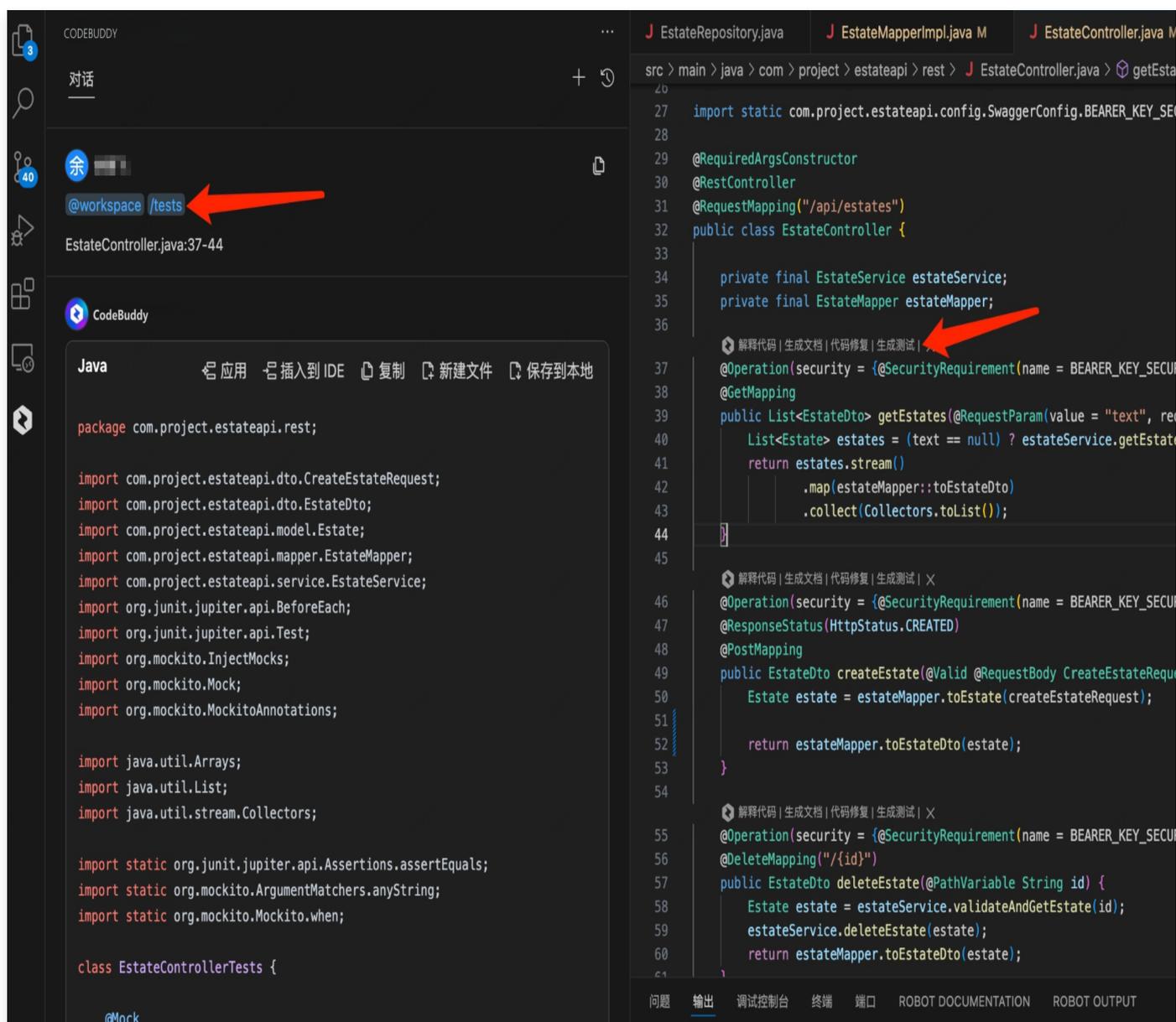
单元测试专注于程序中的最小可测试单元，这通常包括函数、类的方法或模块等。其核心目的是确保这些基本单元能够正确执行特定的功能，并妥善处理各种边界条件和异常情况。单元测试的目的是确保这些基本组成单元按照预期工作，能够正确地执行特定的功能，并且能够妥善处理各种边界条件和异常情况。

以下是腾讯云代码助手在单元测试中的应用实践：

## AI 在单元测试中的应用

### 1. 代码补全与单元测试生成。

腾讯云代码助手基于上下文理解，能够自动推荐最可能的代码片段，包括方法调用、变量声明、循环结构等。此外，它还能根据现有代码结构，自动生成对应的单元测试案例，确保代码变更时功能的稳定性。



The screenshot displays the CodeBuddy interface within an IDE. On the left, a chat window shows the user's request: "@workspace /tests" with a red arrow pointing to it. Below the chat, the CodeBuddy assistant has generated a Java test class named `EstateControllerTests`. The test class includes imports for `org.junit.jupiter.api.Assertions`, `org.mockito.ArgumentMatchers`, and `org.mockito.Mockito`. The test class structure is as follows:

```
class EstateControllerTests {  
    @Mock  
    EstateService estateService;  
    EstateMapper estateMapper;  
    EstateRepository estateRepository;  
    EstateController controller;  
    @BeforeEach  
    void setUp() {  
        estateService = Mockito.mock(EstateService.class);  
        estateMapper = Mockito.mock(EstateMapper.class);  
        estateRepository = Mockito.mock(EstateRepository.class);  
        controller = new EstateController(estateService, estateMapper, estateRepository);  
    }  
    @Test  
    void testGetEstates() {  
        // Test logic for getEstates method  
    }  
}
```

On the right, the IDE shows the source code for `EstateController.java`. A red arrow points to the `getEstates` method, which is annotated with `@GetMapping("/api/estates")`. The method implementation is:

```
public List<EstateDto> getEstates(@RequestParam(value = "text", required = false) String text) {  
    List<Estate> estates = (text == null) ? estateService.getEstates() : estateService.getEstatesByText(text);  
    return estates.stream()  
        .map(estateMapper::toEstateDto)  
        .collect(Collectors.toList());  
}
```

Below the code, the CodeBuddy assistant has generated a `@PostMapping` test case for the `createEstate` method, annotated with `@ResponseStatus(HttpStatus.CREATED)`. The test case is:

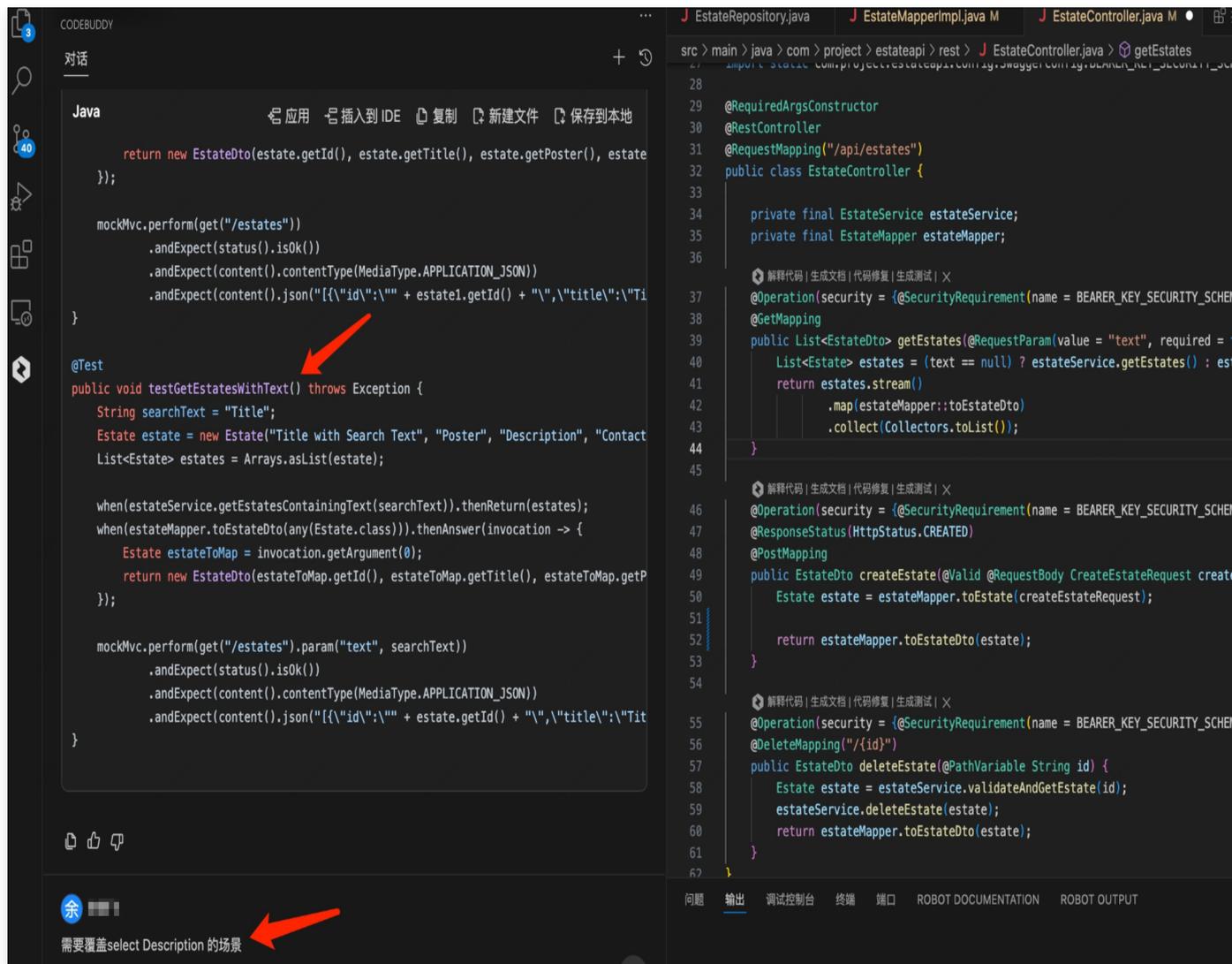
```
@Test  
void testCreateEstate() {  
    CreateEstateRequest request = new CreateEstateRequest("1234567890");  
    EstateDto response = controller.createEstate(request);  
    assertEquals(HttpStatus.CREATED, response.getStatus());  
}
```

如上图所示，可以通过对话框中的 /test 生成单元测试，或者使用 IDE 编码区域中使用每个功能方法的快捷键生成测试来生成该方法的单元测试。

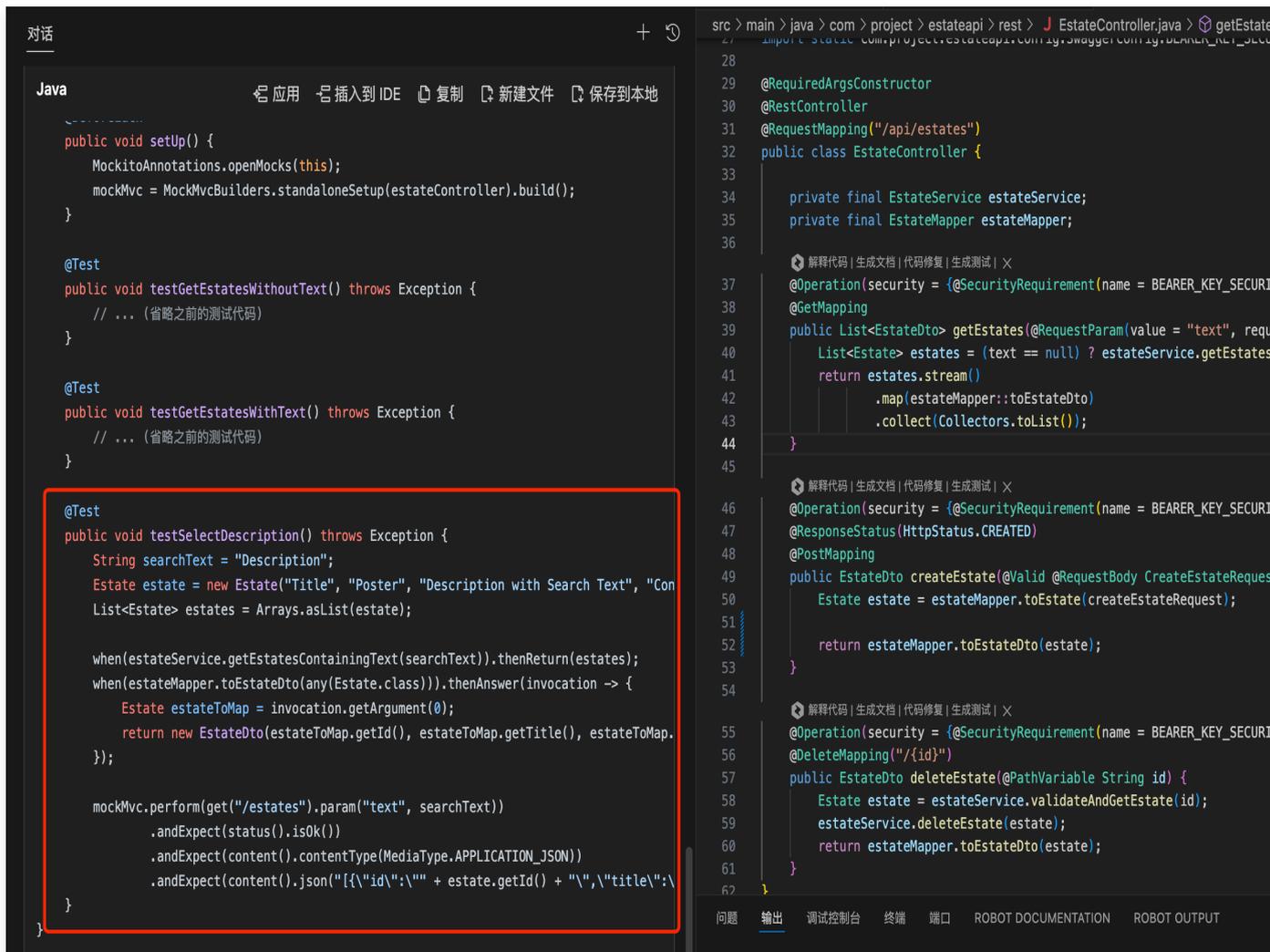
## 2. 多轮对话优化测试用例。

开发者可以通过多轮对话，告诉代码助手更多的信息，让生成的单元测试更符合业务预期。例如，可以指定业务边界条件、特殊的异常处理逻辑、数据处理方式等等。

对于生成的内容，如果有额外的测试场景需求，可以通过多轮对话追加提问。如下图所示：

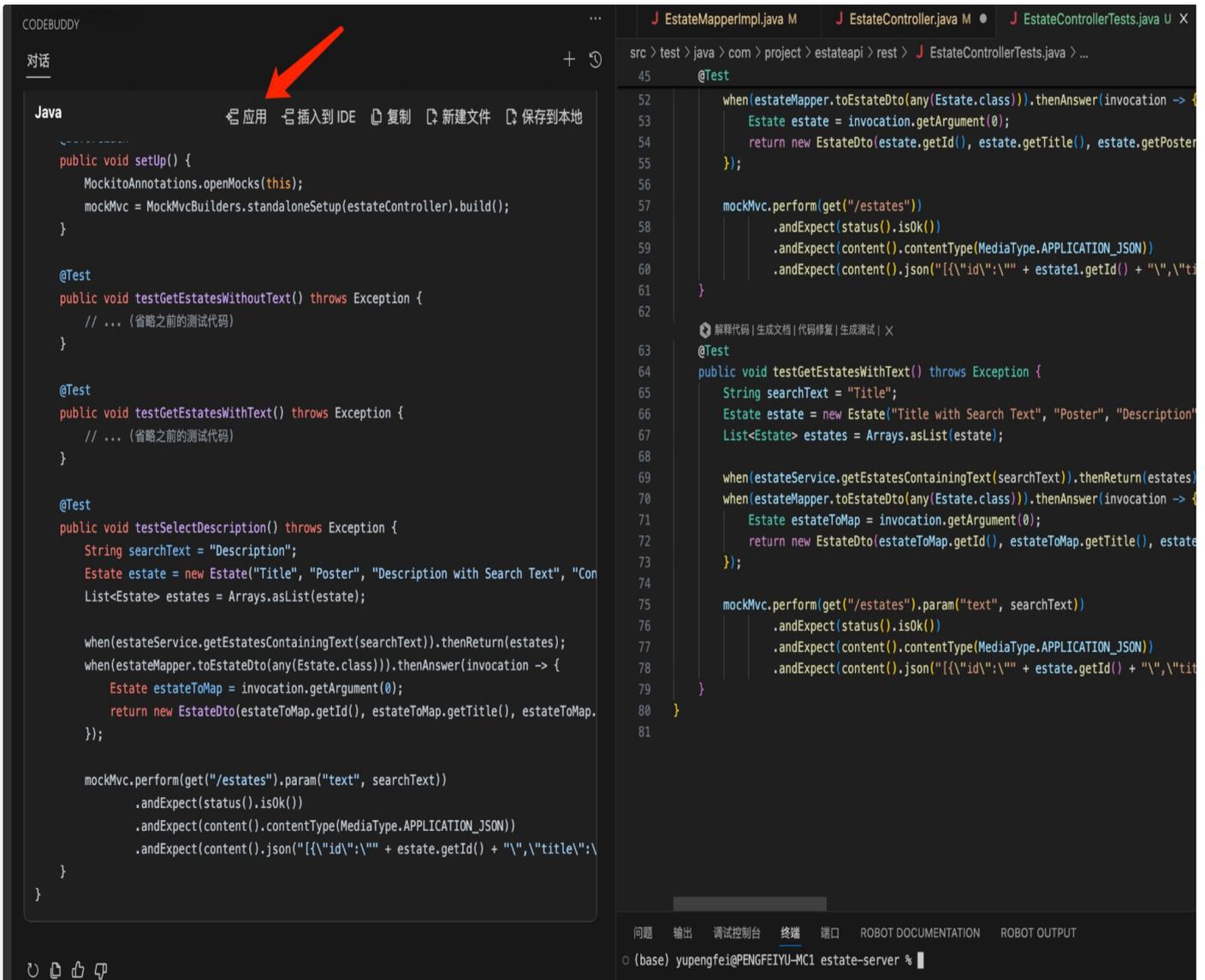


代码助手的对话模型会识别用户意图、结合上下文对话内容优化单元测试结果。如下图所示：



### 3. 一键应用生成的单元测试内容。

通过对话框生成结果中的快捷按钮，包括应用、插入到 IDE、复制、新建文件、保存到本地，开发者可以快速判断和接受生成的单元测试代码。如下图所示：



例如可以选择应用，将对话生成的结果，以 diff 的形式应用到对应的代码文件中，可以做出采纳或放弃的选择处理。如下图所示：

```

EstateControllerTest.java 已完成
@BeforeEach
public void setUp() {
    MockitoAnnotations.openMocks(this);
    mockMvc = MockMvcBuilders.standaloneSetup(estateController).build();
}

@Test
public void testGetEstatesWithoutText() throws Exception {
    // ... (省略之前的测试代码)
}

@Test
public void testGetEstatesWithText() throws Exception {
    // ... (省略之前的测试代码)
}

@Test
public void testSelectDescription() throws Exception {
    String searchText = "Description";
    Estate estate = new Estate("Title", "Poster", "Description with Search Text", "Contact", 1500);
    List<Estate> estates = Arrays.asList(estate);

    when(estateService.getEstatesContainingText(searchText)).thenReturn(estates);
    when(estateMapper.toEstateDto(any(Estate.class))).thenAnswer(invocation -> {
        Estate estateToMap = invocation.getArgument(0);
        return new EstateDto(estateToMap.getId(), estateToMap.getTitle(), estateToMap.getPoster());
    });

    mockMvc.perform(get("/estates").param("text", searchText))
        .andExpect(status().isOk())
        .andExpect(content().contentType(MediaType.APPLICATION_JSON))
        .andExpect(content().json("{\"id\":\"" + estate.getId() + "\", \"title\":\"" + estate.getTitle() + "\", \"poster\":\"" + estate.getPoster() + "\"}"));
}

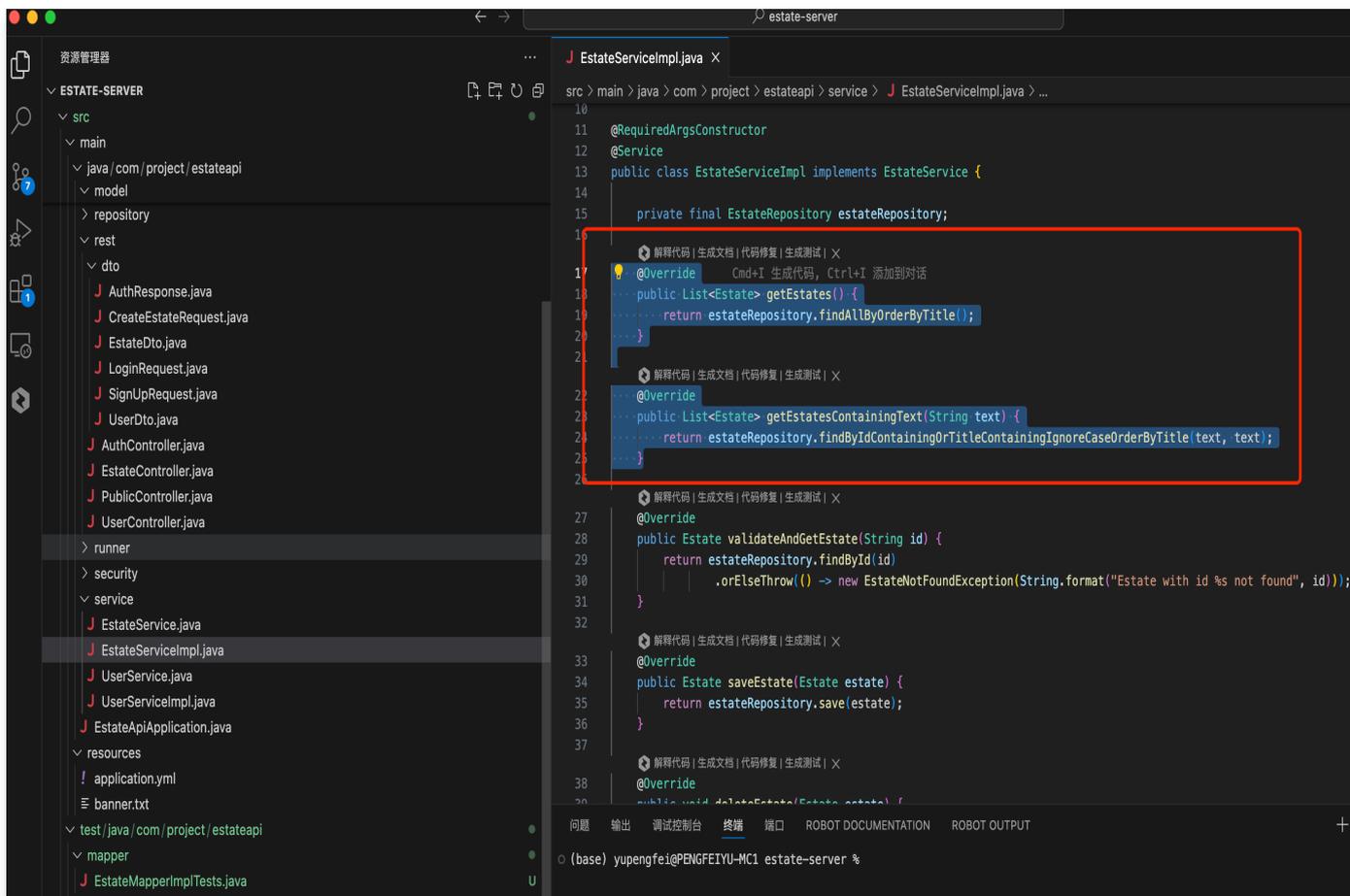
EstateControllerTests.java U
src > test > java > com > project > estateapi > rest > EstateControllerTests.java > ...
63 @Test
64 public void testGetEstatesWithText() throws Exception {
65     String searchText = "Title";
66     Estate estate = new Estate("Title with Search Text", "Poster", "Description", "Contact", 1500);
67     List<Estate> estates = Arrays.asList(estate);
68
69     when(estateService.getEstatesContainingText(searchText)).thenReturn(estates);
70     when(estateMapper.toEstateDto(any(Estate.class))).thenAnswer(invocation -> {
71         Estate estateToMap = invocation.getArgument(0);
72         return new EstateDto(estateToMap.getId(), estateToMap.getTitle(), estateToMap.getPoster());
73     });
74
75     mockMvc.perform(get("/estates").param("text", searchText))
76         .andExpect(status().isOk())
77         .andExpect(content().contentType(MediaType.APPLICATION_JSON))
78         .andExpect(content().json("{\"id\":\"" + estate.getId() + "\", \"title\":\"Title with Search Text\", \"poster\":\"Poster\", \"description\":\"Description\", \"contact\":\"Contact\", \"price\":1500}"));
79 }
80
81 @Test
82 public void testSelectDescription() throws Exception {
83     String searchText = "Description";
84     Estate estate = new Estate("Title", "Poster", "Description with Search Text", "Contact", 1500);
85     List<Estate> estates = Arrays.asList(estate);
86
87     when(estateService.getEstatesContainingText(searchText)).thenReturn(estates);
88     when(estateMapper.toEstateDto(any(Estate.class))).thenAnswer(invocation -> {
89         Estate estateToMap = invocation.getArgument(0);
90         return new EstateDto(estateToMap.getId(), estateToMap.getTitle(), estateToMap.getPoster());
91     });
92
93     mockMvc.perform(get("/estates").param("text", searchText))
94         .andExpect(status().isOk())
95         .andExpect(content().contentType(MediaType.APPLICATION_JSON))
96         .andExpect(content().json("{\"id\":\"" + estate.getId() + "\", \"title\":\"Title\", \"poster\":\"Poster\", \"description\":\"Description with Search Text\", \"contact\":\"Contact\", \"price\":1500}"));
97 }
98
99 }
    
```

## 实际编码案例解析

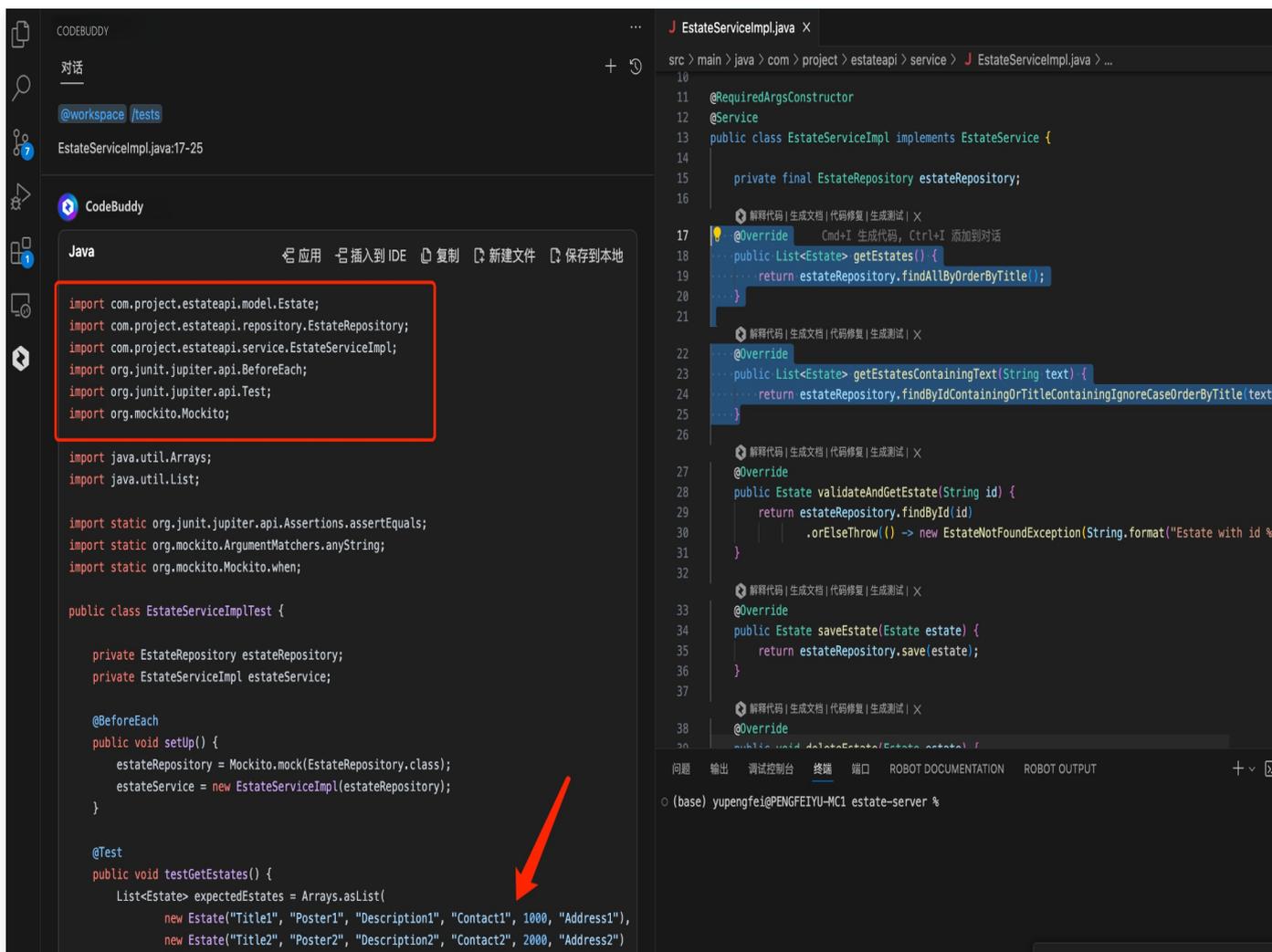
### 1. 针对业务代码准确生成单元测试。

在真实的开发场景中，软件的功能都是异常复杂的，一个完整的软件功能由多个类、方法共同组成。以 Java 开发为例，一个功能的开发，可能由 类的定义、Controller 控制层的方法、Server 接口定义、ServiceImpl 接口实现类方法、Repository 和 DTO 等等部分组成。在生成单元测试的时候，需要考虑到真实业务中的各个组件关联性。

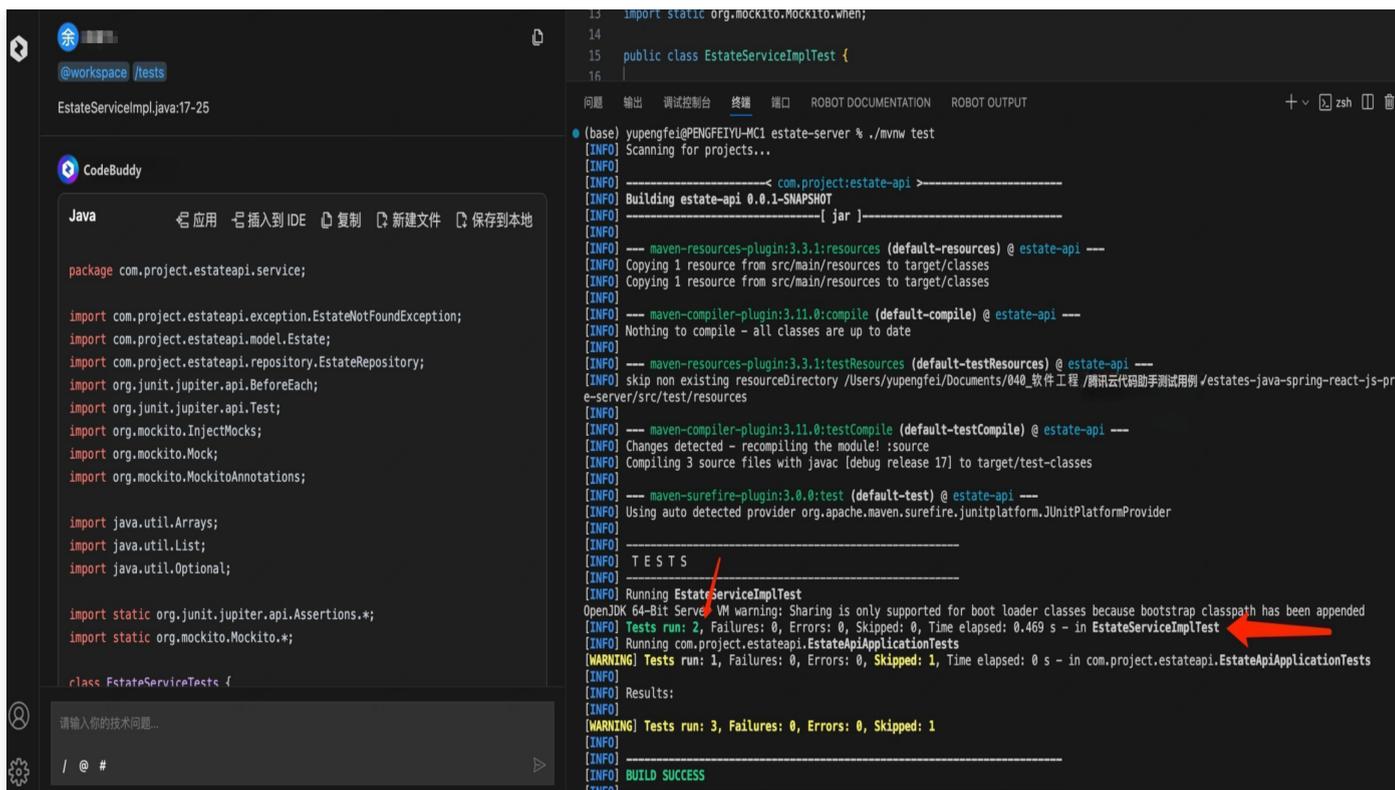
接下来，我们以一个 Java 工程为例，需要代码助手帮我们准确生成接口实现类 EstateServiceImpl.java 中的查询和模糊查询方法对应的单元测试代码。



我们选中 `getEstates` 和 `getEstatesContainingText` 方法之后，使用 `/test` 为其生成单元测试代码。在结果中可以看到，腾讯云代码助手在生成单元测试的时候，会分析代码的工程结构，并在生成的时候智能 import 相关依赖；在生成测试代码的时候，会结合其他文件中的类和方法定义生成测试计划，并准确构造测试数据。如下图所示：



将生成的单元测试内容，插入到指定单元测试文件中，可以直接执行 `mvn test` 命令进行单元测试验证，如下图所示可以看到这里生成的代码可以编译通过，并完成单元测试功能。



## 2. 结合业务逻辑逐步完善测试结果。

在真实的业务开发过程中，有很多测试场景都是隐晦的，它的逻辑可能并没有直接定义在代码内容中，而是需要结合实际需求而定。这种场景下，需要研发人员使用对话功能告诉代码助手这里的业务测试逻辑，继而完善测试代码。

我们可以通过多轮对话能力，继续输入真实业务场景，丰富测试的边界条件，让 AI 持续优化单元测试结果。如下图所示：

对话 + ↻

```
Java 🔗 应用  📄 插入到 IDE  📄 复制  📄 新建文件  📄 保存到本地
```

```
// Act
List<Estate> actualEstates = estateService.getEstatesContainingText(searchText);

// Assert
assertEquals(expectedEstates, actualEstates);
verify(estateRepository, times(1)).findByIdContainingOrTitleContainingIgnoreCaseOrderByTitle(searchT
}
}
```

📄 👍 🔗

**余** 📄

getEstatesContainingText 这个方法在处理的时候，只需要识别 标题 即可，当用户输入的是标题的时候，可以准确检索到。当用户输入的是其他字段的时候，即使字段匹配，也无法获取结果。这里在生成测试用例的时候，需要考虑到 非标题字段（比如描述字段）匹配，但是仍然搜索不到的情况。

EstateServiceImpl.java:23-23

我们将生成的扩展内容应用到之前的单元测试代码中，并执行下这里生成的最终结果，可以看到执行成功。

```

J EstateServiceImpl.java  J EstateServiceImplTest.java U X
src > test > java > com > project > estateapi > service > J EstateServiceImplTest.java > getEstatesContainingText_ShouldNotReturnEstatesMatchingNonTitleFields
64  @Test
78      // Assert
79      assertEquals(expectedEstates, actualEstates);
80      verify(estateRepository, times(1)).findByIdContainingOrTitleContainingIgnoreCaseOrderByTitle(searchText, searchText);
81  }
82
83  @Test
84  void getEstatesContainingText_ShouldNotReturnEstatesMatchingNonTitleFields() {
85      // Arrange
86      String searchText = "description";
87      List<Estate> expectedEstates = Arrays.asList();
88      when(estateRepository.findByIdContainingOrTitleContainingIgnoreCaseOrderByTitle(searchText, searchText))
89          .thenReturn(expectedEstates);
90
91      // Act
92      List<Estate> actualEstates = estateService.getEstatesContainingText(searchText);
93
94      // Assert
95      assertEquals(expectedEstates, actualEstates);
96      verify(estateRepository, times(1)).findByIdContainingOrTitleContainingIgnoreCaseOrderByTitle(searchText, searchText);
97  }
98  }
99
问题  输出  调试控制台  终端  端口  ROBOT DOCUMENTATION  ROBOT OUTPUT
[INFO] skip non existing resourceDirectory /Users/yupengfei/Documents/040_软件工程 /腾讯云代码助手测试用例 /estates-java-spring-react-js-project/estate-server/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.11.0:testCompile (default-testCompile) @ estate-api ---
[INFO] Changes detected - recompiling the module! :source
[INFO] Compiling 3 source files with javac [debug release 17] to target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:3.0.0:test (default-test) @ estate-api ---
[INFO] Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
[INFO]
[INFO]
[INFO] T E S T S
[INFO]
[INFO] Running EstateServiceImplTest
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.501 s - in EstateServiceImplTest
[INFO] Running com.project.estateapi.EstateApiApplicationTests
[WARNING] Tests run: 1, Failures: 0, Errors: 0, Skipped: 1, Time elapsed: 0 s - in com.project.estateapi.EstateApiApplicationTests
[INFO]
[INFO] Results:
[INFO]
[WARNING] Tests run: 5, Failures: 0, Errors: 0, Skipped: 1
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 1.681 s
[INFO] Finished at: 2024-11-10T23:35:38+08:00
[INFO]
(base) yupengfei@PENGFEIYU-MC1 estate-server %

```

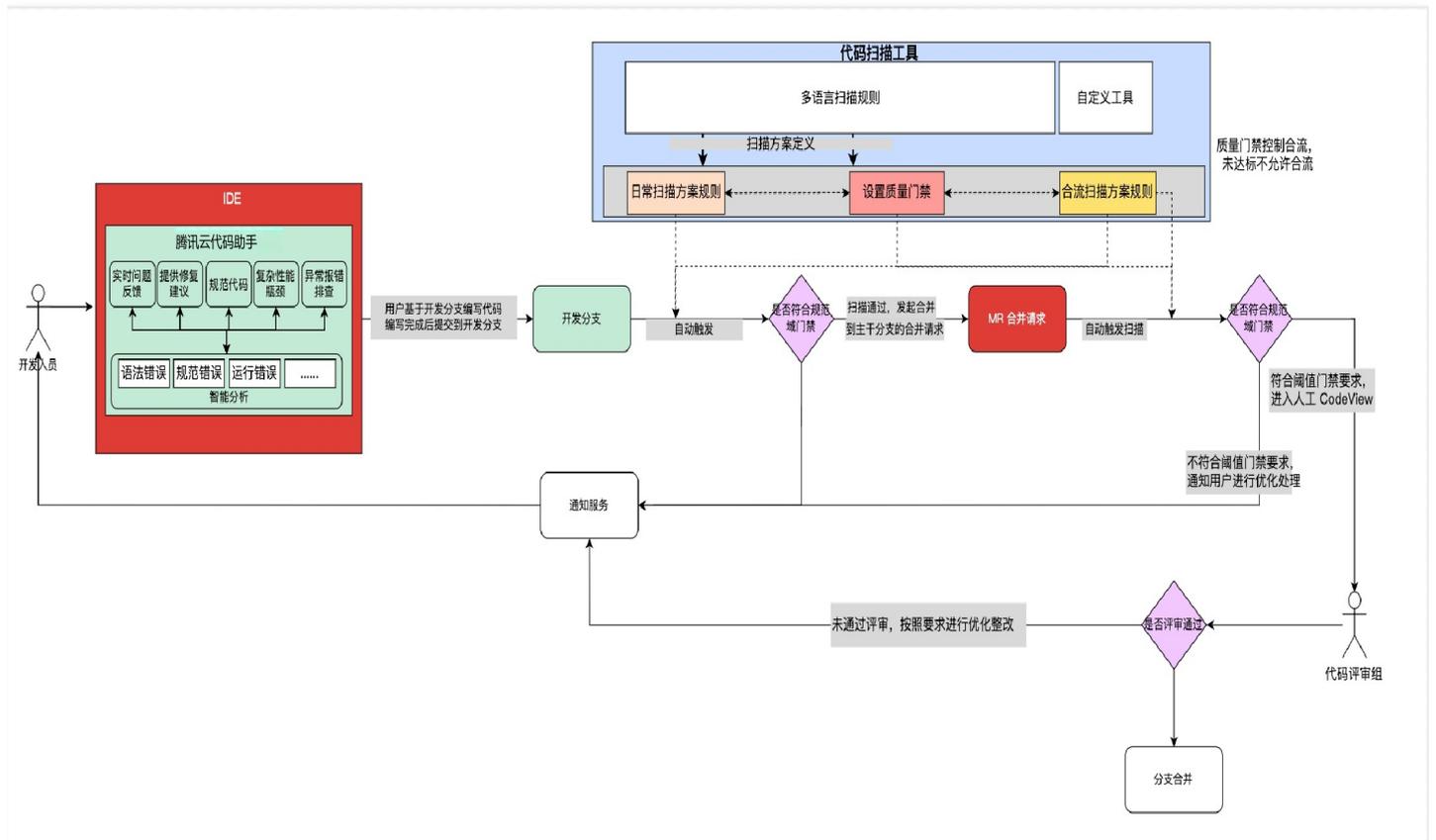
## 总结

腾讯云代码助手结合腾讯混元大模型的功能，已经在单元测试领域展现出了显著的价值，它不仅提高了开发效率，还帮助确保了软件的质量和稳定性。随着技术的不断进步，在软件工程3.0的时代下，AI 在单元测试中的应用将更加广泛和深入，为软件开发带来更多的便利和创新。

# 代码诊断应用实践

最近更新时间：2025-04-14 09:54:23

代码诊断是指通过检查和分析源代码，发现并定位其中的错误、缺陷或不规范之处。传统的代码诊断方法主要依赖于人工审查和简单的静态分析工具，结合流水线的自动化能力并且结合质量门禁建立不同的质量阈值关卡。而基于 AI 的代码诊断是在传统的能力基础上再次进行质量左移，通过代码助手在 IDE 的嵌入，结合先进的人工智能技术，赋予代码诊断过程更高的智能化水平，从而提升诊断的准确性和效率。基于 AI 的代码诊断不仅仅是自动化代码分析，更是通过智能算法对代码进行深度理解和分析，识别出潜在的问题并提供相应的解决方案。本文将为您介绍基于 AI 代码诊断的应用实践。

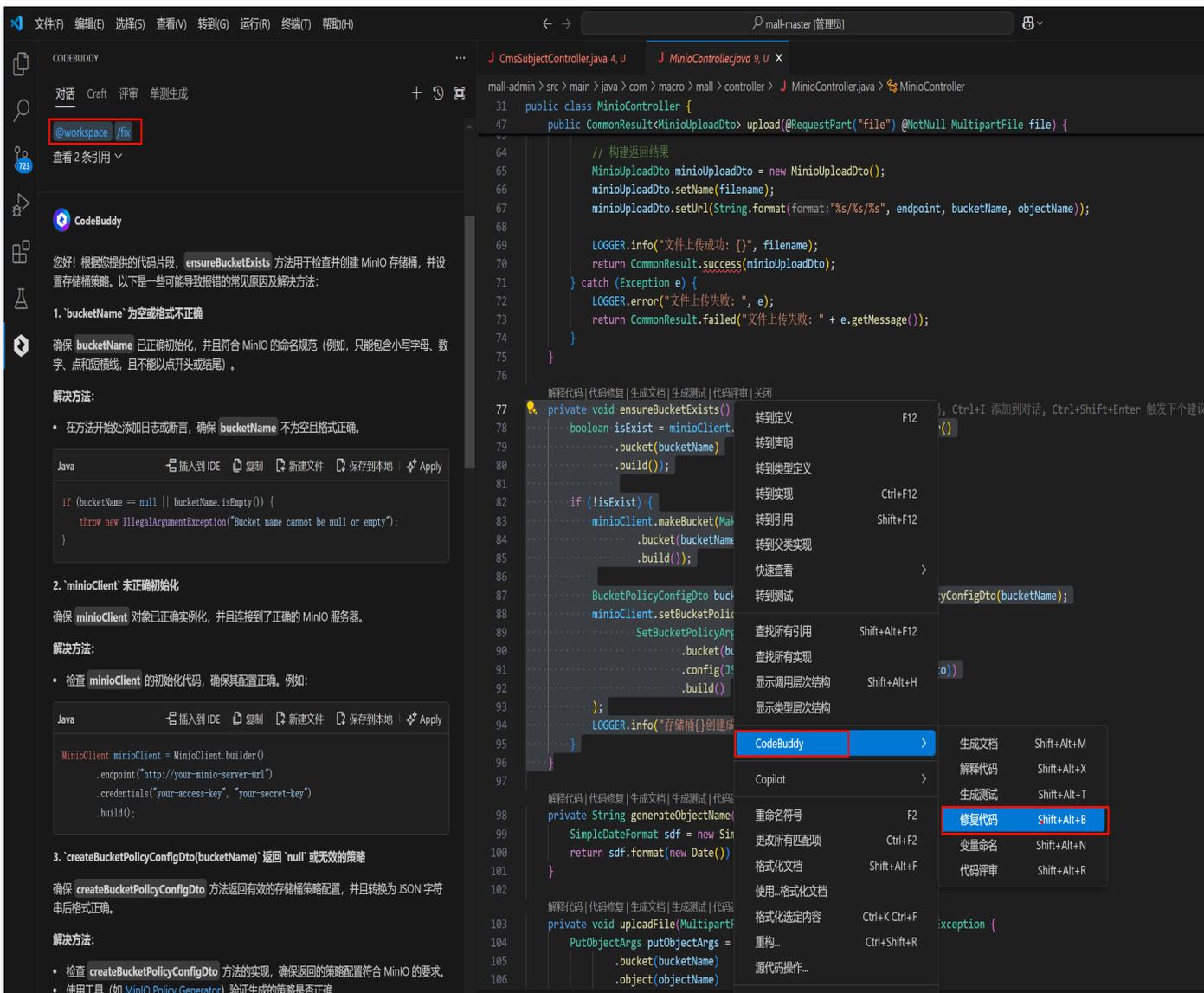


## AI 在代码诊断的应用

腾讯云代码助手通过感知能力针对光标所在位置的上下文进行分析，能够自动提示当前位置代码的状态，通过触发代码修复功能自动提供当前代码问题对应的解决方案，同时提供修复的实例代码进行参考，确保整体的完整性和修改后功能的稳定性。

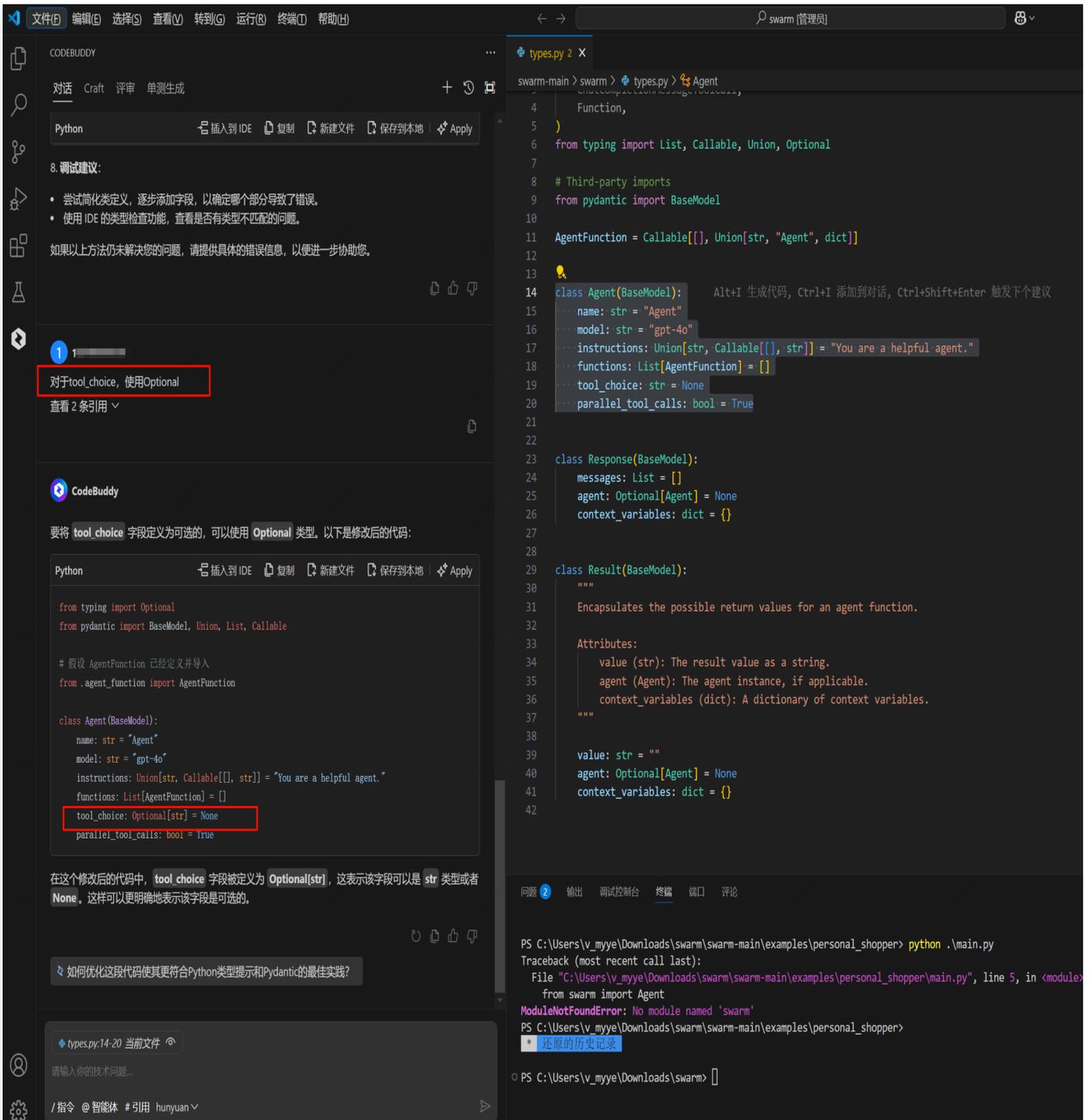
### 1. 代码诊断。

通过对话框中的 /fix 触发代码诊断，或者使用 IDE 编码区域中使用每个功能方法的快捷键代码修复来触发该方法的代码诊断。



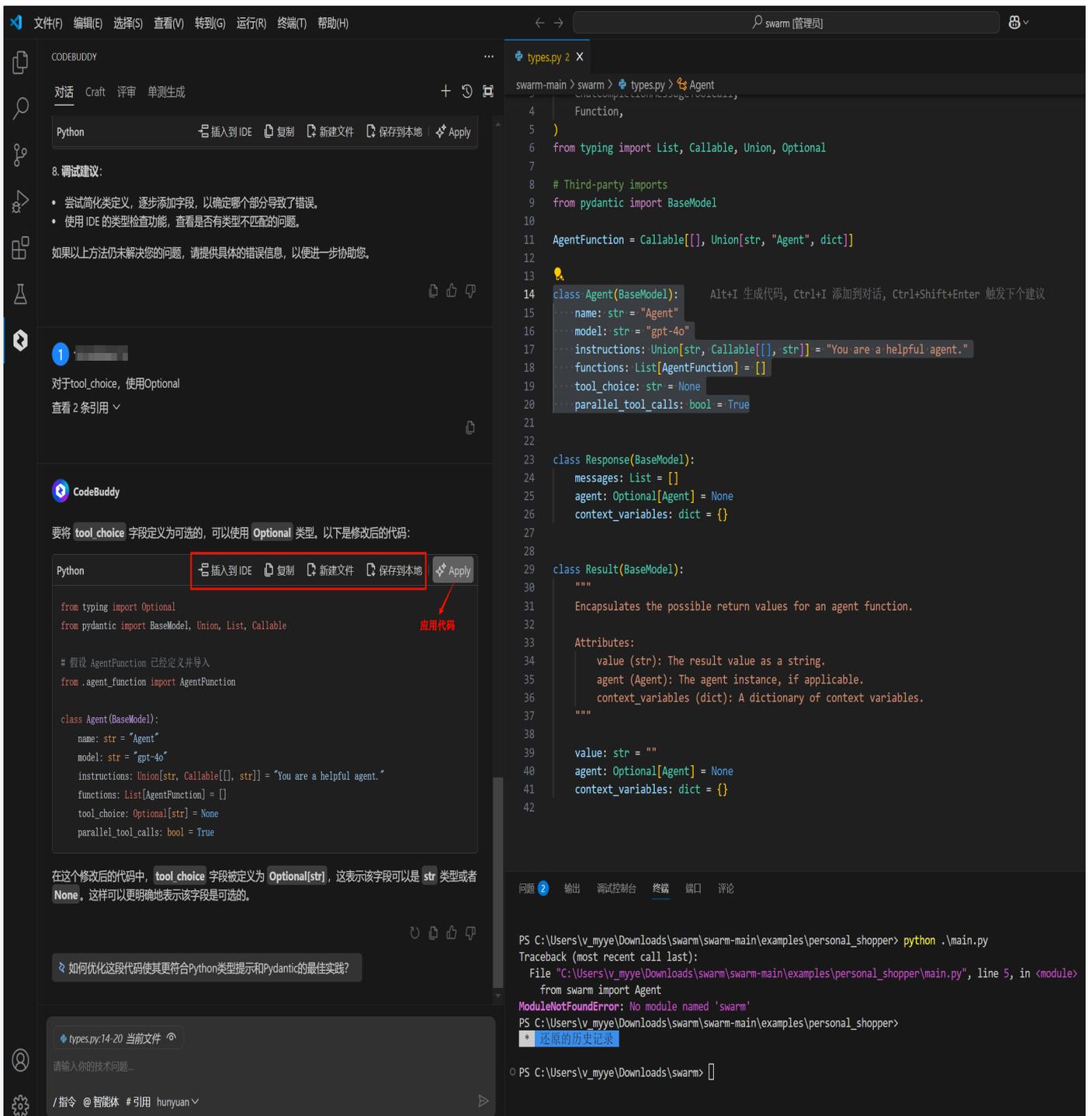
## 2. 多轮对话优化代码诊断。

通过多轮对话，告诉 AI 更多的信息，让代码诊断的内容更符合研发人员的预期。例如，可以指定业务边界条件、特殊的异常处理逻辑、数据处理方式等。代码助手的对话模型会识别用户意图结合上下文对话内容优化代码诊断结果。

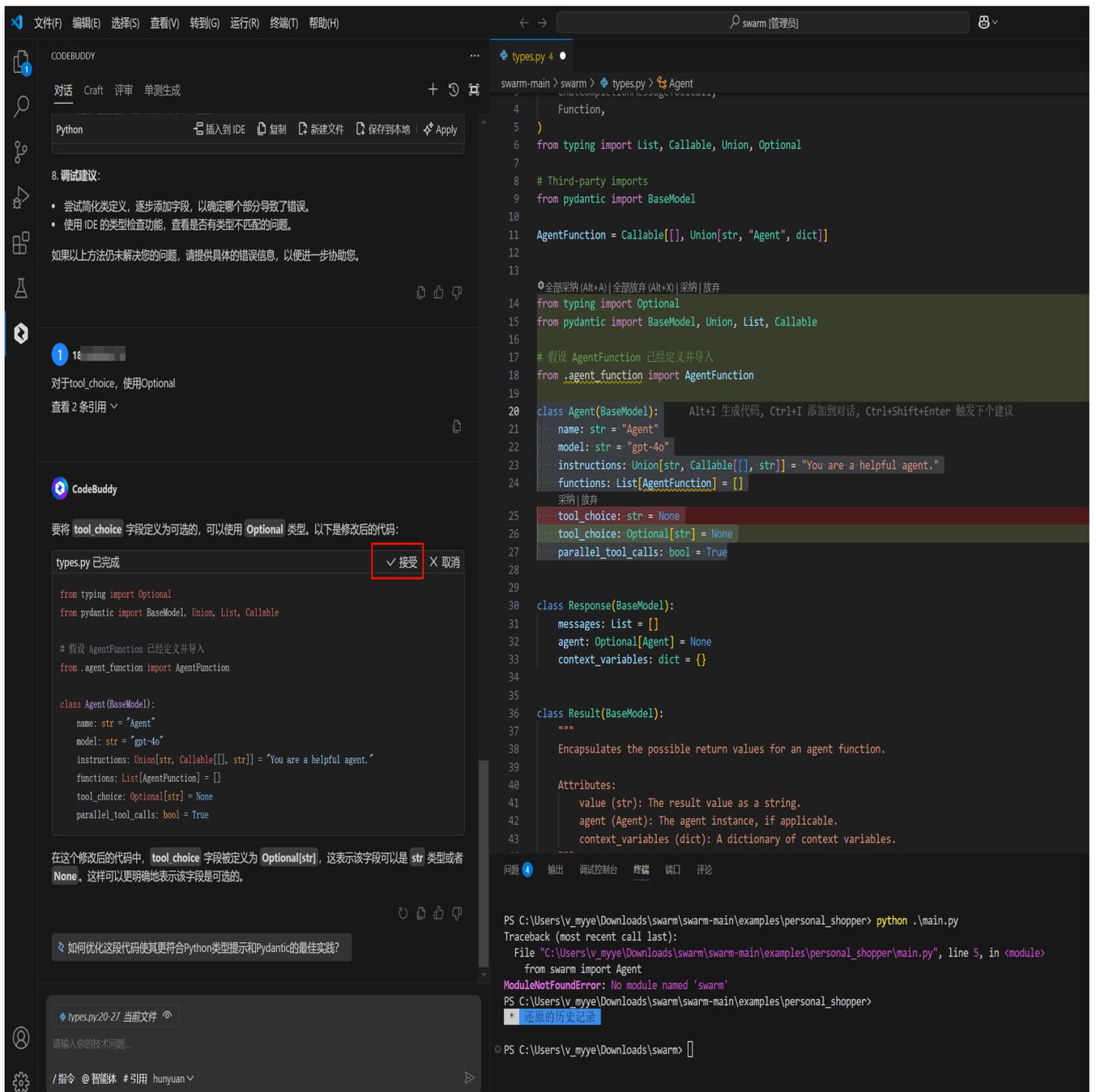


### 3. 一键应用代码诊断内容。

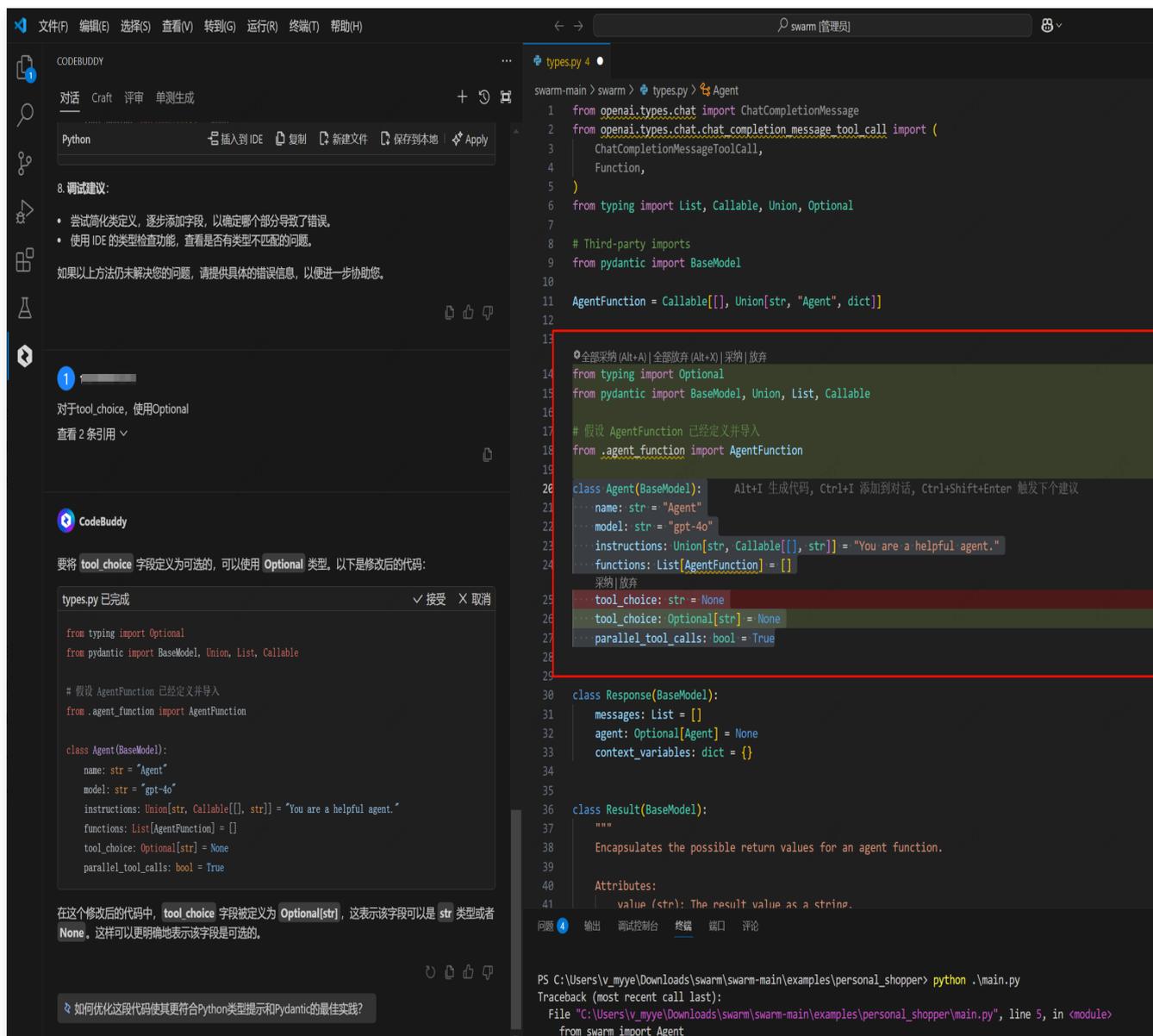
通过对话框结果中的快捷按钮，例如应用、插入到 IDE 等，开发者可以快速判断与接受生成的诊断建议代码。



### 3.1 可以选择应用代码，并且接受将对话诊断建议代码的结果，直接插入到对应的代码文件中。



3.2 可以在 IDE 编码区域看到应用这部分通过 diff 的能力进行高亮地区分诊断建议代码和当前文件代码的对比修改情况，让研发人员快速识别到改动并判断是否接受。

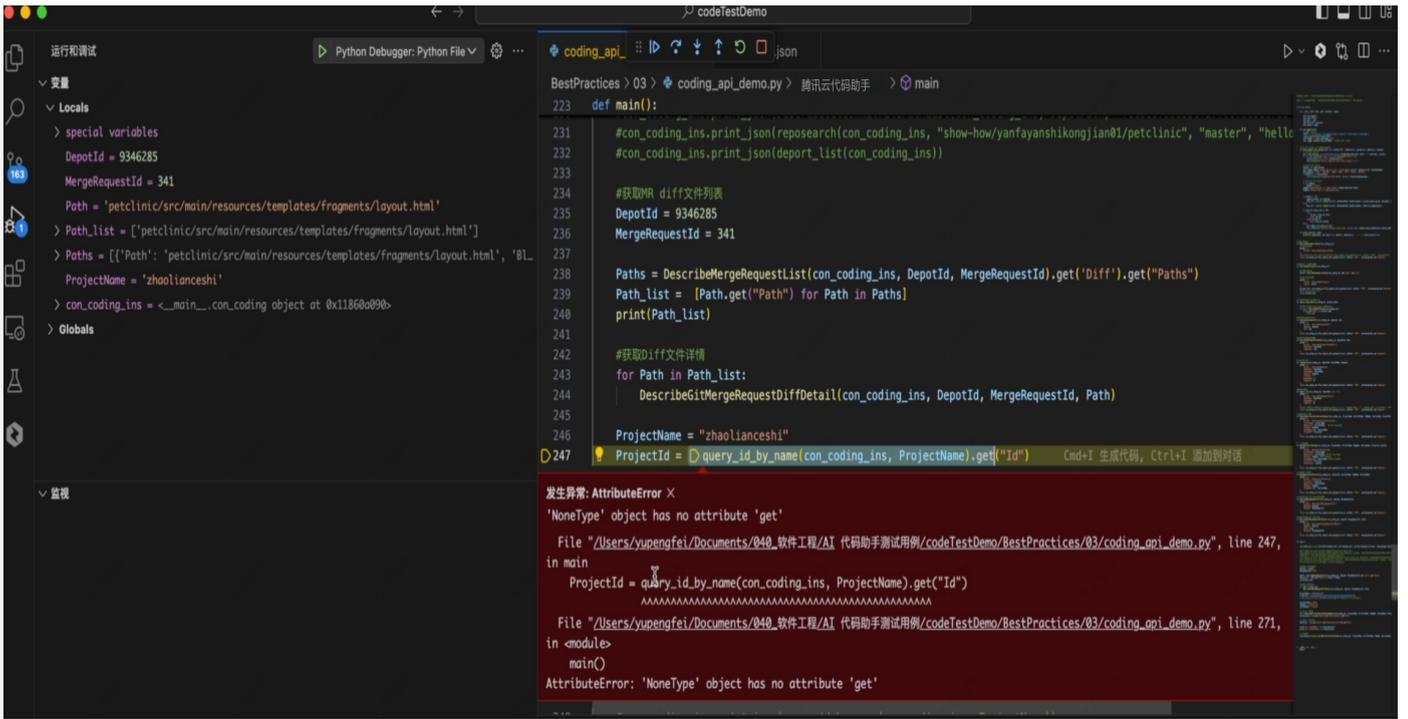


## 实际场景演示

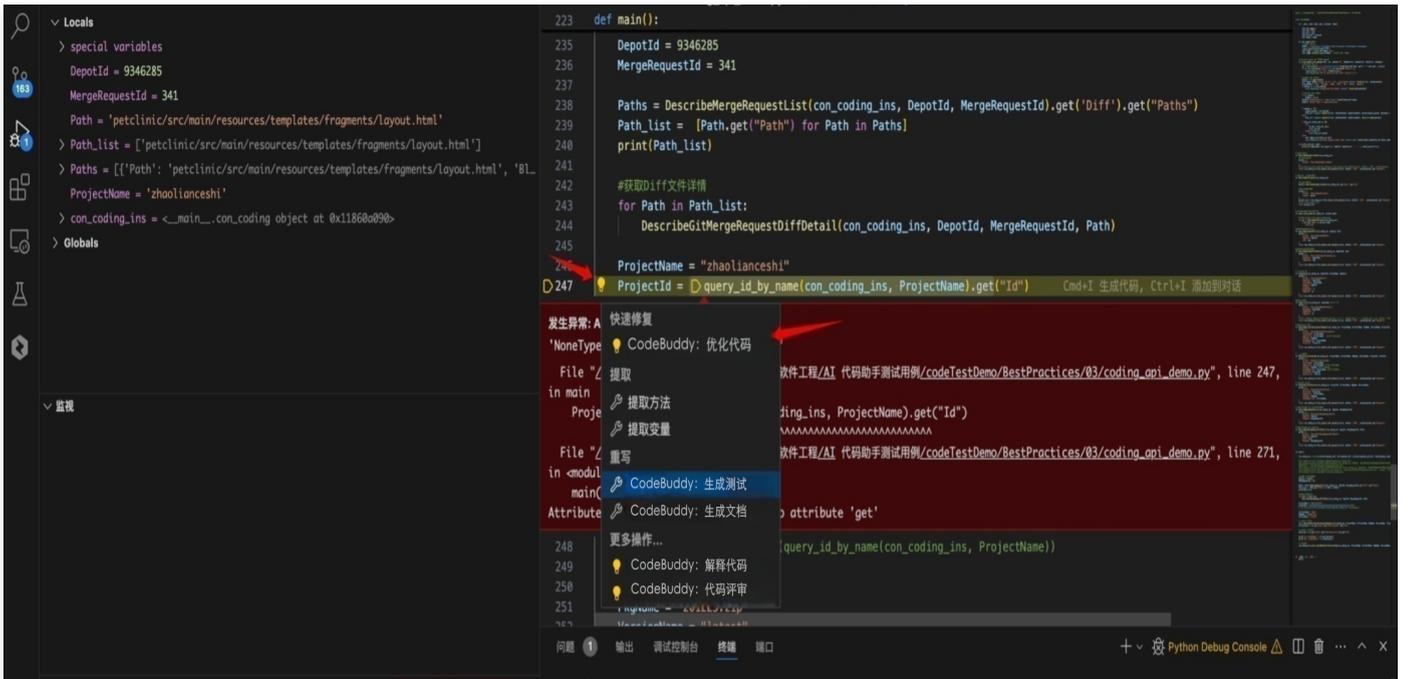
### Debug 调试时诊断修复

在开发过程中，开发者常常会通过 DeBug 方式进行程序代码的调试，调试过程中会遇到各种突发的小错误。通过对 DeBug 断点的快速定位异常信息，能够实时监控代码编辑器，即时提示语法错误、异常错误等问题，通过代码修复能力帮助开发者迅速修正，避免问题积累。

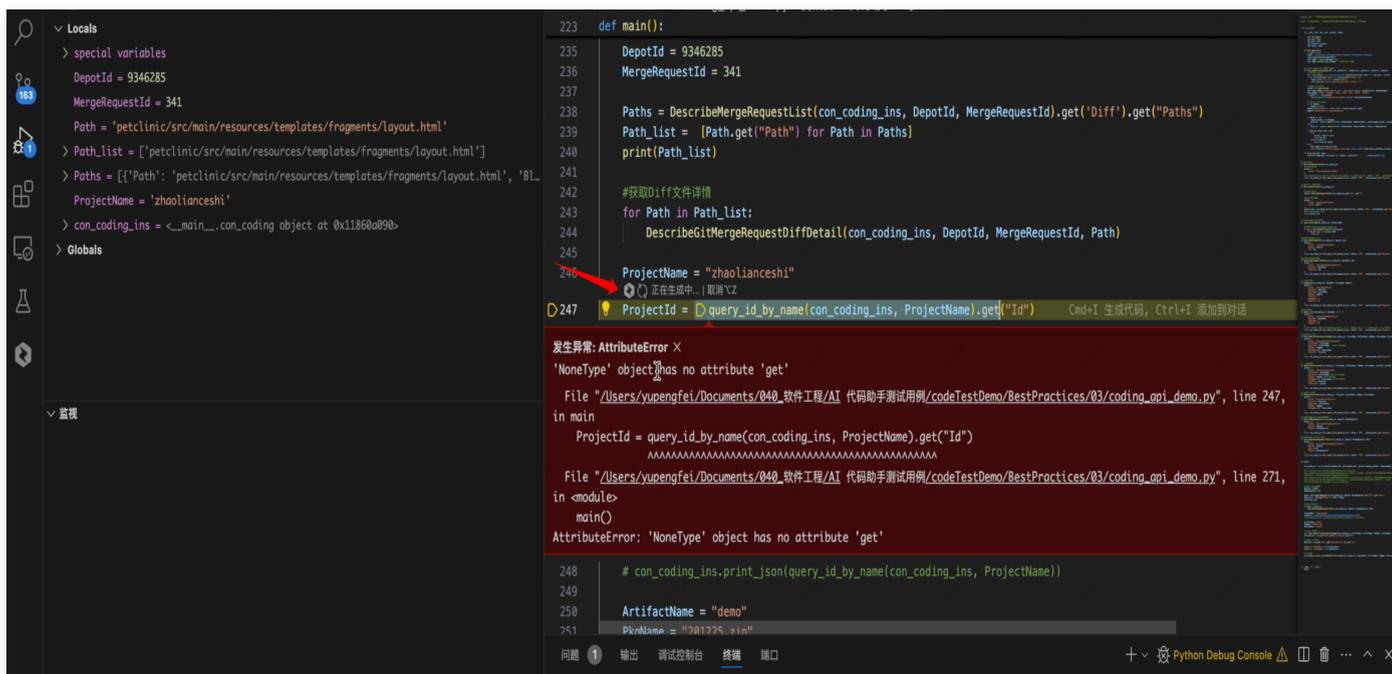
#### 1. 启动 Debug 时出现异常：



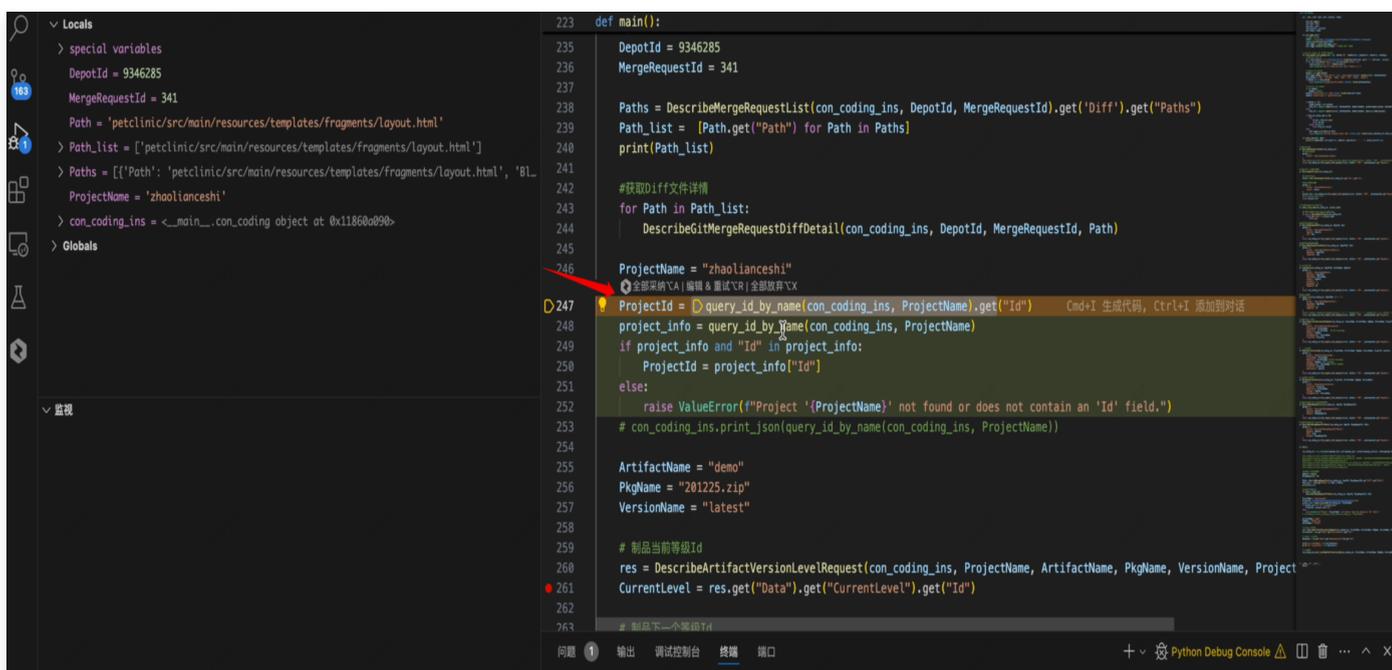
2. 通过 CodeBuddy: 优化代码功能获取解决方案:



3. 正在获取解决方案:



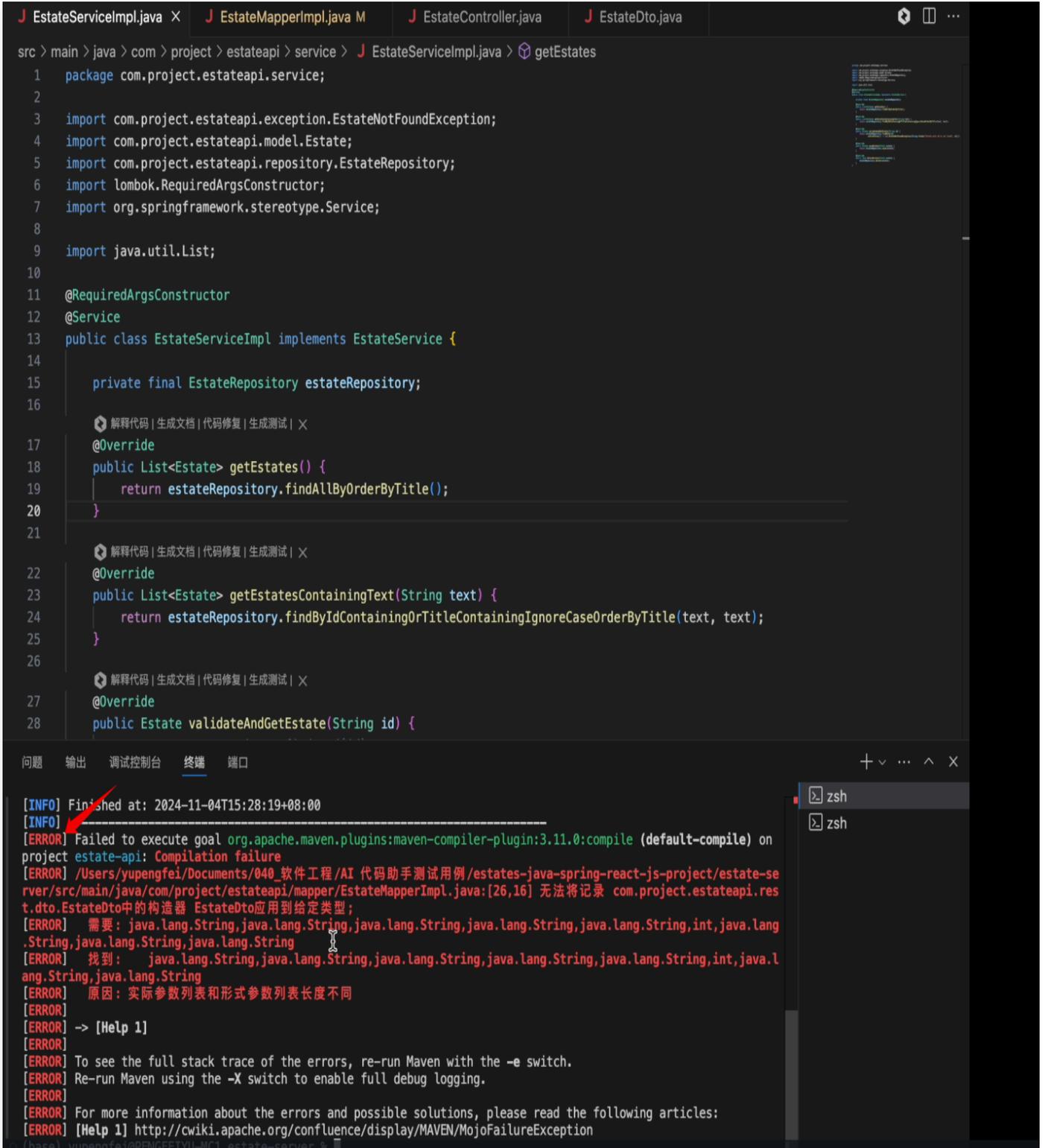
#### 4. 可以选择采纳对代码进行修复。



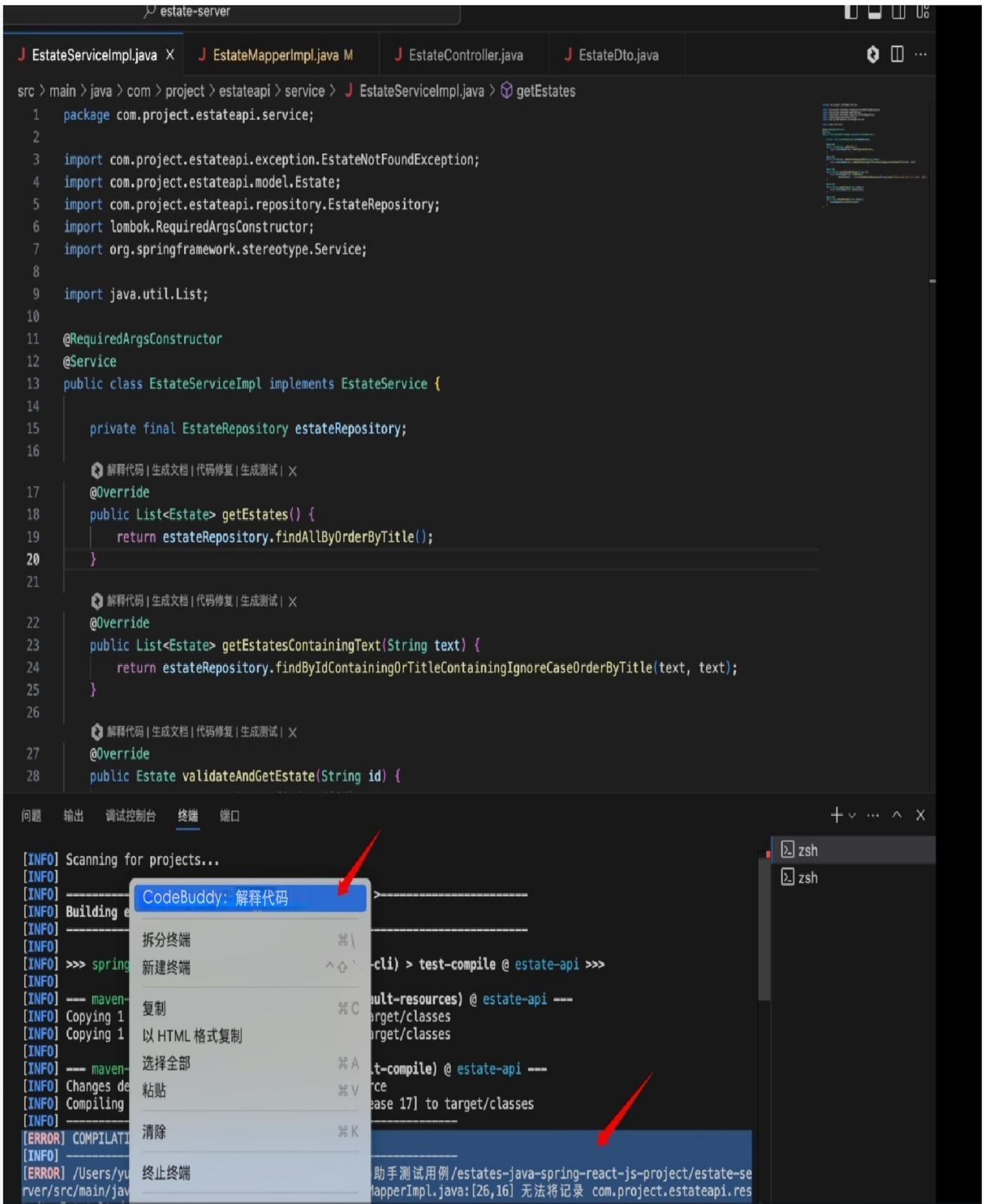
## 程序运行错误时定位及修复

软件运行过程中常常也会出现异常问题，包含数组越界、空指针、类型不匹配等报错情况，通过代码诊断结合多轮对话获取解决方案，帮助开发者快速的定位并解决问题。

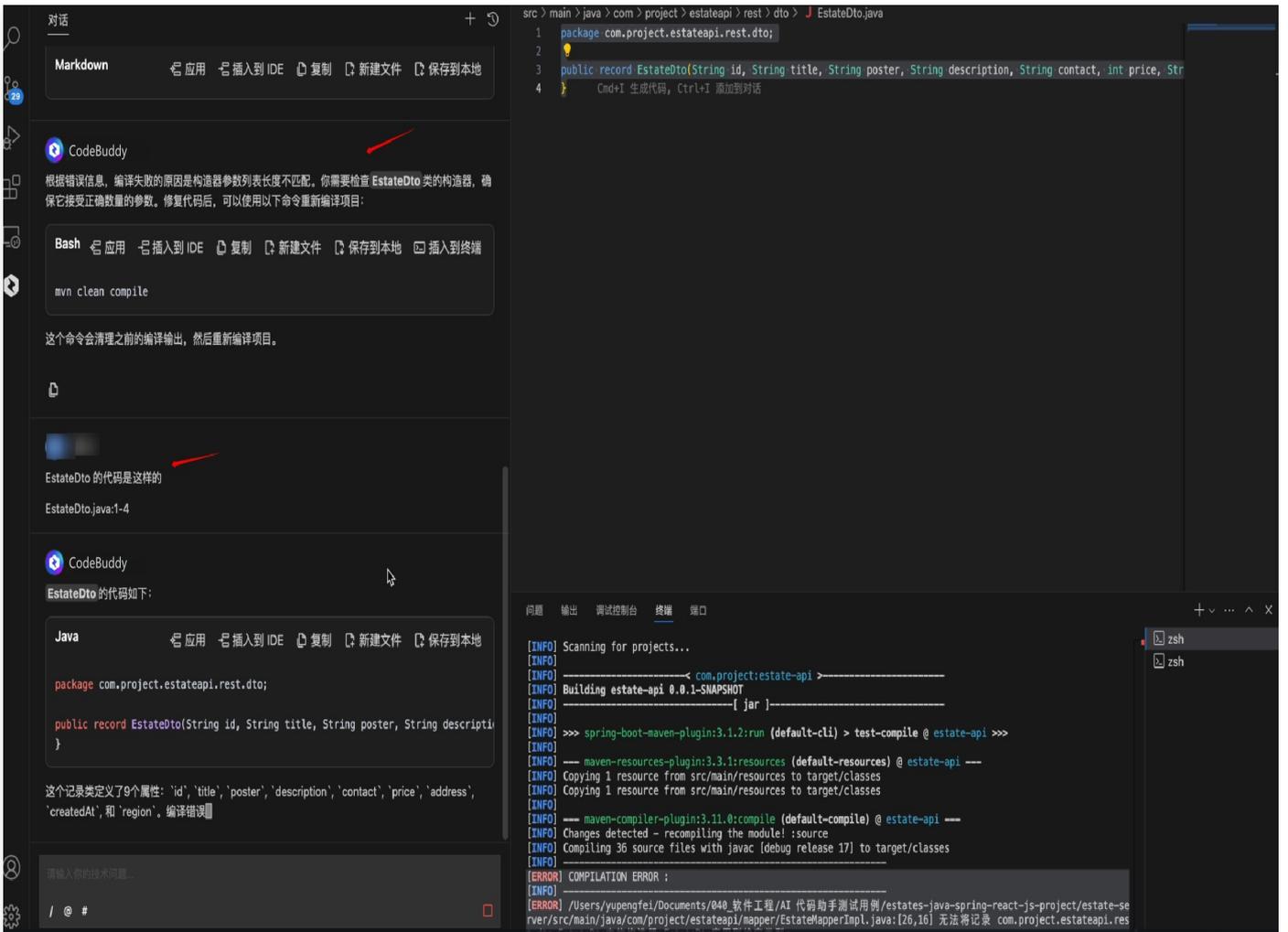
### 1. 运行错误报错：



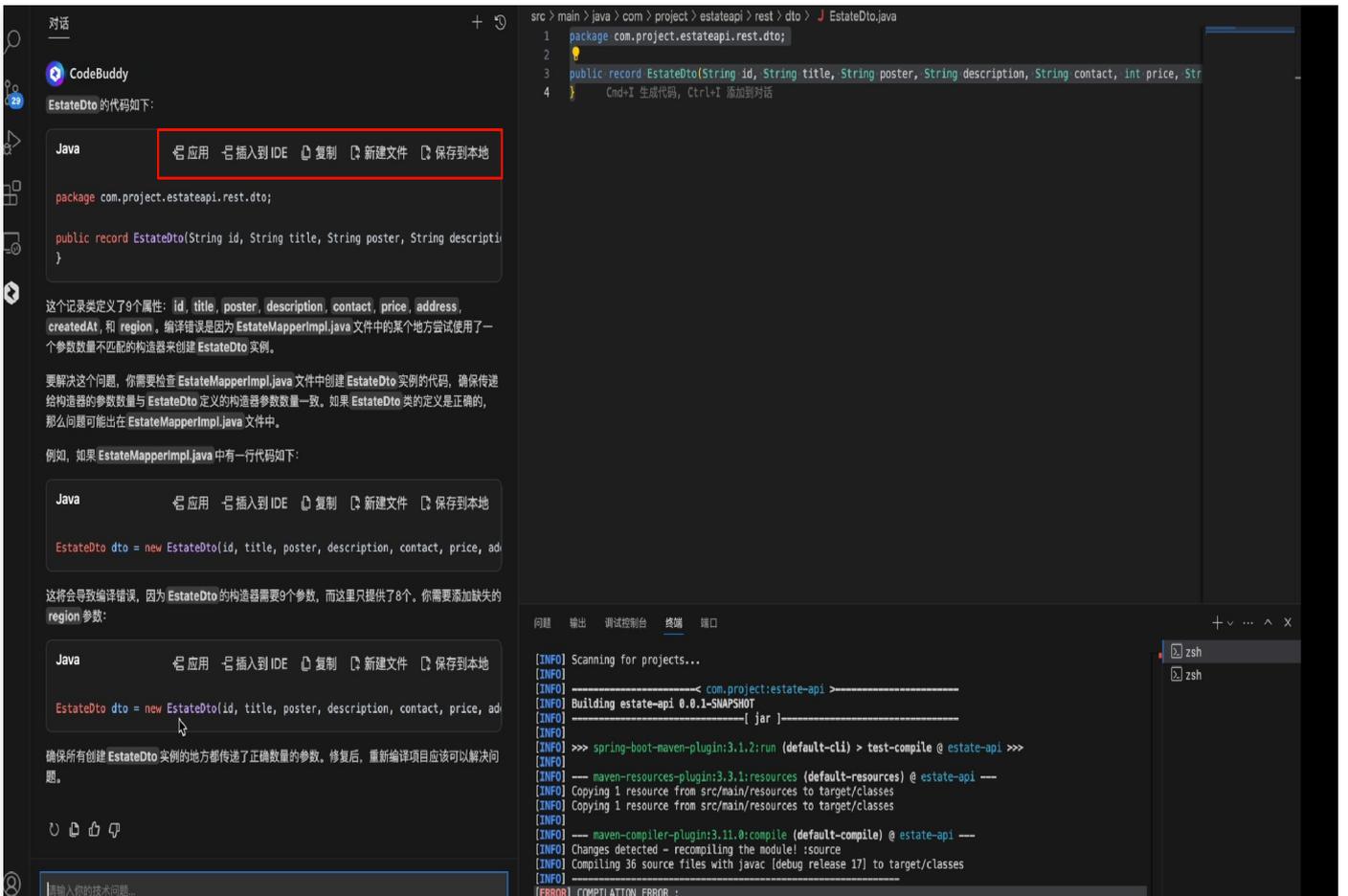
2. 选中异常报错信息通过 CodeBuddy: 解释代码功能进行问题分析:



### 3. 获取异常问题解释并结合多轮对话进一步分析问题：



4. 获取最终解决方案，通过代码助手对话框的应用能力把修复代码插入到方法中：



## 总结

腾讯云代码助手的代码诊断能力可以很好地为开发者提供有效的修复建议，通过对异常问题的快速定位及修复，能够缩短研发周期，提升效率，提高软件代码质量。相信在未来，随着人工智能技术的快速发展，基于 AI 的代码诊断能力会愈发强大。

# 企业管理实践

## 构建企业专属知识库

最近更新时间：2025-04-14 09:54:23

### 引言

在当前，大模型生成结果并不是100%准确的，它会产生幻觉，主要是因为以下两点：

- 训练大模型的数据是有时间限制的，如果您询问大模型超出语料数据日期的问题，则大模型无法准确回答。
- 大模型的数据来源于训练语料，如果是公司内部的私有数据，则大模型无法感知。

本文将介绍如何构建腾讯云代码助手的自定义 RAG（Retrieval-Augmented Generation，检索增强生成）知识库低成本更新代码助手知识内容、学习企业内部场景资料，帮助研发充分利用腾讯云代码助手的 RAG 知识库功能，提升研发效率与质量。

### 前提条件

- 适用版本：腾讯云代码助手企业旗舰版（限免）、腾讯云代码助手企业专享版、腾讯云代码助手私有化版本。
- 适用人员：腾讯云代码助手创建人、管理员。

### 自定义 RAG 知识库场景介绍

RAG 知识库的工作原理基于信息检索与生成增强机制。当研发人员提出问题或需求时，系统首先会在知识库中进行高效的信息检索，快速定位相关的知识片段。然后，利用这些检索到的信息作为上下文，结合强大的语言模型进行生成增强，从而为研发人员提供更精准、更全面的答案或代码示例。可以广泛应用于企业软件开发知识落地多个场景。

#### 1. 编程语言知识场景

- 知识库示例：Java 编程规范、Python 最佳实践、C++ 标准库文档、关联项目代码库或 API 文档。
- 需求解释：研发人员需要熟悉特定编程语言的语法、特性和最佳实践，以确保代码的质量和可维护性。
- 进一步拆分：
  - 常用库和框架。
  - 编程规范和最佳实践。
  - 错误处理和调试技巧。

#### 2. 软件架构知识场景

- 知识库示例：微服务架构设计、单体应用架构优化、事件驱动架构模式。
- 需求解释：合理的软件架构设计能够提高系统的可扩展性、可维护性和性能。
- 进一步拆分：
  - 架构风格和模式。

- 系统分解和模块化。
- 性能优化策略。
- 安全和容错设计。

### 3. 数据库知识场景

- 知识库示例：SQL 查询优化技巧、NoSQL 数据库使用指南、数据库事务管理。
- 需求解释：研发人员需要了解如何高效地存储、检索和管理数据，以满足业务需求。
- 进一步拆分：
  - 数据库设计和规范化。
  - 查询语言和优化技巧。
  - 数据库性能调优。
  - 数据安全和备份恢复。

### 4. 版本控制知识场景

- 知识库示例：Git 使用要求、版本控制最佳实践、分支策略。
- 需求解释：版本控制系统是软件开发中不可或缺的工具，有助于团队协作和管理代码变更。
- 进一步拆分：
  - 版本控制工具内部使用要求。
  - 分支和合并策略。
  - 代码审查和冲突解决。
  - 持续集成和持续部署（CI/CD）操作规范。

### 5. 测试知识场景

- 知识库示例：单元测试框架、集成测试方法、性能测试要求。
- 需求解释：全面的测试能够确保软件的质量和稳定性，及时发现并修复问题。
- 进一步拆分：
  - 单元测试和集成测试。
  - 性能测试和负载测试。
  - 自动化测试历史脚本。

### 6. 安全知识场景

- 知识库示例：OWASP-Web 安全指南、加密算法参考项目、Java 语言安全编码实践。
- 需求解释：软件安全是至关重要的，研发人员需要了解如何防止常见的安全漏洞和攻击手段。
- 进一步拆分：
  - 常见安全漏洞和攻击手段。

- 安全编码规范和最佳实践。
- 加密和解密技术要求。

## 构建高质量自定义知识库

### 1. 明确目标与范围

在构建软件研发知识库之前，首先要明确其目标和范围。确定知识库旨在解决哪些具体问题，服务于哪些用户群体，以及涵盖哪些研发领域的知识和经验。可以参考下方四个方面评估文本：

1. 相关性：文本内容与问题主题的匹配程度。
2. 帮助性：文本中信息对回答问题的直接贡献。
3. 准确性：文本信息的正确性和可靠性，不包含错误或逻辑问题。
4. 简洁性：文本是否简明扼要，无多余或无关信息。无关信息越多，简洁性越低。

### 2. 收集与整理知识

1. 内部知识来源（推荐上传项目相关代码库压缩包）：
  - 团队内部的技术文档、项目总结报告、团队规范要求。
  - 开发过程中的场景最佳实践文档、复盘文档。
  - 团队成员的专业知识和技能分享文档。
2. 外部知识来源（推荐使用外部官方发布API文档或仓库）：
  - 学术论文、行业报告、专业书籍。
  - 开源项目代码、技术论坛讨论。
  - 行业会议、研讨会的技术分享。

### 3. 语料文档准备

- **语料文档的基础性作用：**语料文档是 RAG 知识库的核心组成部分，其质量直接决定了知识库的有效性和实用性。丰富、准确且格式规范的语料文档能够确保知识库在信息检索和生成增强过程中提供高质量的结果。
- **格式要求与规范：**
  - **文档格式：**推荐使用 Markdown 格式，它简洁易读，支持标题、段落、列表、代码块等多种元素的清晰表达，方便知识库系统对内容进行解析和处理。
  - **编码规范：**统一采用 UTF - 8 编码，以确保文档在不同系统和环境下的兼容性，避免出现乱码等问题，保证知识内容的准确传递。
    - **内容结构：**
      - **标题：**使用清晰明确的标题来划分文档的不同章节或主题，便于系统和研发人员快速定位和理解内容。标题应遵循一定的层次结构，例如使用 #、##、### 等符号来区分不同级别的标题。命名尽量控制在10字以内，涵义简洁明了，避免标题中出现无意义的数字、符号或缩写。
      - **段落：**段落内容应简洁明了，逻辑连贯，围绕一个核心主题展开阐述。避免冗长复杂的句子和段落，以提高信息的可读性和检索效率。

- **列表**: 对于列举项、步骤等内容, 使用无序列表 ( - ) 或有序列表 ( 1. 2. 3. ) 进行展示, 使信息更加清晰有序, 易于理解和提取。
- **数据清洗要求**:
  - **去除冗余信息**: 仔细检查文档内容, 删除与主题无关的广告、重复的表述、过时的信息等冗余内容, 确保知识库的简洁性和高效性。
  - **减少干扰信息**: 尽可能减少文档中包含表情的特殊字符、批注、页眉、页脚或水印信息, 以免造成解析问题。
  - **统一术语**: 在涉及专业术语、技术名词时, 应确保整个文档中的表述一致。例如, 对于 “Function ( 函数 ) ” 不应在其他地方出现 “方法 ( Method ) ” 或 “类 ( Class ) ” 等不同写法表达同一指向对象, 避免因术语不统一导致检索结果不准确。
- **构建问答对**: 因为模型类产品最终的交互是问答的形式, 所以在构建知识库时, 如果能基于用户可能的提问方式, 构建问答对, 作为知识库原始数据, 一般能达到更好的效果, 但构建问答对的过程需要耗费较多人力以及运营成本, 低成本做法用户可考虑对文档中的重要概念或词汇打上标签, 帮助 AI 识别关键词从而更好地理解文档主要内容。

## 4. 自定义知识库数据范围

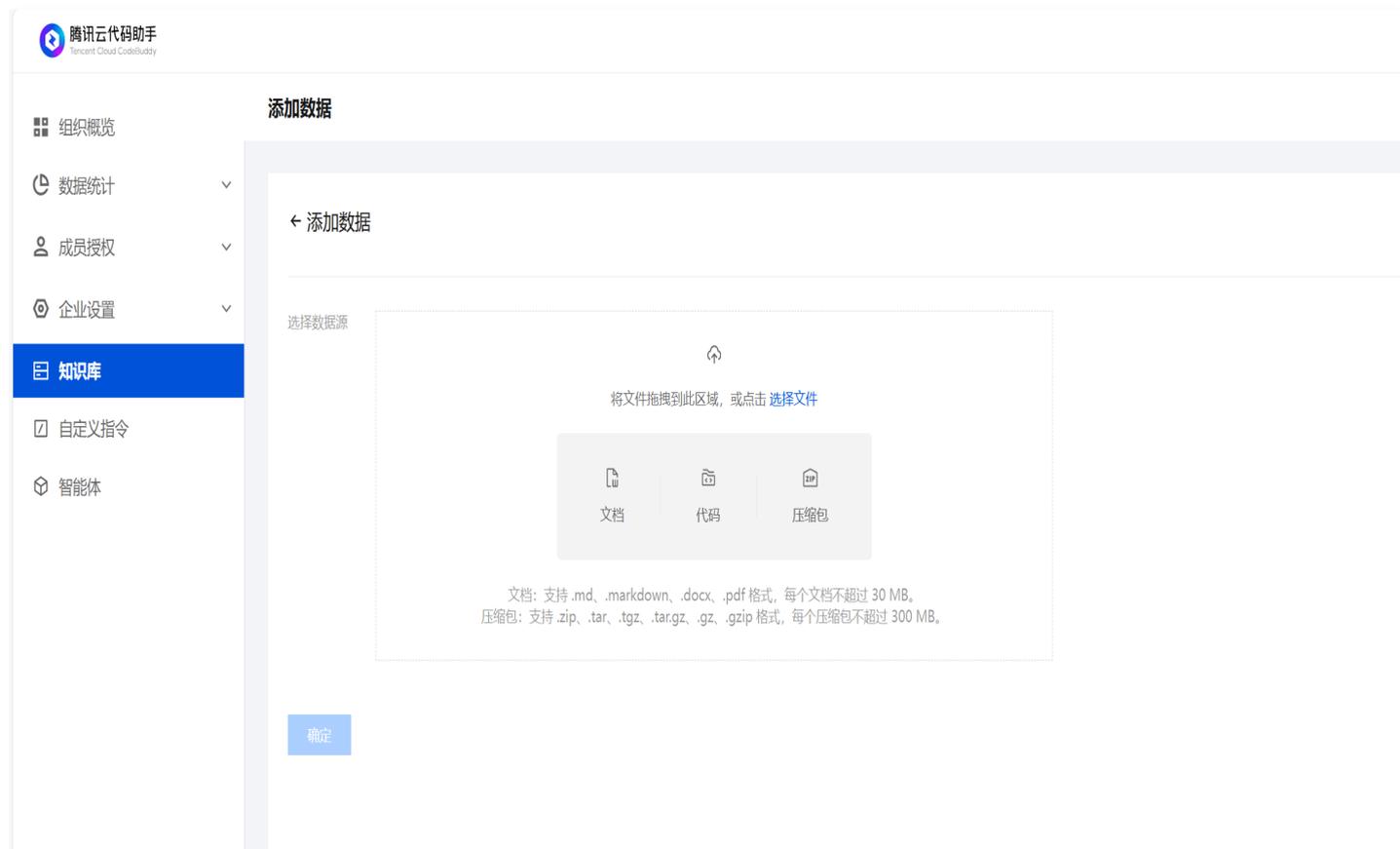
代码助手当前模型聚焦文本文档类型的数据, 主要形式为自上而下、自左向右编辑的文字。文本格式需要遵循常见中文写作规范, 提供 Embedding 向量模型选择 ( 已支持 bge-m3 ); 尽可能地使用文字表达信息, 如果语料文档中图像的文字比较少且包含重要信息, 建议将信息转录成文字的形式。确保所有核心图都有图示说明 ( 即图解或说明 ), 图示说明应清晰地解释图中的主题。

- **文档格式**: 支持 .md、.markdown、.docx、.pdf 格式, 每个文档不超过30MB。推荐上传 .md 格式文档。
- **代码文件格式**:
  - 支持主流编程语言源文件格式, 例如:
    - Python: .py 文件, 包含 Python 源代码。
    - Java: .java 文件, 包含 Java 源代码。
    - C/C++: .c 或 .cpp 文件, 包含 C 或 C++ 源代码
    - JavaScript: .js 文件, 包含 JavaScript 源代码。
    - HTML/CSS/JavaScript: .html、.css、.js 文件, 用于网页开发。
    - Visual Basic: .vbp、.frm、.bas 等文件, 用于 VB 编程开发。
  - 支持配置文件格式, 例如:
    - INI: 用于存储程序的配置信息, 如数据库连接配置。
    - JSON: 用于存储结构化配置信息, 常用于 Web 开发和 API 交互。
    - YAML: 用于存储结构化配置信息, 常用于 Web 开发和 @DevOps 领域。
  - 支持数据文件格式。例如:
    - CSV: 用于存储表格数据, 常用于数据分析和机器学习领域。
    - JSON: 用于存储结构化数据, 常用于 Web 开发和 API 交互。

- XML：用于存储和传输数据，常用于 Web 服务和配置文件。
- Jupyter Notebook：.ipynb 文件，包含代码、文本、图表等，用于交互式编程和教学。

腾讯云代码助手支持将文档与代码文件进行混合压缩后上传，支持常见压缩包格式

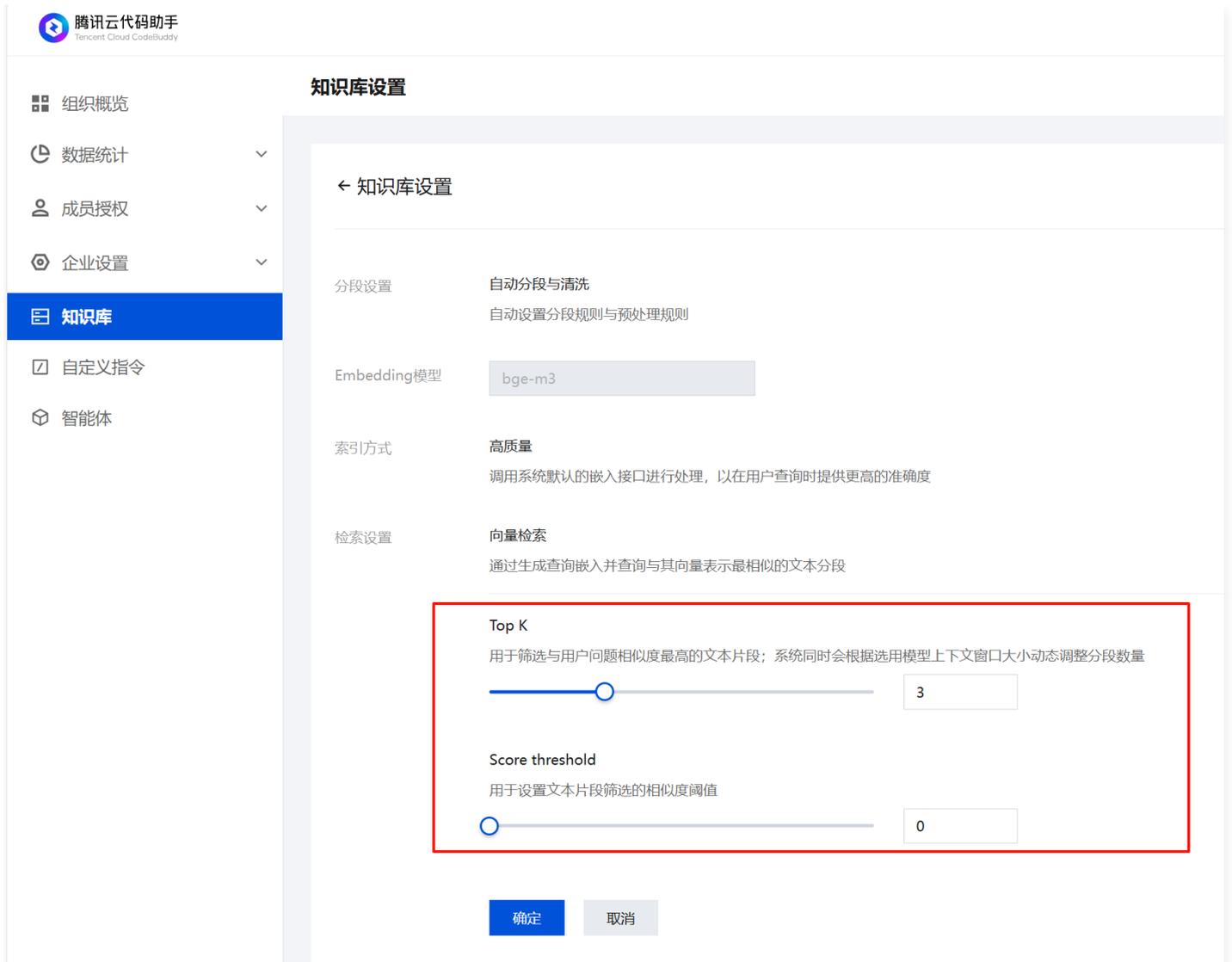
（.zip、.tar、.tgz、.tar.gz、.gz、.gzip 格式，每个压缩包不超过300MB，压缩包内文件数量不限）。



如果当前知识库内容容量较大可以进行主题拆分后上传为多个主题知识库，在进行知识库问答时分布调用不同主题知识库同时进行问答。

## 5. 自定义知识库配置

在腾讯云代码助手后台，进入知识库 > 自定义知识库 > 具体自定义知识库 > 设置路径下可以对自定义知识库的检索进行配置，配置页面及配置项解释如下：



## 1. Top K 值:

配置介绍：用于筛选与用户问题相似度最高的文本片段，建议设置区间为3到5，较小的值返回更少但更精确的结果，较大的值返回更多的候选结果。

## 2. Score Threshold 值:

配置介绍：设置结果的相关性分数阈值，建议设置区间为0.5到0.8，较低的值返回更多结果，较高的值只返回高度相关的结果。Threshold 的区间模型给的是-1到1，0是一个中间态。默认 Threshold 是0，用户可以根据自己实际业务适当设置高一些。

更多腾讯云代码助手自定义 RAG 知识库使用请参见 [RAG 知识库](#)。