WorkBuddy Enterprise

CodeBuddy



腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2026 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

CodeBuddy
- 产品简介
  - 产品概述
  - 产品优势
- 安装插件
  - JetBrains IDE 使用插件
  - Visual Studio Code 使用插件
- 快速开始
  - 产品安装
  - 登录及退出
  - 功能入门
- 操作指南
  - 代码补全
  - 技术对话
  - 单元测试
  - 代码评审
  - 代码解释
  - 内联对话
  - 工程理解
  - RAG 知识库
  - MCP
  - Craft 智能体
    - 终端 shell integration 问题指引
    - Craft 的使用
    - Craft 撤回检查点
  - 自定义规则
  - 防火墙配置
  - 网络代理配置
  - 个人版账号管理
  - 小程序开发 IDE
    - 全新内核微信 IDE + CodeBuddy 4.3 版本
    - 技术对话功能
    - 高级设置
    - 对话模型接入和切换
  - Visual Studio

版权所有：腾讯云计算（北京）有限责任公司

接入指南
使用指南
Xcode
接入指南
使用指南
高级功能
model.json 配置指南
Plan 模式
subagents 使用指南
Skills
Hooks 功能使用文档
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
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
帮助与反馈

版权所有：腾讯云计算（北京）有限责任公司

CodeBuddy 产品简介

产品概述

最近更新时间：2026-07-20 21:46:06

CodeBuddy 同时支持插件、IDE 和 CLI 三种编程形态，覆盖从专业开发者到零基础用户的全场景需求。CodeBuddy 插件版是以插件形式安装到编辑器（VS Code 或者 JetBrains 系列 等 IDE）中辅助编程，传统开发者主导，AI 打辅助的编码模式；CodeBuddy IDE 主打 “对话即编程”，通过将 AI 深度融入从需求规划、产品设计、代码研发、产品部署的全流程，实现从产品构思到产品发布的一站式高效交付平台；而 CodeBuddy Code 是一款面向专业工程师的 AI CLI 工具，能无缝融入现有开发流程，支持在命令行中用自然语言驱动开发全流程，实现高度自动化。

形态选择

根据您的开发需求，可参考下方的选择建议：

赋能多样开发场景	CodeBuddy 插件	CodeBuddy IDE	CodeBuddy Code
需求用户	日常编码开发者 / 特定 IDE 使用者	产品 / 设计师 / 全栈开发 / 编程初学者	DevOps / 运维 / SRE / 资深开发者
核心优势	• 即插即用，零成本学习 • 无缝融入现有工作流 • 轻量级资源占用	• 产设研一体工作台 • 腾讯生态集成：CloudBase / EdgeOne Pages / CloudStudio • 可视化调试	• 系统级操作：Shell / 文件 / 网络 • 无头环境支持 • 任务编排能力：需求规划 / Sub Agent
应用场景	**日常效率提升** • 单 / 多文件日常编码 • 语法补全 • 快速注释生成	**0-1 沉浸式编码辅助** • 快速原型 MCP 开发 • 设计稿转代码 • 复杂业务逻辑可视化开发，交互调试。	**复杂业务的自动化** • 无头环境的编码自动化 • CI / CD 部署与运维闭环 • 复杂项目批量重构
使用指引	可直接在各大 IDE 插件市场搜索 **腾讯云代码助手**	• 国际版 • 国内版	**终端执行：npm install -g @tencent-ai/codebuddy-code**（依登录方式选择国际 / 国内版）

支持的常见语言和框架

理论上，CodeBuddy 不限制任何类型的代码语言，甚至对于中文内容的输入，都能进行自动补全。

分类	支持的语言
前端	HTML、JavaScript、CSS、Vue、React、Bootstrap 等。
后端	JavaScript (Node.js)、Python、Java、C#、Ruby、PHP 等。
客户端	Java、Kotlin、Swift、Objective-C、C# 等。
其他	C++、Go、Rust、TypeScript、Shell、PowerShell、R、MATLAB、Perl、Lua 等。

核心功能

CodeBuddy 插件

智能补全代码信息

腾讯云代码助手可以对 IDE 中的业务代码进行全面解析，在随后的代码编程中进行自动补全。上下文的代码质量越高、内容越丰富，补全的代码信息越准确。

精准修复错误代码

腾讯云代码助手可以快速检查代码中的语法错误和逻辑错误，帮助开发者规范代码格式。

清晰解释既有代码

当开发者需要处理别人的代码，或者在既有代码中进行延展开发时，可以通过腾讯云代码助手快速理解初始代码，降低理解成本。

按需生成单元测试

腾讯云代码助手会根据函数、方法、内容逻辑生成相关的测试代码，快速实现单元测试。

智能审查本地代码

腾讯云代码助手推出智能评审辅助的功能，开发者能够在项目开发过程中及时发现并解决本地代码变更可能引入的问题，从而提高代码质量，加速开发流程。

清晰理解工程代码

腾讯云代码助手提供 @workspace 和 #Codebase 能力，开发者可以对工程进行提问，能够帮助开发者快速获取与整个代码仓库相关的问题答案。不管是对代码结构、函数和类关系、项目工程依赖，还是复杂的代码逻辑和业务流程有疑问，都能够提供精确且与上下文相关的解答。

版权所有：腾讯云计算（北京）有限责任公司

人工智能技术对话

腾讯云代码助手将对话问答功能与 IDE 进行了集成打通；开发者可以随时随地，以各种方式向腾讯云代码助手进行技术对话咨询，所获取的代码内容可以一键插入编辑器当中。

支持自定义指令

通过设置自定义指令，开发者可以快速调用常用的代码片段或函数，避免重复输入和查找；同时，自定义指令功能允许用户调整 AI 的响应方式，使其更符合个人的编程风格和习惯；最后，可以帮助团队成员保持一致的编码风格和规范，有助于提高代码的可读性和可维护性。

支持创建 RAG 知识库

腾讯云代码助手支持企业构建专属的知识库，在提供研发问答和智能编码服务时，更加精准、全面，有效增强了模型输出的效果，助力开发者高效解决问题，提升编码效率。

支持对话大模型接入和切换

腾讯云代码助手除了支持腾讯自研的混元大模型之外，也支持其它例如 DeepSeek 大模型的接入，有效提升了对话的理解，帮助开发者实现更加强大、高效、安全的 AI 编程体验。

CodeBuddy IDE

需求文档 PRD 生成

只需用自然语言描述产品构想， AI 即可辅助进行深度需求分析，自动生成结构化的需求文档（PRD）。

产品原型 / 交互 / 设计生成

- **自然语言生成**：可以将抽象的文字描述转化为可交互的产品原型。
- **图像化生成**： 支持上传手绘草图或参考图片，智能识别并生成高保真设计稿。
- **组件化生成**： 基于组件库（目前已内置 TDesign、MUI、Shadcn，持续更新中），快速拼装生成符合规范的设计稿。
- **实时视觉优化**：支持通过自然语言指令（如“让风格更有科技感”、“按钮调成蓝色”）对原型 / 设计稿进行实时视觉迭代和优化。

设计稿转代码

内置 Figma，支持直接将设计稿转化为前后端可维护源码，打通从设计到开发的“最后一公里”。

内置 BaaS 服务

内置 Supabase / Tencent CloudBase 后端服务，无需手动配置，自动处理数据库、用户认证、腾讯云生态等需求。

支持一键部署

内置 Tencent CloudStudio / EdgeOne Pages 部署平台，一键部署至沙箱环境 / 生产环境，并生成可分享链接，方便团队快速验证反馈。

全能 AI 编程助手

为开发者提供覆盖全工作流的编码支持，全新升级：

- **软件开发智能体**: 自主完成多文件代码生成和改写，即刻落地可执行应用。
- **实时代码续写**: 提供多行编辑的智能代码补全，并基于上下文理解及编辑行为，实时预测下一个改动点。
- **行间 / 终端内联对话**：无需离开编辑器或终端，即可直接与 AI 对话获取实时帮助，解决代码、命令行问题或执行脚本。

模型信息

CodeBuddy，使用了腾讯自研的混元助手大模型，具体备案信息如下：

模型名称	腾讯混元助手大模型
备案单位	深圳市腾讯计算机系统有限公司
备案号	Guangdong-TencentHunyuan-20230901
备案公告	国家互联网信息办公室关于发布生成式人工智能服务已备案信息的公告

版权所有：腾讯云计算（北京）有限责任公司

产品优势

最近更新时间：2026-07-20 19:49:32

对比 ChatGPT

内嵌 IDE，开发效率更高

使用 CodeBuddy 插件，开发者在不离开代码编写环境的前提下就能进行 AI 技术问题的提问或者获取代码。并且可以直接在 IDE 中插入 AI 提供的代码片段，效率更高。

自动识别上下文

CodeBuddy 可以自动识别 IDE 编辑器中可视范围内的所有代码，并基于代码内容提供补全的代码信息。业务代码越多，补全的代码越准确。

核心价值

提高开发生产力，缩短产品研发周期

为开发者提供切实有效的编程辅助工具，帮助开发者更快完成项目开发，提升人效、提高生产力。

帮助实现人效管理，掌控团队研发生产力

提供实时的团队开发提效效果数据，提供全面的研发投入数据依据，帮助管理者进行人效管理。

版权所有：腾讯云计算（北京）有限责任公司

安装插件
JetBrains IDE 使用插件

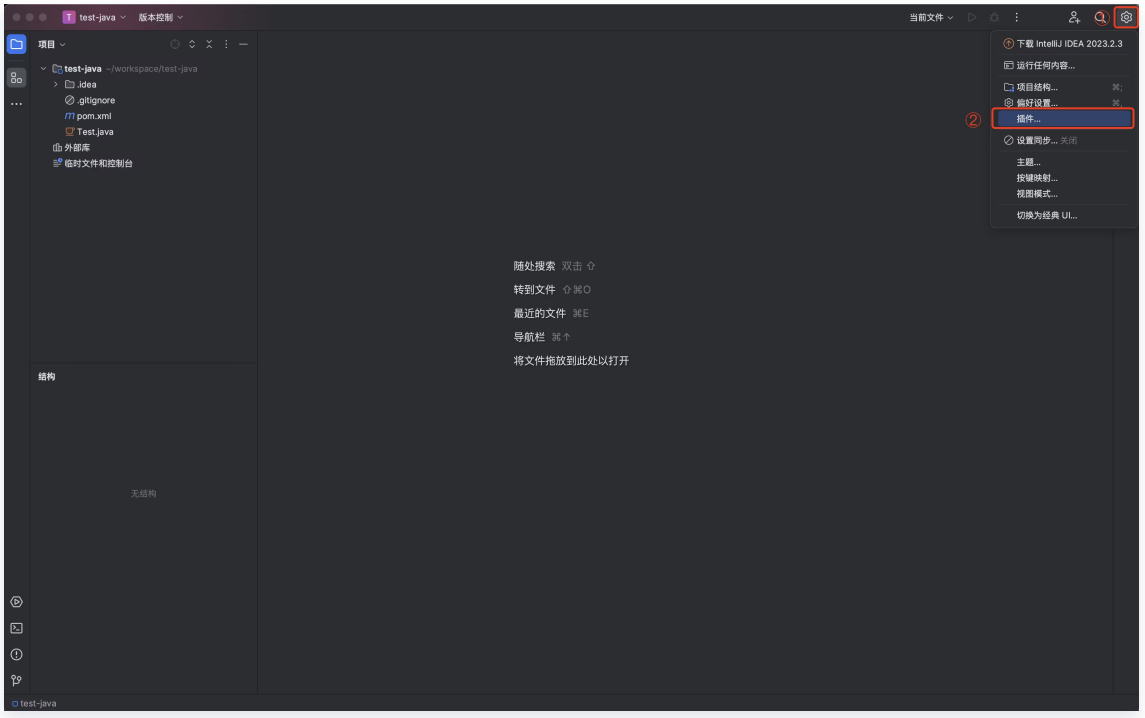
最近更新时间：2026-07-20 19:49:31

前提

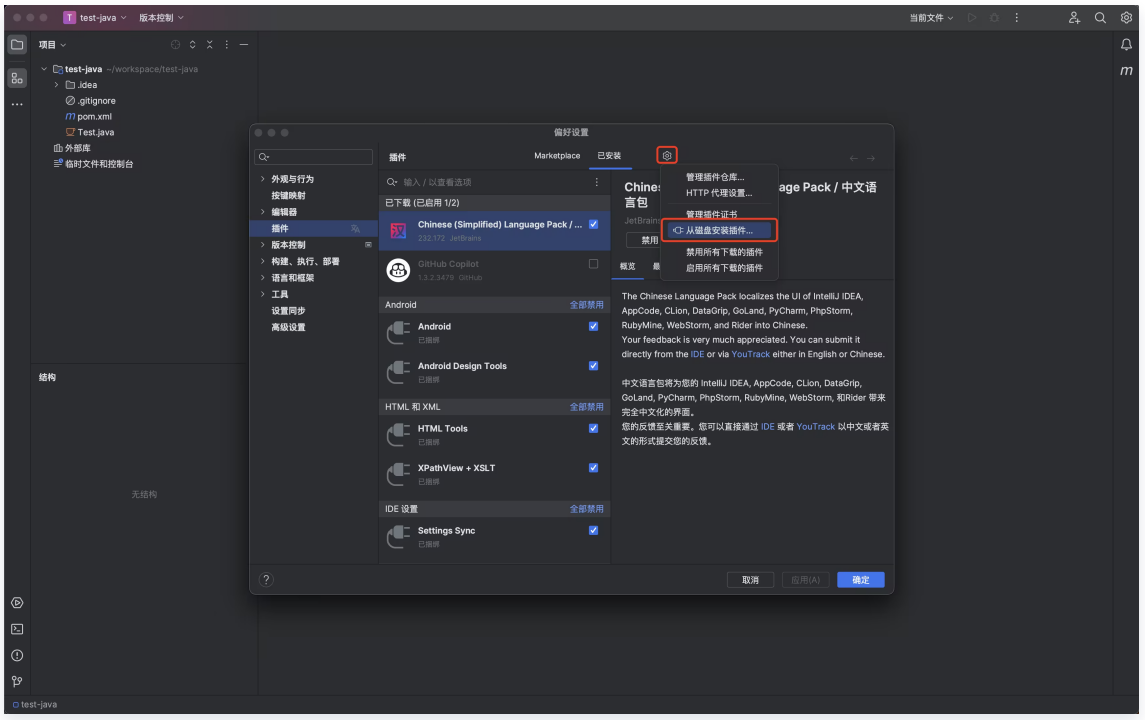
1. 使用 macOS 系统的电脑安装 JetBrains IDE。

   下载任意 JetBrains 的 IDE 产品，例如 IntelliJ IDEA、WebStorm、PyCharm、PhpStorm、GoLand 等 IDE。
2. 下载 AI 代码助手，单击 链接 下载插件。
3. 安装 AI 代码助手插件。

   初次使用，需要使用上面的 链接 下载对应的插件包，然后手动安装离线插件包，如下图所示：



从您本地文件系统选择前面下载好的离线插件包即可（不要解压 zip 包）。如下图所示：



安装成功后的效果：



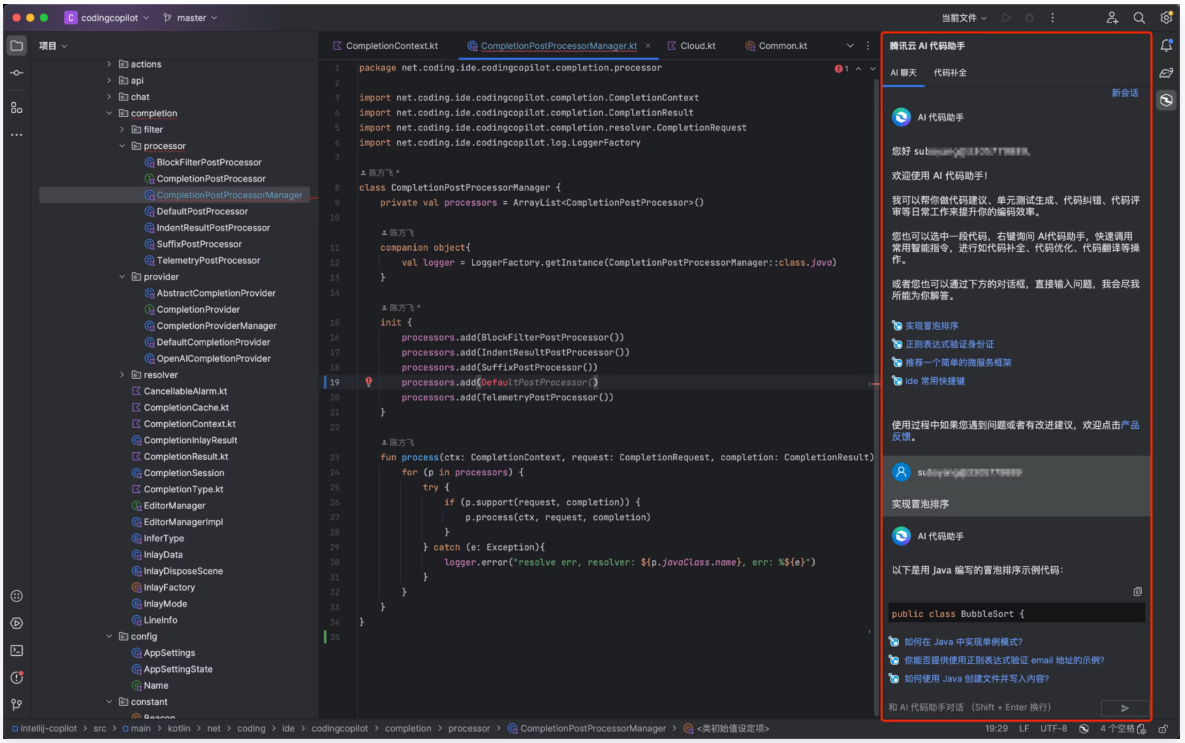
开始使用 CodeBuddy

CodeBuddy 主要提供两类功能：AI 助手对话功能和代码补全功能。

AI 助手对话功能

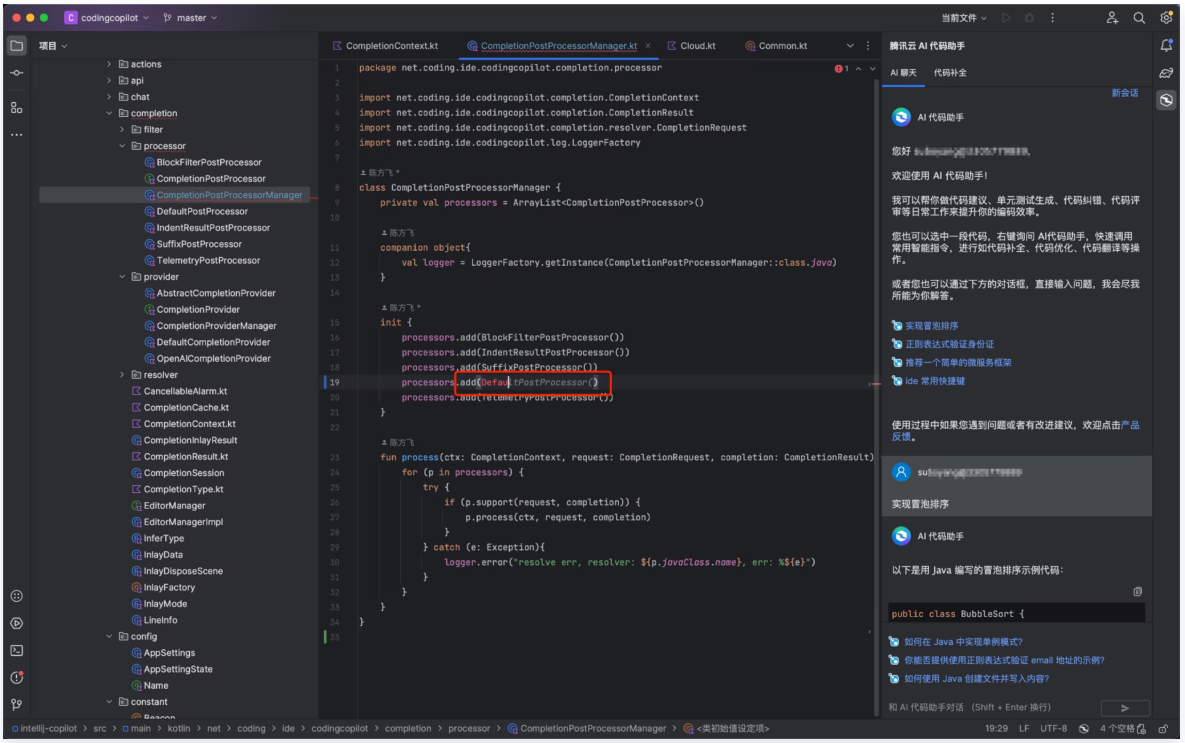
AI 助手对话面板与传统的聊天软件类似，只是开发者只能与 AI 助手进行对话，并且只能咨询一些有关编程技术的问题。对话面板适合咨询一些比较泛的问题，一些复杂的问题，需要开发者多回合沟通才能得到最终答案，并且答案

的形式是纯文字或者是代码 + 文字说明等形式。如下所示：



代码补全功能

目前支持的主流开发语言的补全，例如：c、cpp、css、go、html、java、javascript、kotlin、typescript、react、python、sql、markdown 等。只需要使用编辑器打开代码文件，编辑代码文件时候，AI 代码助手会在合适时机和位置自动触发智能代码补全提示，如下图所示：



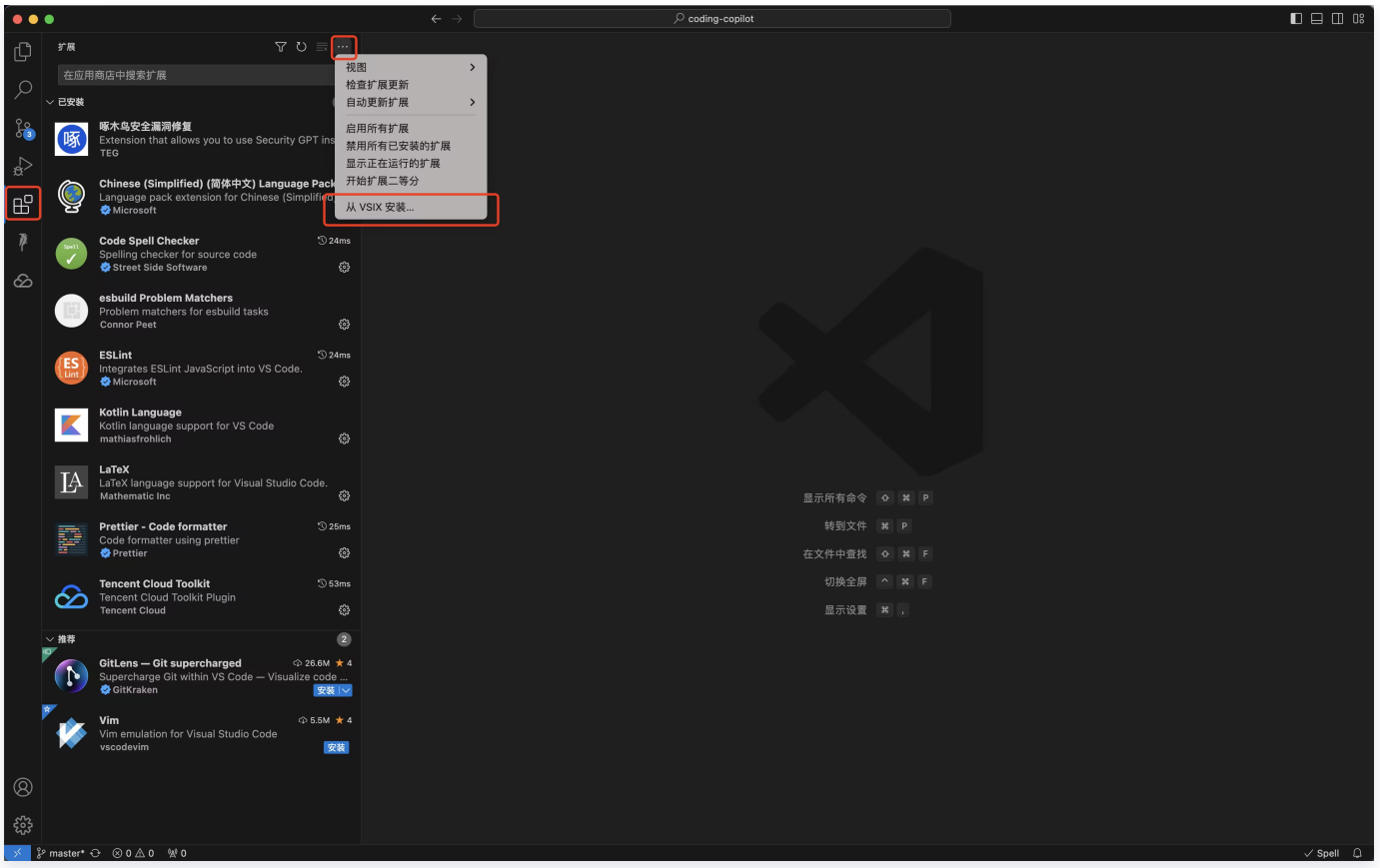
Visual Studio Code 使用插件

最近更新时间：2026-07-20 19:49:32

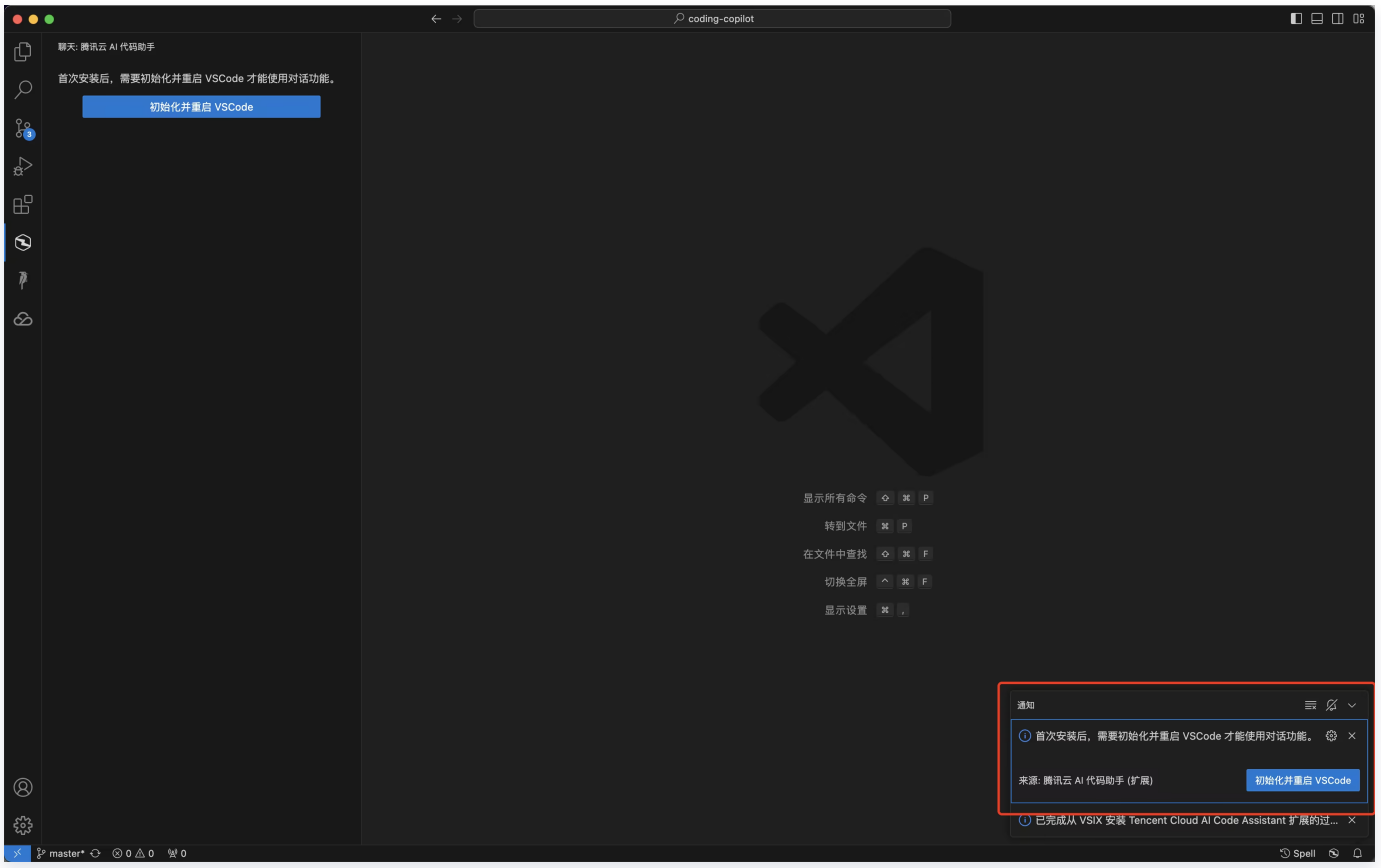
前提条件

1. 安装 Visual Studio Code（版本要求1.79以上）。
2. 下载 AI 代码助手，单击 链接 下载插件并完成安装。
3. 安装 AI 代码助手插件。

   初次使用，需要使用上面的 链接 下载对应的插件包，然后手动安装离线插件包，如下图所示：



由于插件部分能力使用 VSCode 实验 API，所以安装插件后会提示需要**初始化并重启 VSCode**，如下图所示：



请单击**初始化并重启 VSCode**。

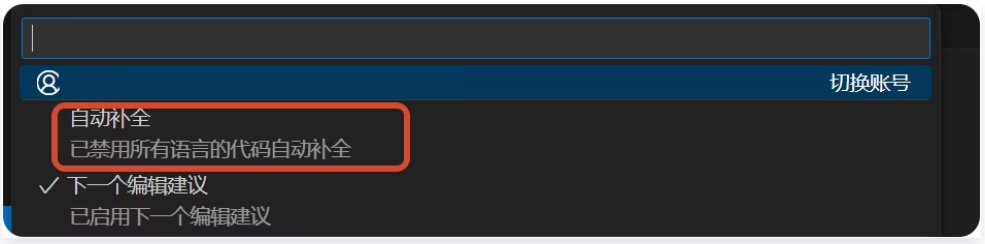
> **注意：**
> - 如果不单击**初始化并重启 VSCode**，AI 助手对话功能将不可用。
> - 如果 VSCode 版本低于1.79.0，AI 助手对话功能将不可用。

4. 登录插件，指引参见登录及退出 。

启用与关闭代码补全

启用

VSCode 右下角单击 AI 代码助手 Logo，顶部会出现详细设置，单击**启用补全**，即可启用。



禁用

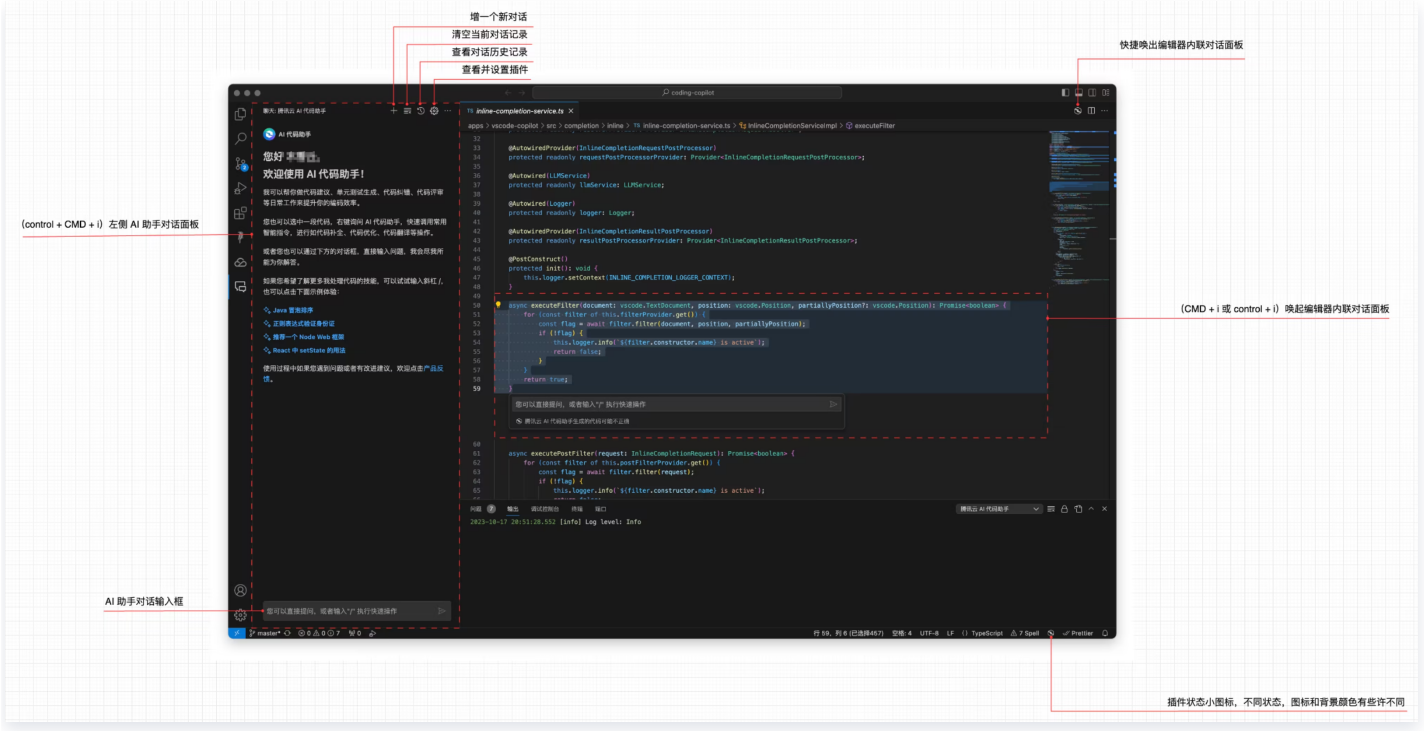
IDE 右下角单击 AI 代码助手 Logo，在详细设置中单击自动补全关闭。

开始使用 CodeBuddy

腾讯云 AI 代码助手主要提供两类功能：AI 助手对话功能和代码补全功能。

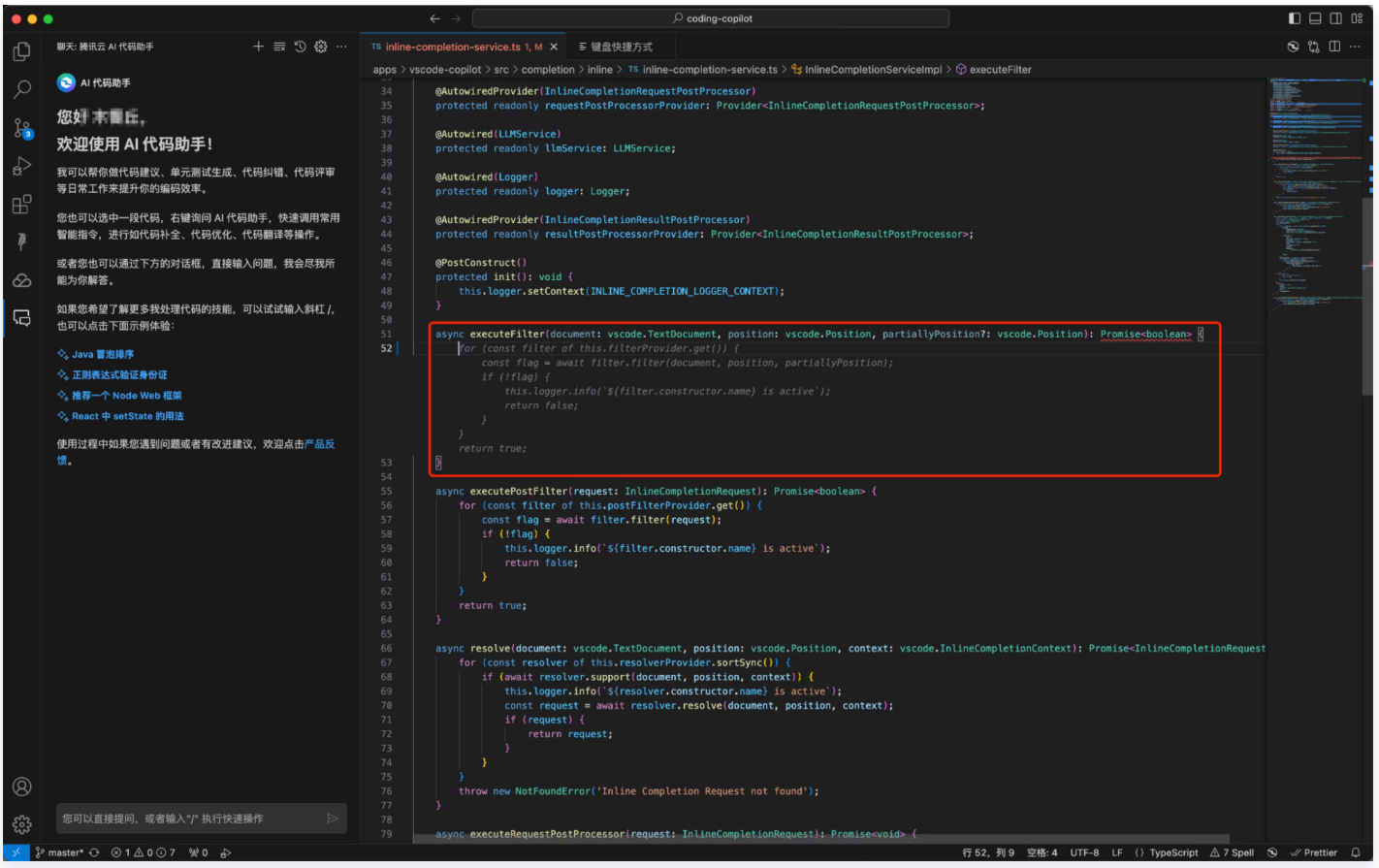
AI 助手对话功能

AI 助手对话面板包括：左侧对话面板和编辑器内联对话面板。左侧对话面板，和传统的聊天软件类似，只是开发者只能与 AI 助手进行对话，并且只能咨询一些有关编程技术的问题。
相较于编辑器内联对话面板，左侧对话面板更适合咨询一些比较泛的问题，并且问题答案的不确定性很大，需要开发者多回合沟通才能得到最终答案，并且答案不一定有代码，或者是代码 + 文字说明等各种各样的形式。至于答案里面的代码部分是否需要复制或者插入到代码文件中，操作相对灵活自由，开发者自由选择。
而编辑器内联对话面板是与选中的代码进行对话，要求 AI 助手对选中的代码做处理。处理的方式可以通过自然语言交互。对选中的代码处理后的结果，是明确要求开发者选择是接受，还是取消。左侧对话面板和编辑器内联对话面板的概览图如下所示：



代码补全功能

目前支持的主流开发语言的补全，例如：c、cpp、css、go、html、java、javascript、kotlin、typescript、react、python、sql、markdown 等。
只需要使用编辑器打开代码文件，编辑代码文件时，AI 代码助手会在合适时机和位置自动触发智能代码补全提示，如下图所示：


coding-copilot
聊天: 腾讯云 AI 代码助手
AI 代码助手
欢迎使用 AI 代码助手!
我可以帮你做代码建议、单元测试生成、代码纠错、代码评审等日常工作来提升你的编码效率。
您也可以选中一段代码，右键询问 AI 代码助手，快速调用常用智能指令，进行如代码补全、代码优化、代码翻译等操作。
或者您也可以通过下方的对话框，直接输入问题，我会尽我所能为你解答。
如果您希望了解更多我处理代码的技能，可以试试输入斜杠 /，也可以点击下面示例体验：
Java 冒泡排序
正则表达式验证身份证
推荐一个 Node Web 框架
React 中 setState 的用法
使用过程中如果您遇到问题或者有改进建议，欢迎点击产品反馈。
您可以直接提问，或者输入"/" 执行快速操作
inline-completion-service.ts 1, M
键盘快捷方式
apps > vscode-copilot > src > completion > inline > inline-completion-service.ts > InlineCompletionServiceImpl > executeFilter
@AutowiredProvider(InlineCompletionRequestPostProcessor)
protected readonly requestPostProcessorProvider: Provider<InlineCompletionRequestPostProcessor>;
@Autowired(LLMService)
protected readonly llmService: LLMService;
@Autowired(Logger)
protected readonly logger: Logger;
@AutowiredProvider(InlineCompletionResultPostProcessor)
protected readonly resultPostProcessorProvider: Provider<InlineCompletionResultPostProcessor>;
@PostConstruct()
protected init(): void {
this.logger.setContext(INLINE_COMPLETION_LOGGER_CONTEXT);
}
async executeFilter(document: vscode.TextDocument, position: vscode.Position, partiallyPosition?: vscode.Position): Promise<boolean> {
for (const filter of this.filterProvider.get()) {
const flag = await filter.filter(document, position, partiallyPosition);
if (!flag) {
this.logger.info(`${filter.constructor.name} is active`);
return false;
}
}
return true;
}
async executePostFilter(request: InlineCompletionRequest): Promise<boolean> {
for (const filter of this.postFilterProvider.get()) {
const flag = await filter.filter(request);
if (!flag) {
this.logger.info(`${filter.constructor.name} is active`);
return false;
}
}
return true;
}
async resolve(document: vscode.TextDocument, position: vscode.Position, context: vscode.InlineCompletionContext): Promise<InlineCompletionRequest
for (const resolver of this.resolverProvider.sortSync()) {
if (await resolver.support(document, position, context)) {
this.logger.info(`${resolver.constructor.name} is active`);
const request = await resolver.resolve(document, position, context);
if (request) {
return request;
}
}
}
throw new NotFoundError('Inline Completion Request not found');
}
async executeRequestPostProcessor(request: InlineCompletionRequest): Promise<void> {
master*
行 52，列 9
空格: 4
UTF-8
LF
TypeScript
7 Spell
Prettier

快速开始
产品安装

最近更新时间：2026-07-20 19:49:30

CodeBuddy 插件

插件对主流 IDE 版本要求

IDE	最低版本要求
Visual Studio Code	1.82
Visual Studio	17.6（VS 2022）
IntelliJ IDEA	2022.2
PyCharm	2022.2
GoLand	2022.2
CLion	2022.2
PhpStorm	2022.2
Android Studio	Flamingo | 2022.2.1
微信开发者工具 IDE	1.06.2409140

注意：

- 其他更多 JetBrains 系列 IDE 版本要求请参见 JetBrains 插件市场 。
- 除市场版本外，JetBrains 另外提供低版本兼容插件，最低可兼容至2020.3，您可点击 此处 下载。受低版本 JetBrains 能力限制，此插件包无法体验最新的产品功能，建议尽量升级 JetBrains 版本，使用正式版本插件。

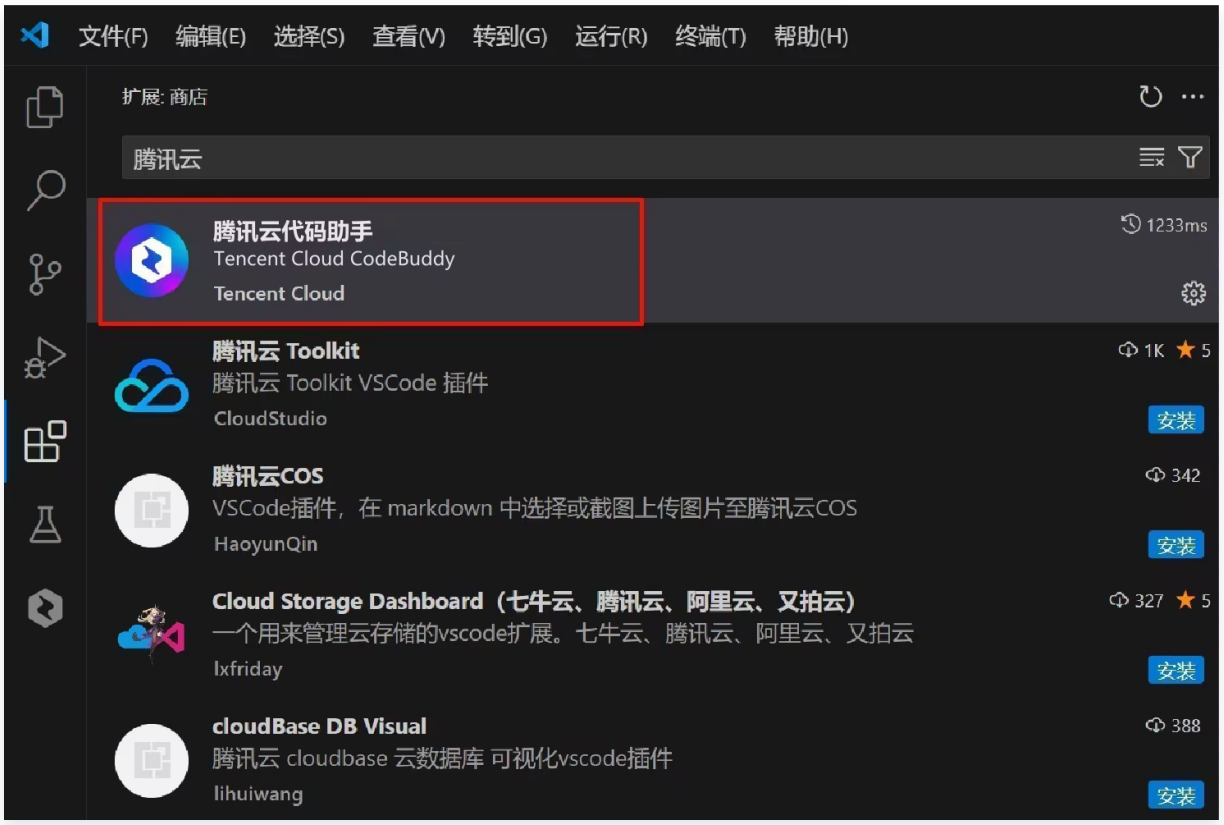
VS Code

1. 安装 Visual Studio Code：建议安装版本 1.82 及以上，单击前往 VS Code 官网下载 。
2. 安装插件：
   - **快速安装：** 单击一键安装腾讯云代码助手 ，即可快速跳转到腾讯云代码助手的安装界面。

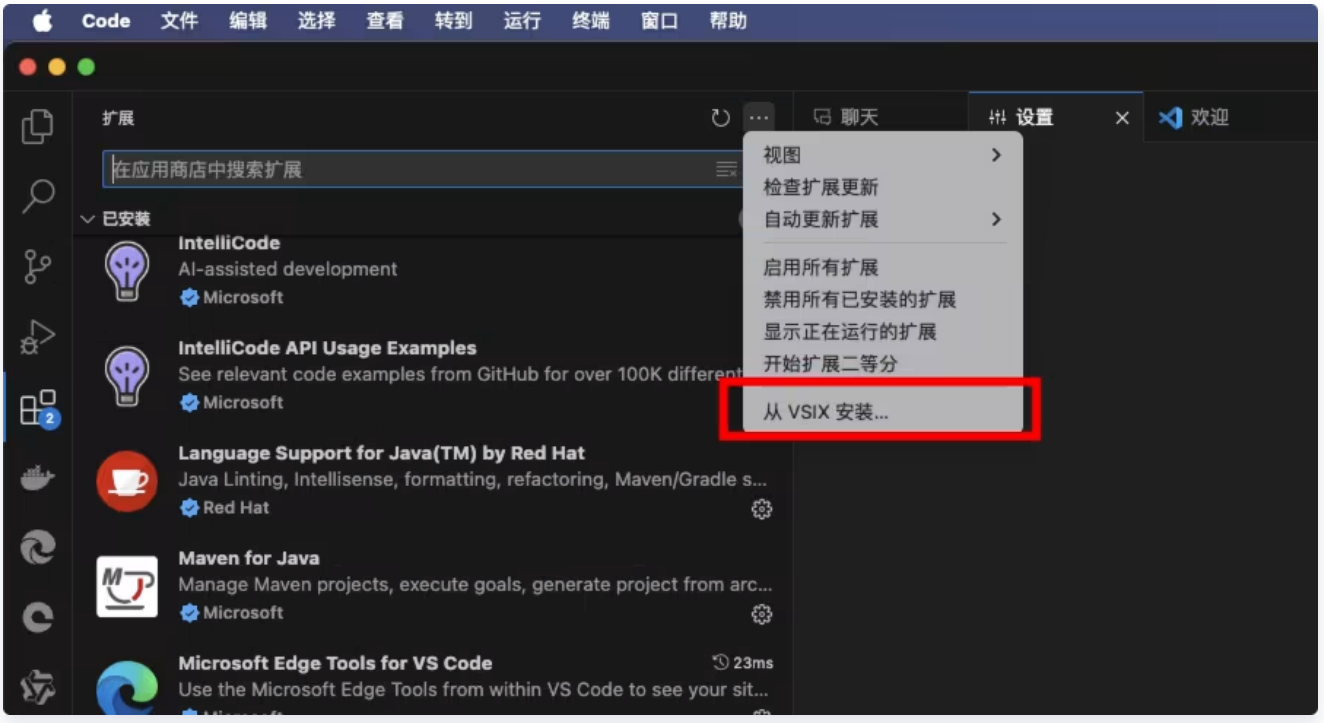
说明：

> 使用快速安装时，请保证已正确安装 Visual Studio Code，否则无法进行一键安装。

- **从 IDE 插件市场安装**：您也可以在 VS Code 插件市场手动搜索**腾讯云代码助手**并下载安装。



- **下载安装包**：您也可以 单击下载最新版本安装包 ，并前往 VS Code 手动安装。



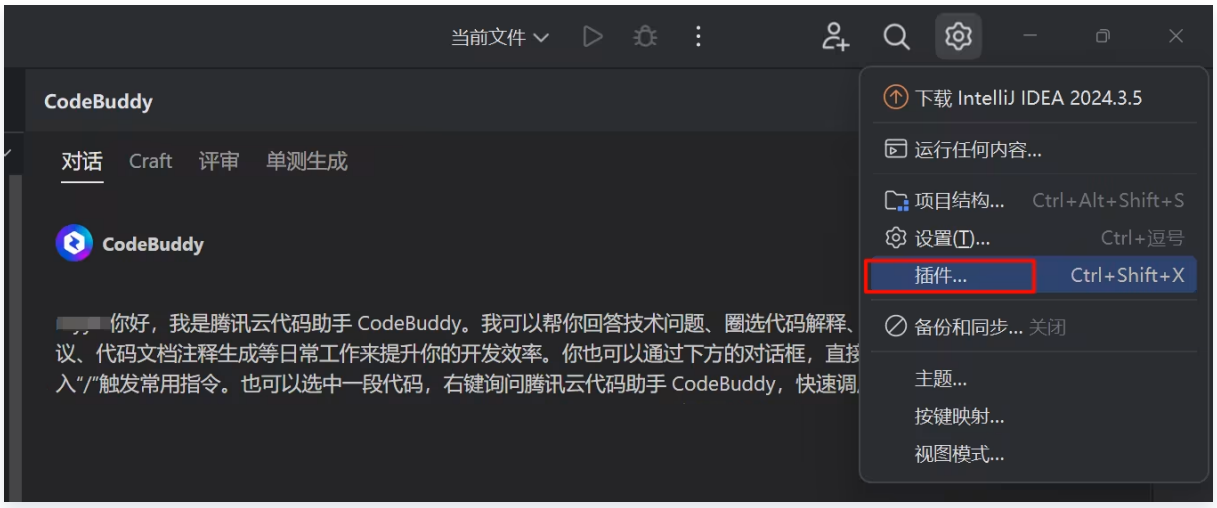
Visual Studio

以 VS 2022 为例（VS 2026 安装操作相同），具体详情请参见 Visual Studio 2022-安装指引 。

JetBrains IDEs

1. 安装 JetBrains IDEs：下载并安装任意 JetBrains 的 IDE 产品，例如 IntelliJ IDEA、WebStorm、PyCharm、PhpStorm、GoLand 等 IDE。
2. 安装插件：
   - **从 IDE 插件市场安装。**

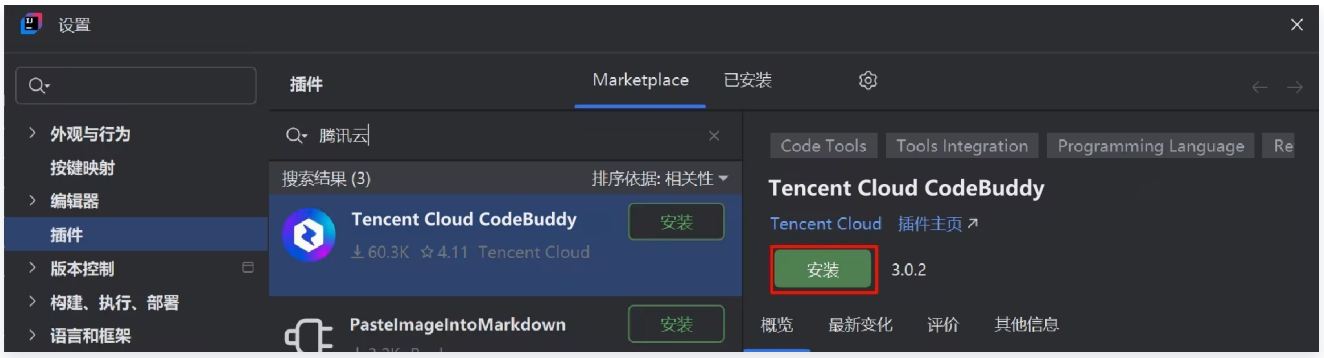
     a. 打开插件市场：在右上角的设置中，单击**插件**。



     b. 搜索**腾讯云代码助手**。

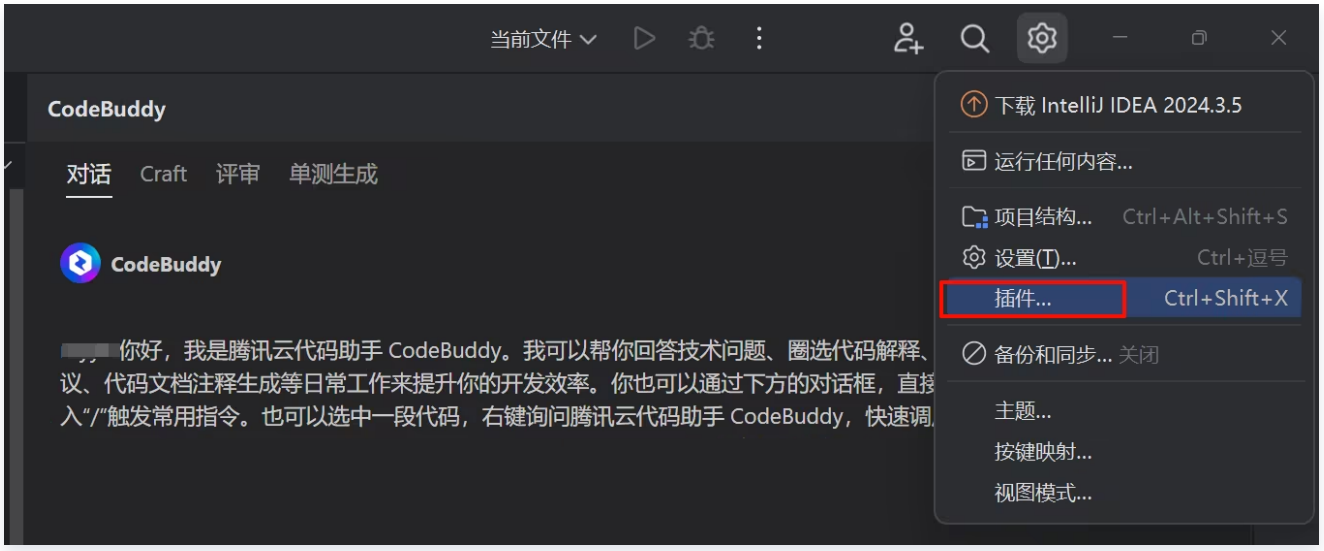


     c. 在搜索结果中单击**安装**。

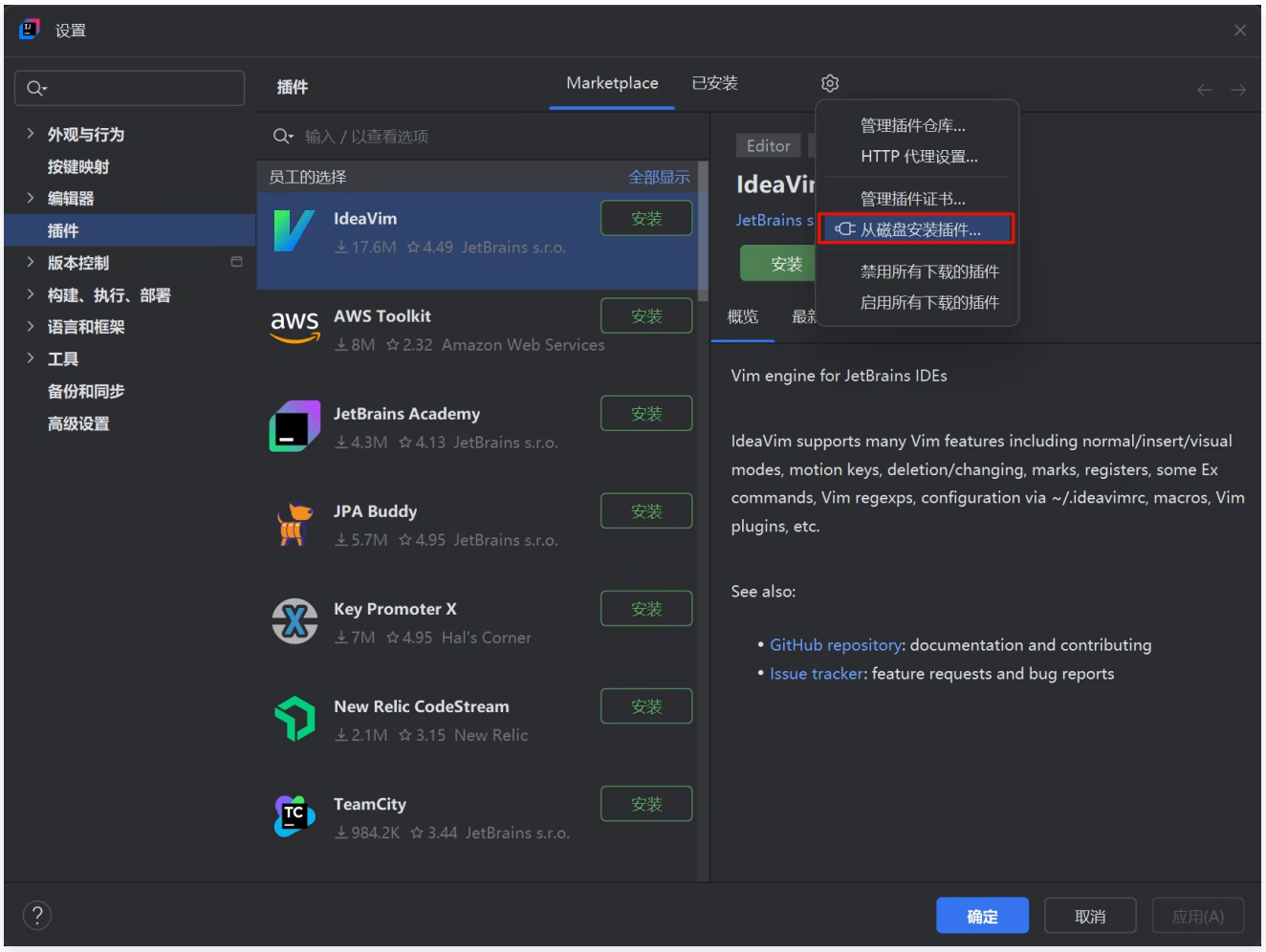


- **下载安装包。**

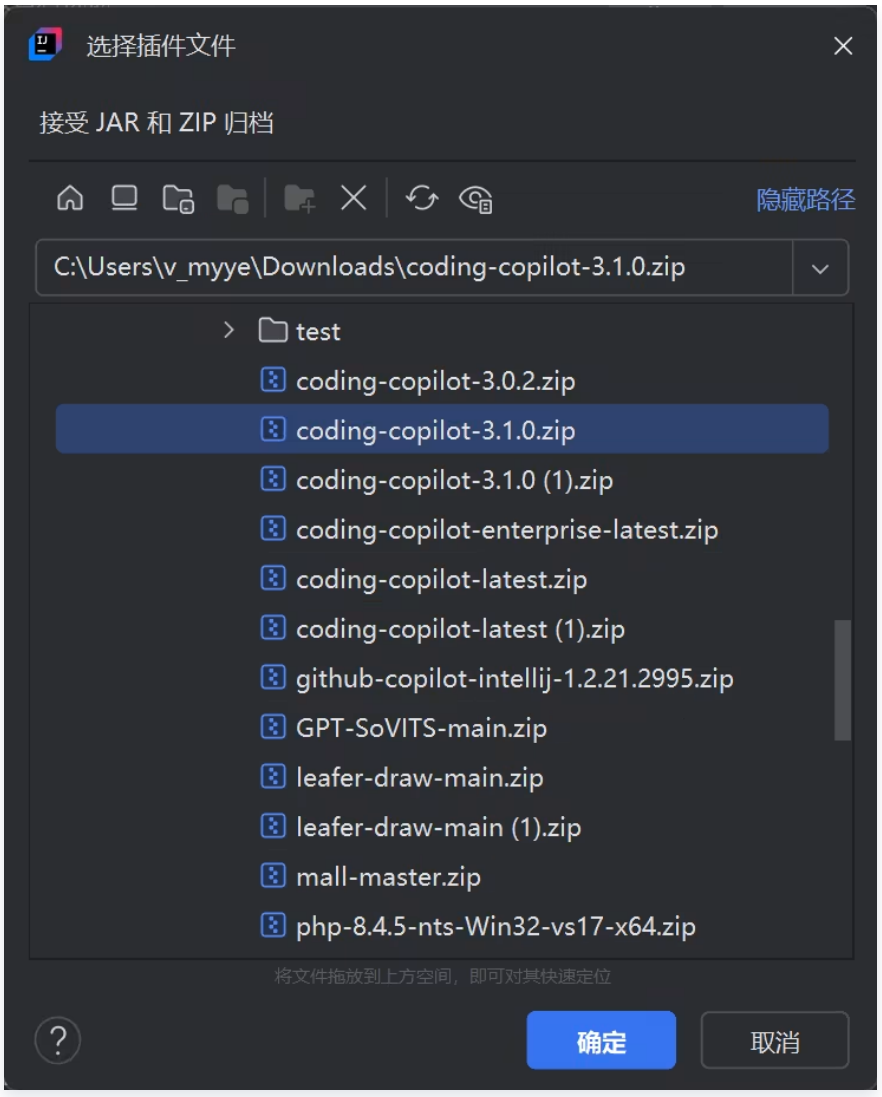
  a. 单击下载**最新版本安装包**，并前往 JetBrains IDEs 手动安装。



  b. 选择**从磁盘安装插件**。


设置
插件
Marketplace
已安装
输入 / 以查看选项
外观与行为
按键映射
编辑器
插件
版本控制
构建、执行、部署
语言和框架
工具
备份和同步
高级设置
员工的选择
全部显示
IdeaVim
17.6M 4.49 JetBrains s.r.o.
安装
AWS Toolkit
8M 2.32 Amazon Web Services
JetBrains Academy
4.3M 4.13 JetBrains s.r.o.
JPA Buddy
5.7M 4.95 JetBrains s.r.o.
Key Promoter X
7M 4.95 Hal's Corner
New Relic CodeStream
2.1M 3.15 New Relic
TeamCity
984.2K 3.44 JetBrains s.r.o.
管理插件仓库...
HTTP 代理设置...
管理插件证书...
从磁盘安装插件...
禁用所有下载的插件
启用所有下载的插件
Editor
概览
Vim engine for JetBrains IDEs
IdeaVim supports many Vim features including normal/insert/visual modes, motion keys, deletion/changing, marks, registers, some Ex commands, Vim regexps, configuration via ~/.ideavimrc, macros, Vim plugins, etc.
See also:
GitHub repository: documentation and contributing
Issue tracker: feature requests and bug reports
确定
取消
应用(A)

c. 选择已下载的插件包。



Android Studio

1. **下载 Android Studio**：访问 Android Studio 进行下载。
2. **安装插件**：安装插件的方式与 JetBrains IDEs 一致，可参考进行安装。

微信开发者工具 IDE

1. 前往 微信开发者平台 IDE 下载微信开发者工具，注意需要下载最新**开发版 Nightly Build**。

下载 / 开发版更新日志

微信开发者工具（开发版 Nightly Build）下载地址与更新日志

> 日常构建版本(基于 NW.js 0.54.1)，用于尽快修复缺陷和敏捷上线小的特性；开发自测验证，稳定性欠佳。 1.06系列版本不再支持 Windows7，建议开发者升级 Windows 版本。 1.05系列版本还是支持 Windows7 系统，不过原则上只会修复重大bug, 不会再随着1.06的节奏发版。 1.05 内核稳定版下载地址

2.06.2508042 macOS x64、macOS ARM64

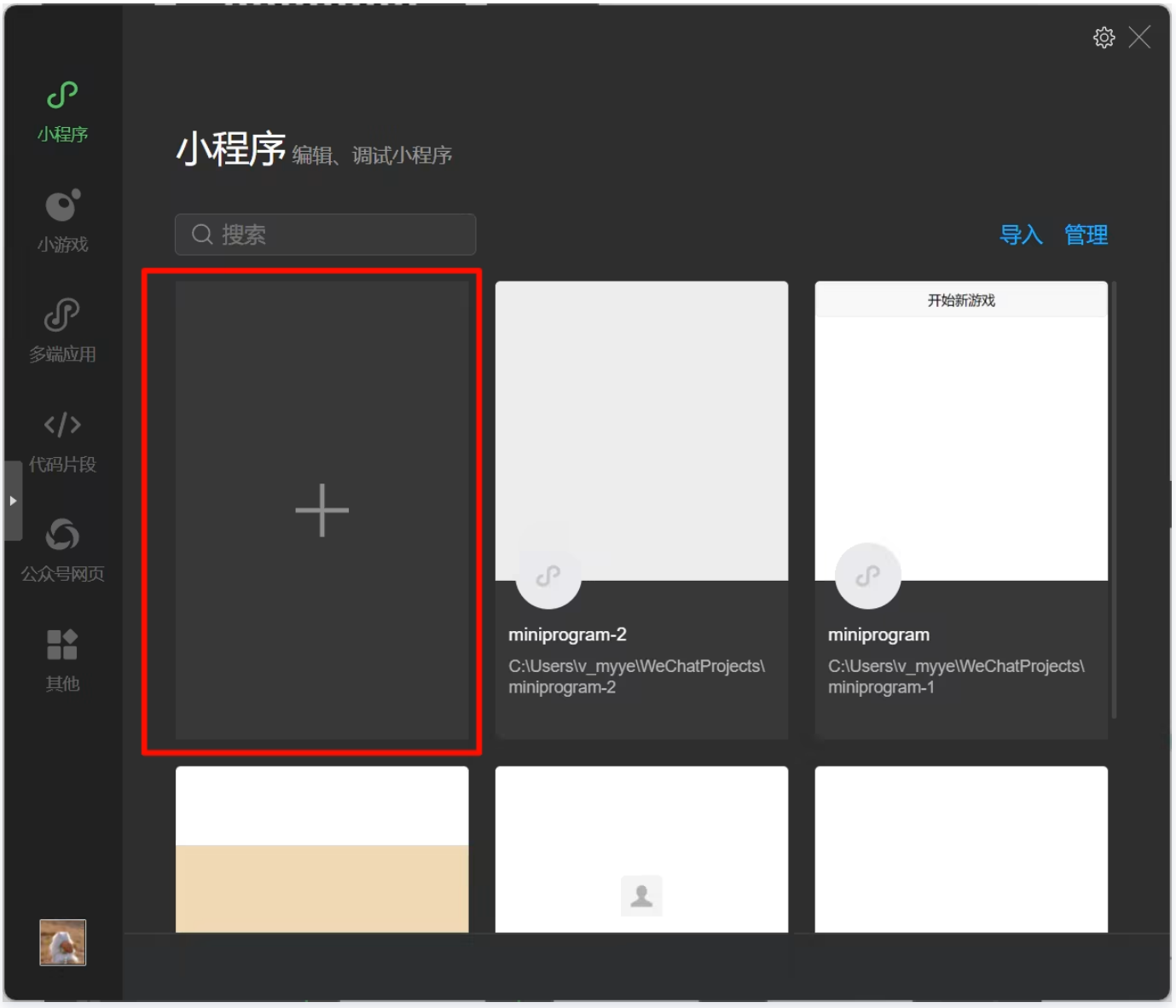
> 日常构建版本(基于 NW.js 0.81.1), 更新了底层内核，使用 node 20.7.0 以及 chromium 118，支持新特性；开发自测验证，稳定性欠佳。

2.01.2508072 Windows 64、Windows 32、macOS x64、macOS ARM64

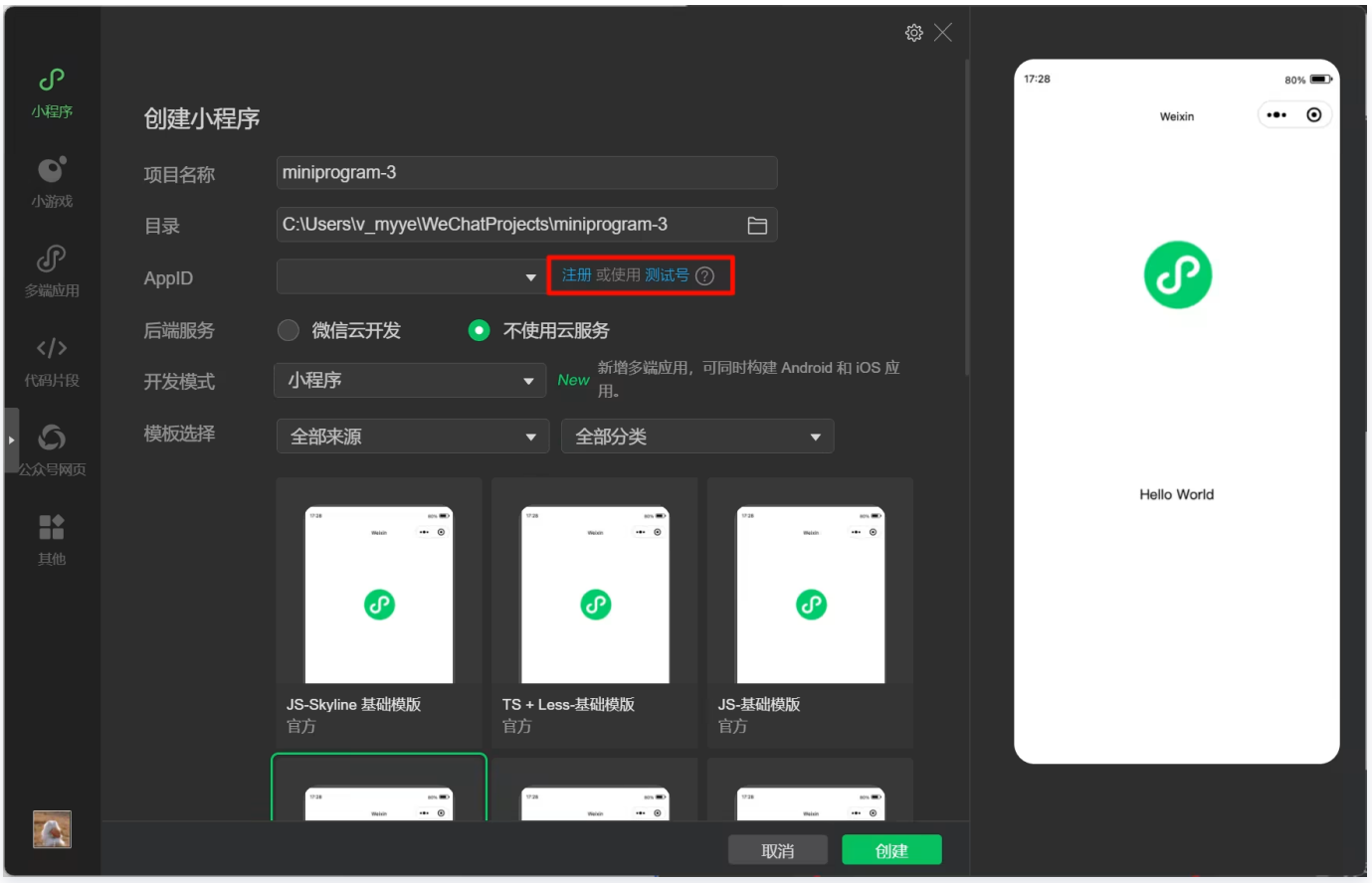
2. 安装完成后，打开微信开发者工具，使用微信扫码登录微信开发者工具。



3. 登录后，创建您所需的项目，或者选择您以前所创建的项目并打开。



4. 如果选择创建项目，则根据您的实际需求进行创建。如果没有 AppID，则可以单击**注册**完成小程序账号注册；或者直接单击测试号，使用测试号作为 AppID，但使用测试号时目前暂不支持选择微信云开发。



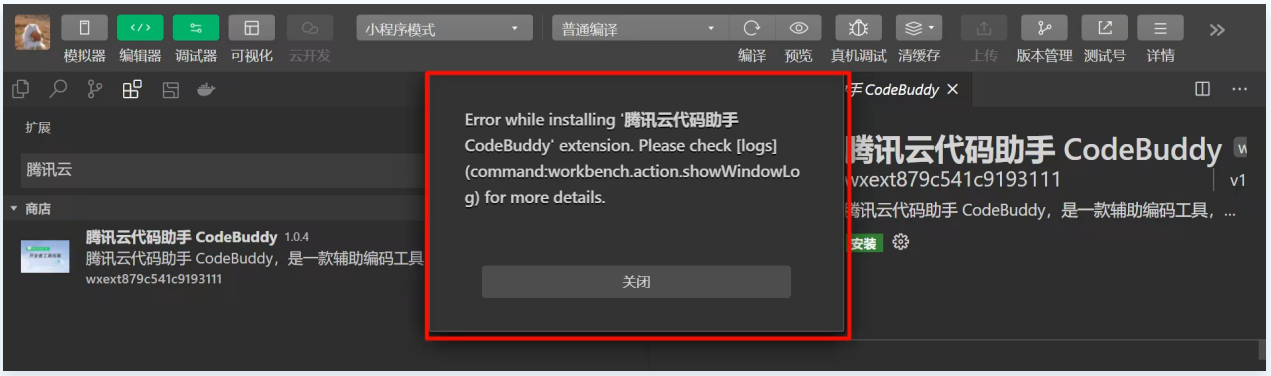
> **注意：**
> 这里所创建的项目属于演示的项目，并非固定，在实际开发项目中需要根据自己的实际情况填写/选择信息，创建属于自己的项目。

5. 单击**扩展区域**图标，展开热门或者在搜索栏中输入**腾讯云代码助手** 或 **CodeBuddy**，单击腾讯云代码助手右边的**安装**，等待安装完成即可。



说明：

腾讯云代码助手插件安装完成后，建议重启一下微信开发者工具。如果出现安装异常，例如下图，您可参考 不同端功能常见问题 进行解决。



CodeBuddy IDE

1. 访问 CodeBuddy IDE 国内版 进行下载。



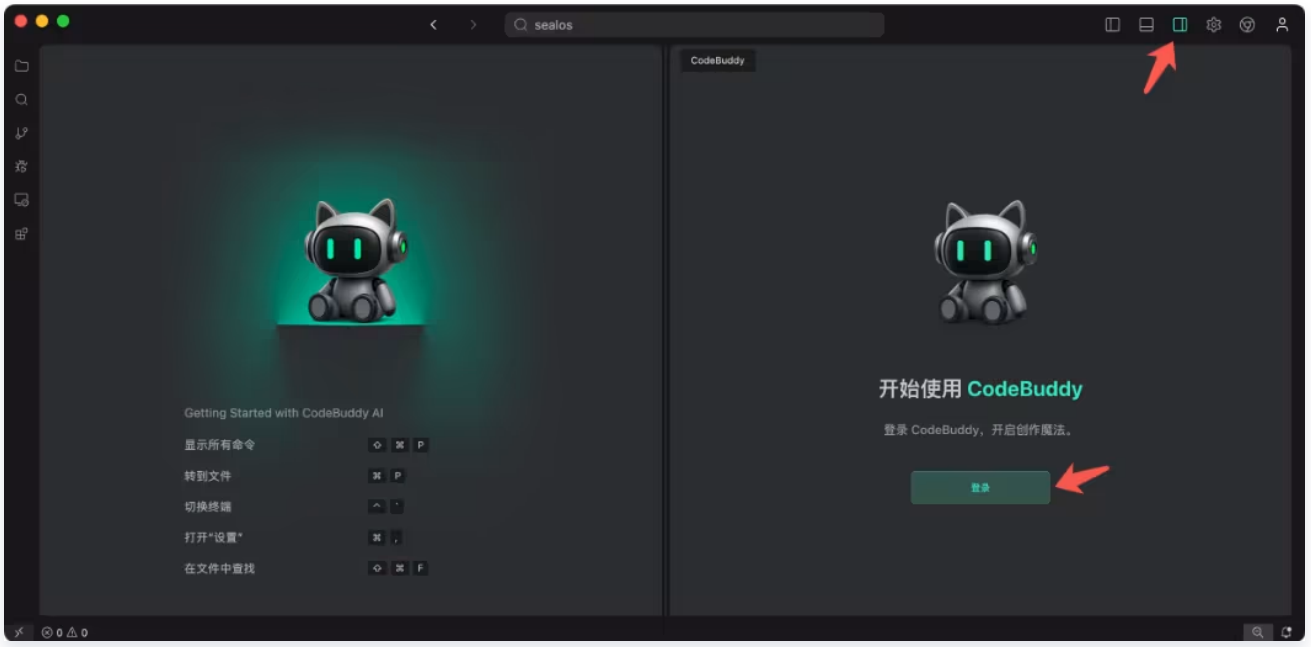
2. 安装 CodeBuddy IDE。

在 Mac 中安装

a. 将左侧安装包拖拽到 Applications 中进行安装。

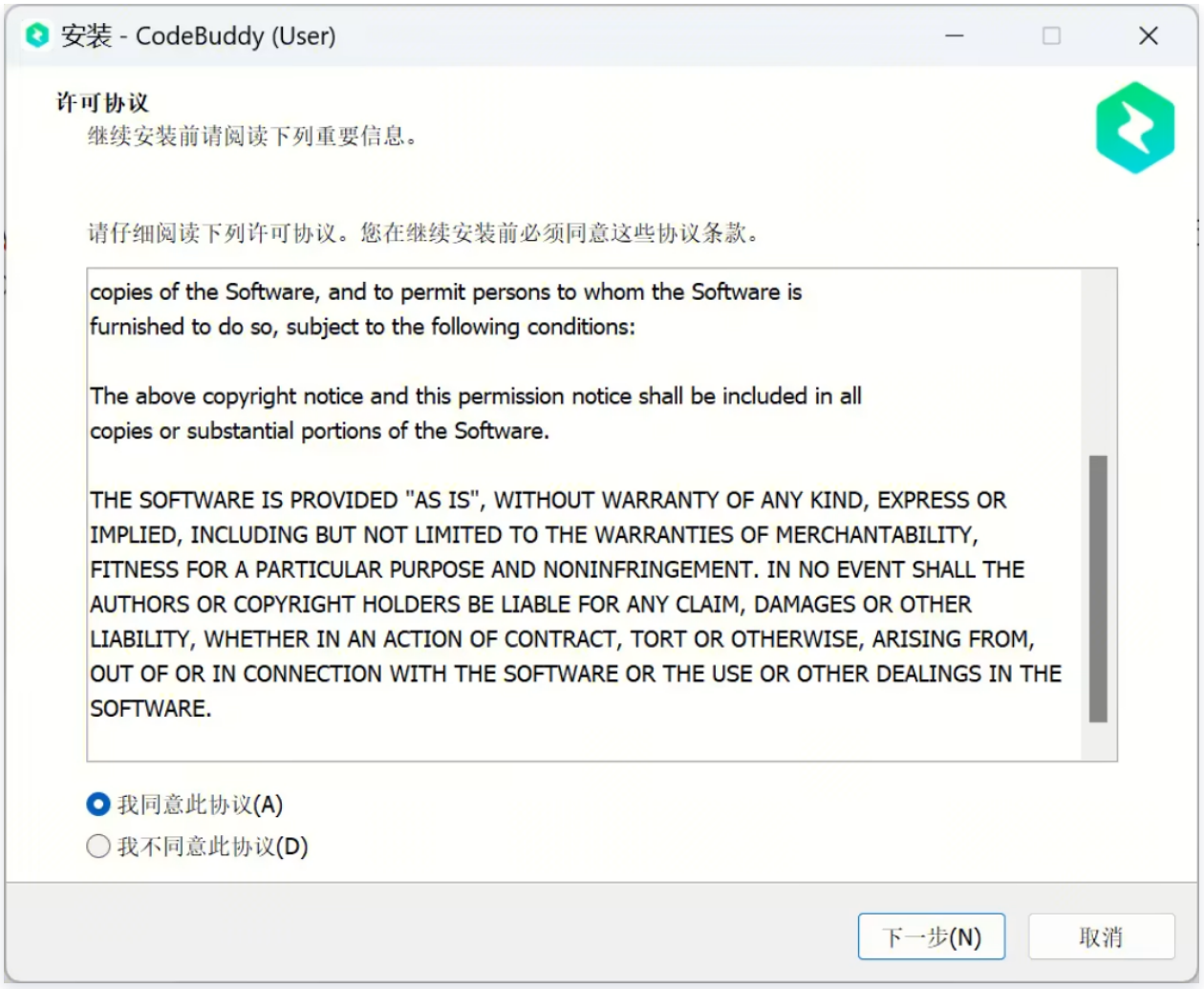


b. 安装成功后，打开 IDE 进行登录即可。

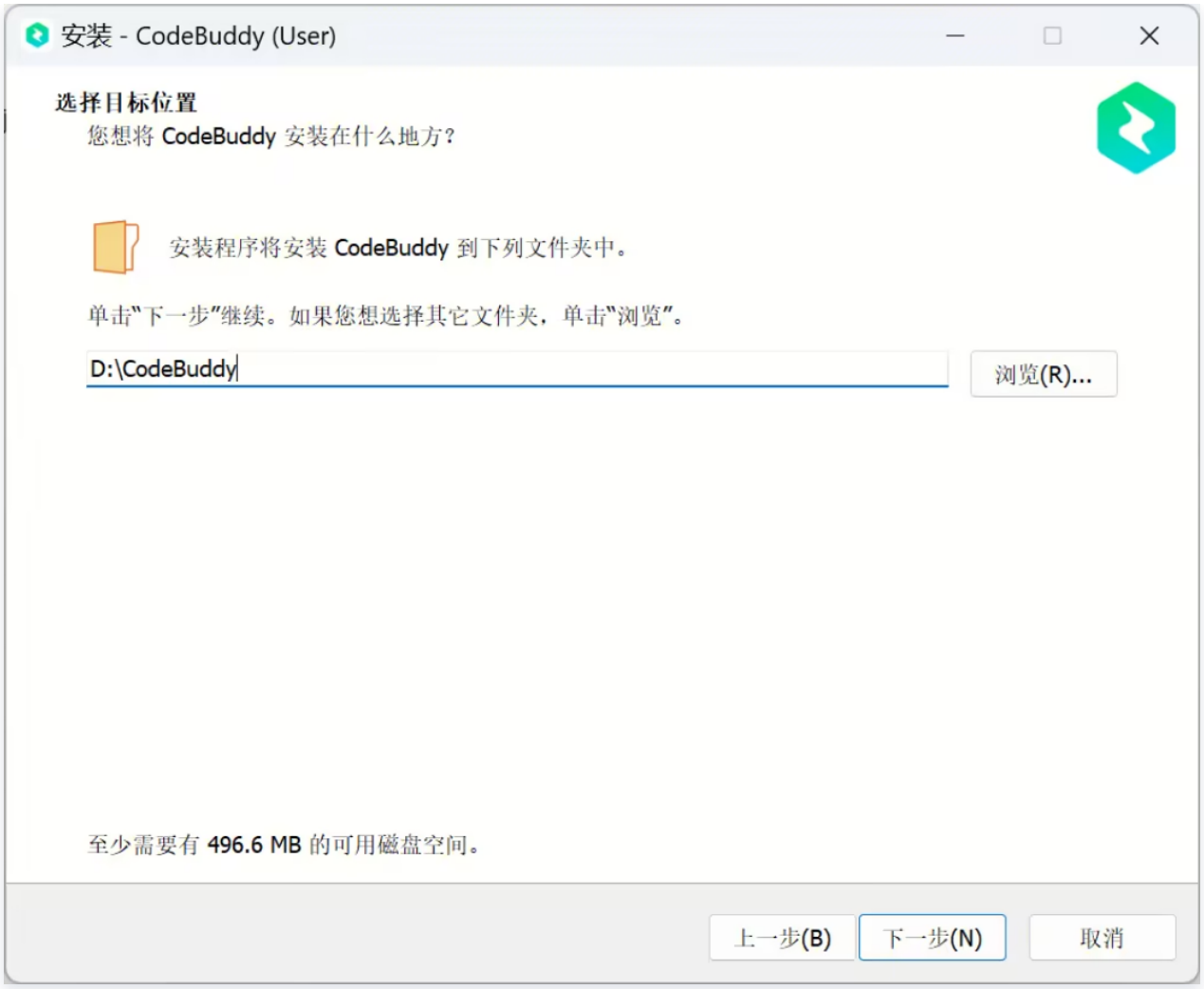


在 Windows 中安装

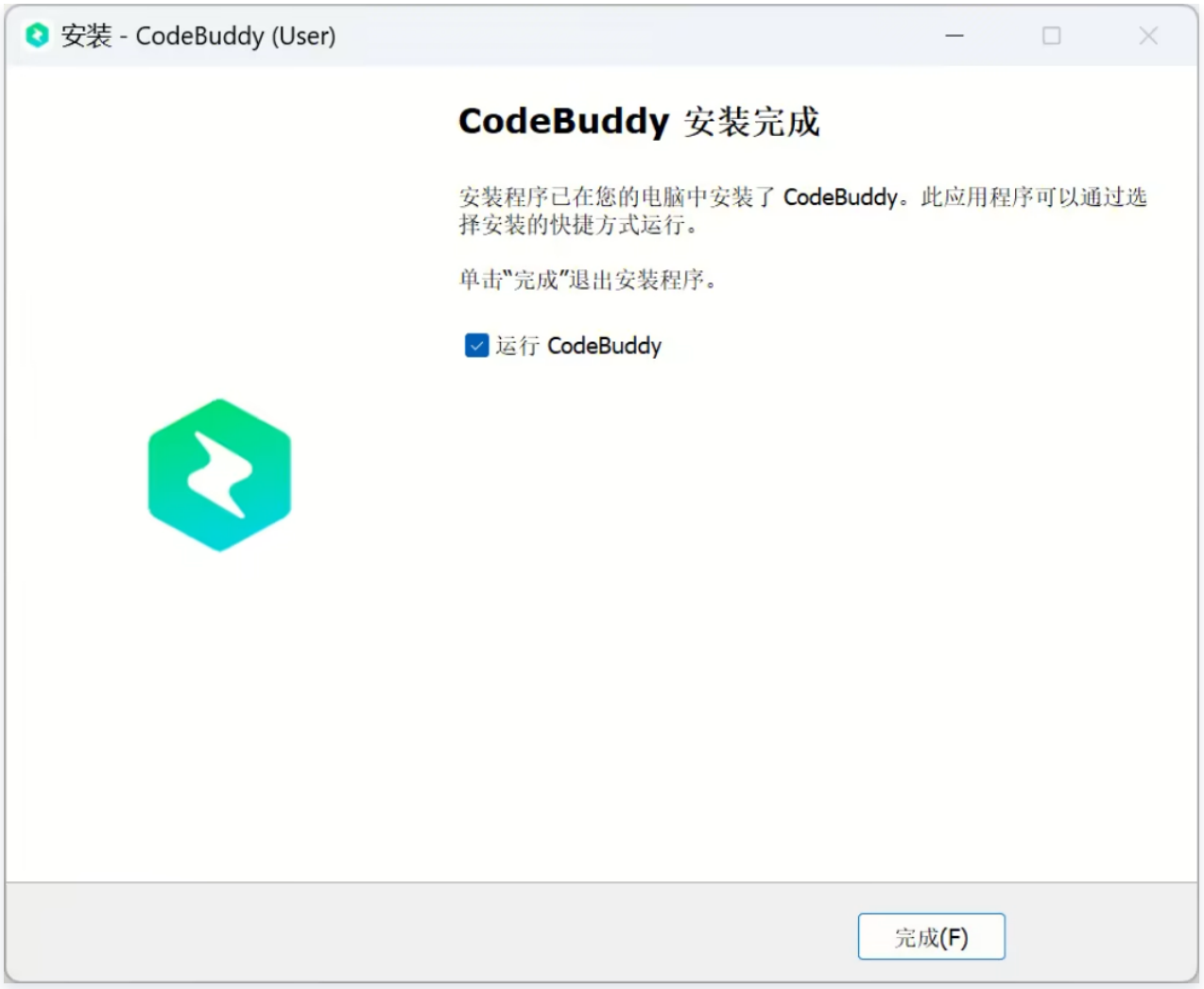
a. 下载安装包后，双击 exe 安装包，接受协议。



b. 选择需要安装的位置，然后一直选择下一步即可。


安装 - CodeBuddy (User)
选择目标位置
您想将 CodeBuddy 安装在什么地方？
安装程序将安装 CodeBuddy 到下列文件夹中。
单击"下一步"继续。如果您想选择其它文件夹，单击"浏览"。
D:\CodeBuddy
浏览(R)...
至少需要有 496.6 MB 的可用磁盘空间。
上一步(B)
下一步(N)
取消

c. 安装完成如下图所示。


安装 - CodeBuddy (User)
CodeBuddy 安装完成
安装程序已在您的电脑中安装了 CodeBuddy。此应用程序可以通过选择安装的快捷方式运行。
单击"完成"退出安装程序。
运行 CodeBuddy
完成(F)

登录及退出

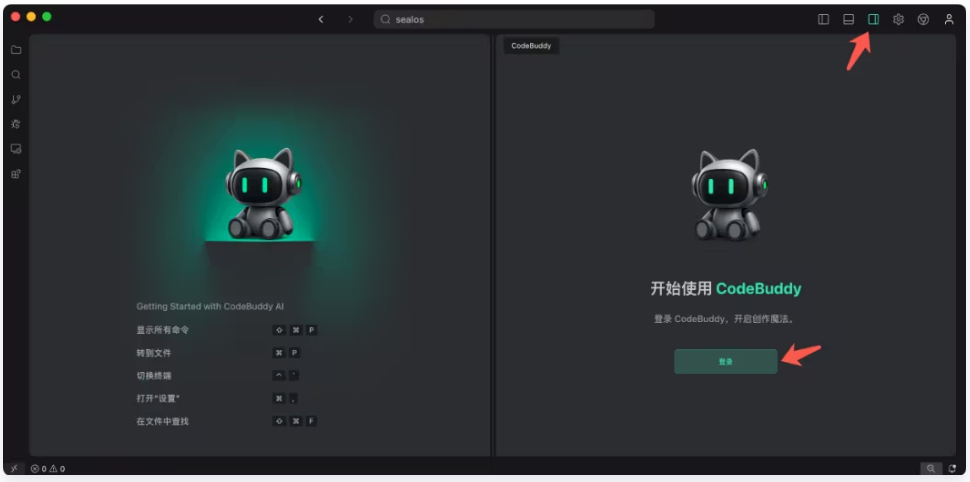
最近更新时间：2026-07-20 19:49:31

个人版

CodeBuddy IDE

登录

1. 打开 IDE，单击**登录**。



2. 选择登录方式，支持选择微信或者手机号登录。



退出登录

1. 单击右上方的**账户**。



2. 在账户中选择**退出登录**即可。



CodeBuddy 插件

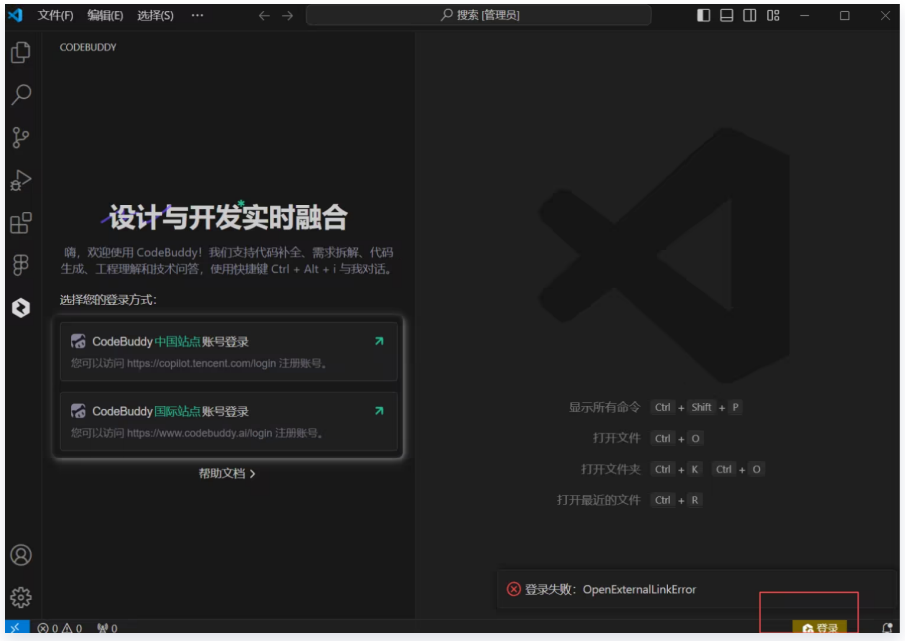
一、安装通用插件

通过 IDE 插件市场下载腾讯云代码助手，详情请参见 产品安装 。

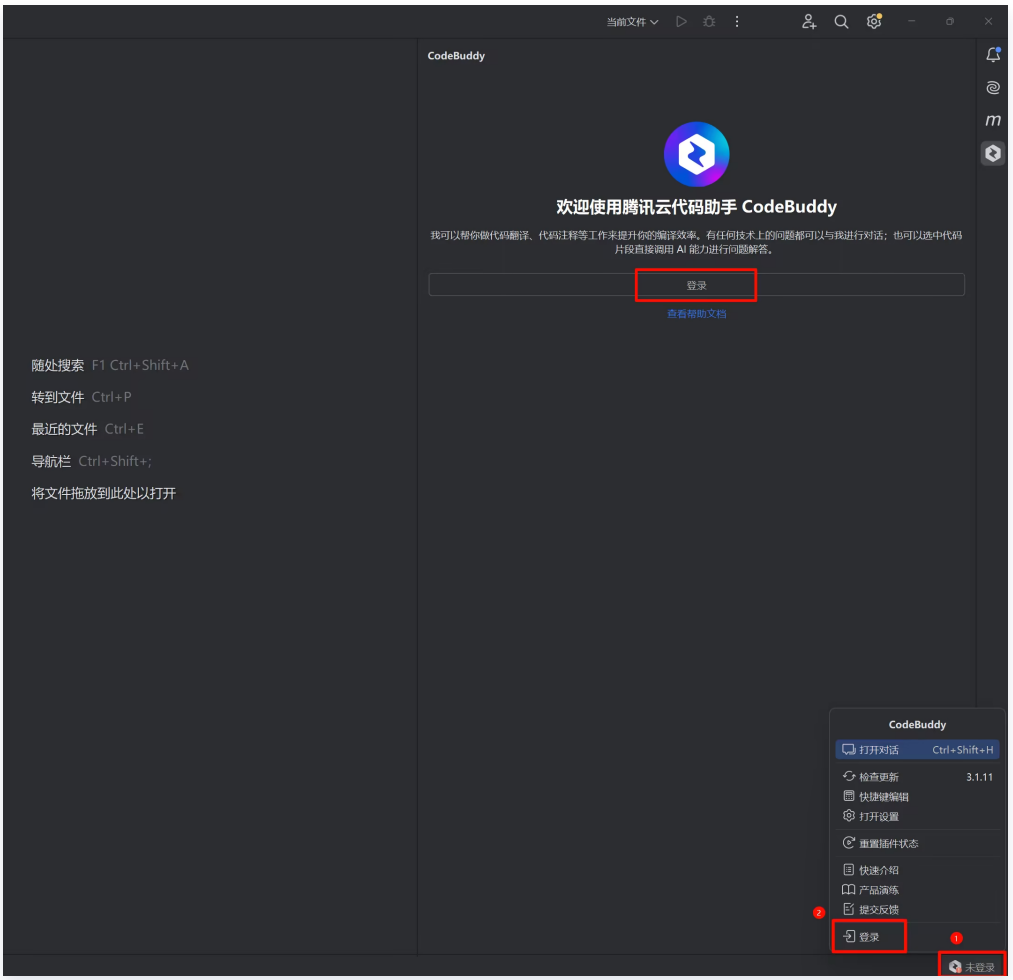
二、登录插件

1. 单击**底部 icon**，或在 CodeBuddy 插件登录页触发登录。

VS Code



JetBrains IDEs



2. 可以使用手机号或手机微信扫码，授权成功即可登录完成。



企业版

SaaS 企业版（旗舰版）登录

一、安装通用插件

通过 IDE 插件市场下载腾讯云代码助手，详情请参见 插件安装 。

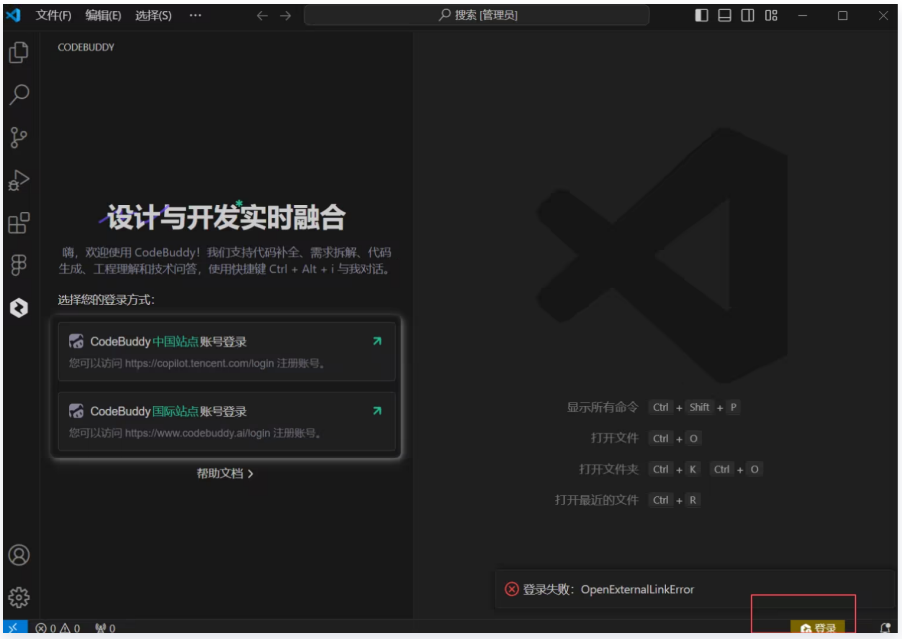
二、加入企业

联系所在企业的管理员完成成员添加，管理员如何添加成员详情请详见 旗舰版快速入门 。

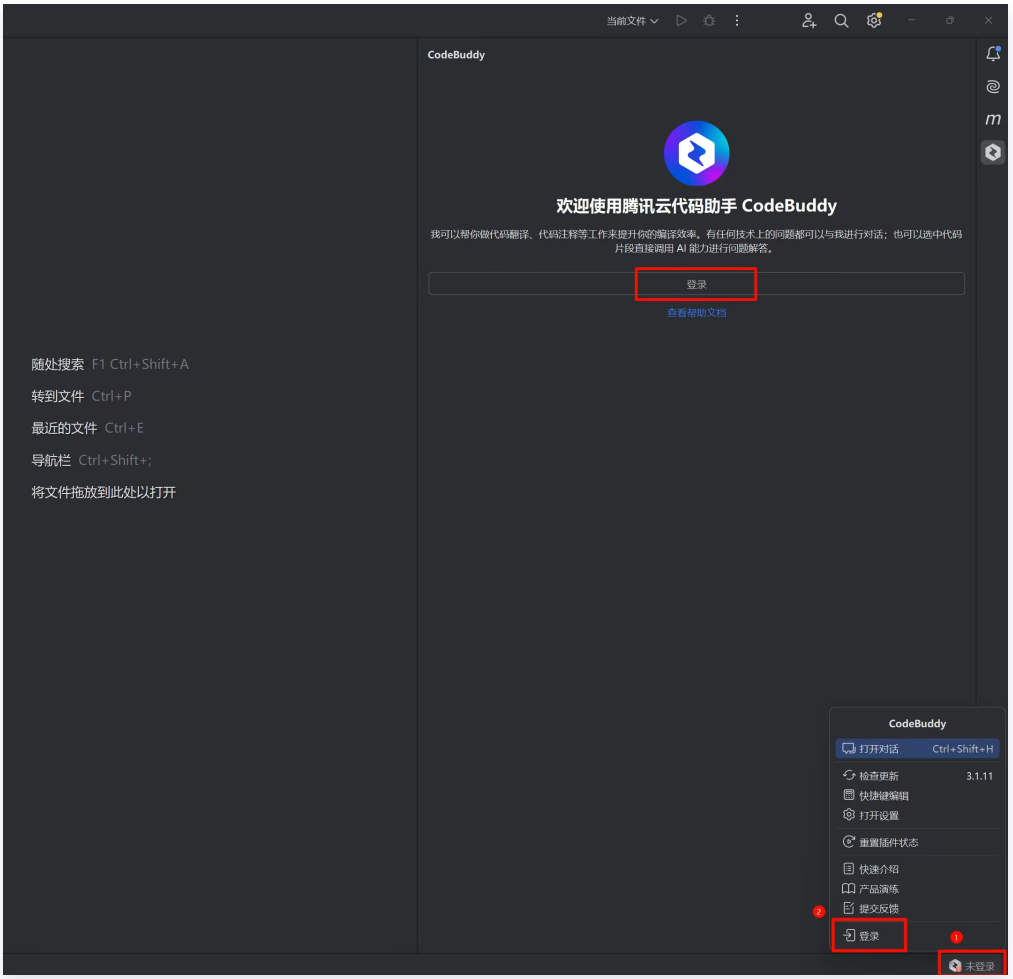
三、登录插件

1. 单击**底部 icon**，或在 CodeBuddy 插件登录页触发登录。

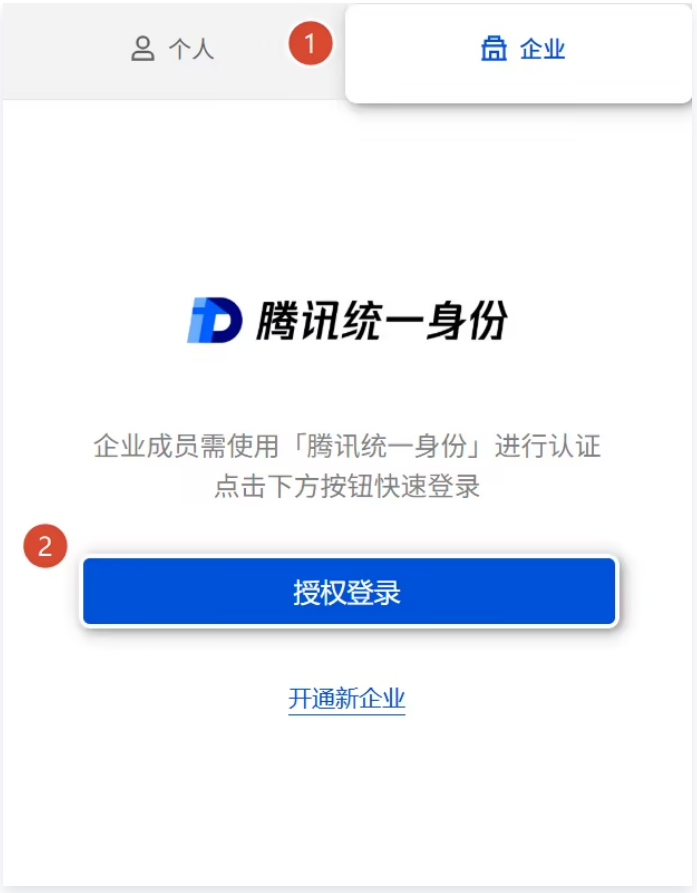
VS Code



JetBrains IDEs



2. 选择**腾讯统一身份**登录。



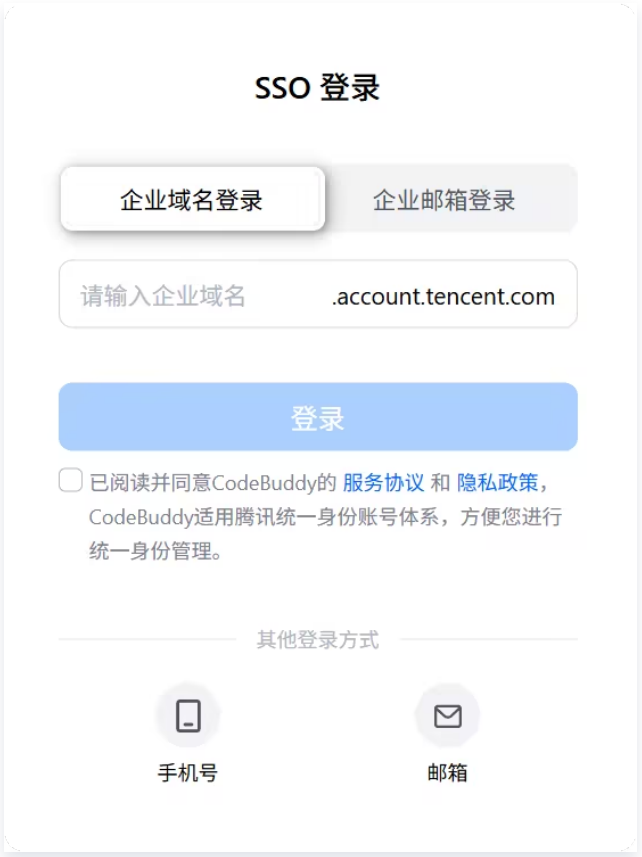
3. 选择企业在**腾讯统一身份**中配置支持的登录方式。
   - **手机验证码登录：**输入在**腾讯统一身份**添加成员时填入的手机号，获取验证码登录。



- **邮箱登录：输入在腾讯统一身份添加成员时填入的邮箱地址，获取验证码登录。**

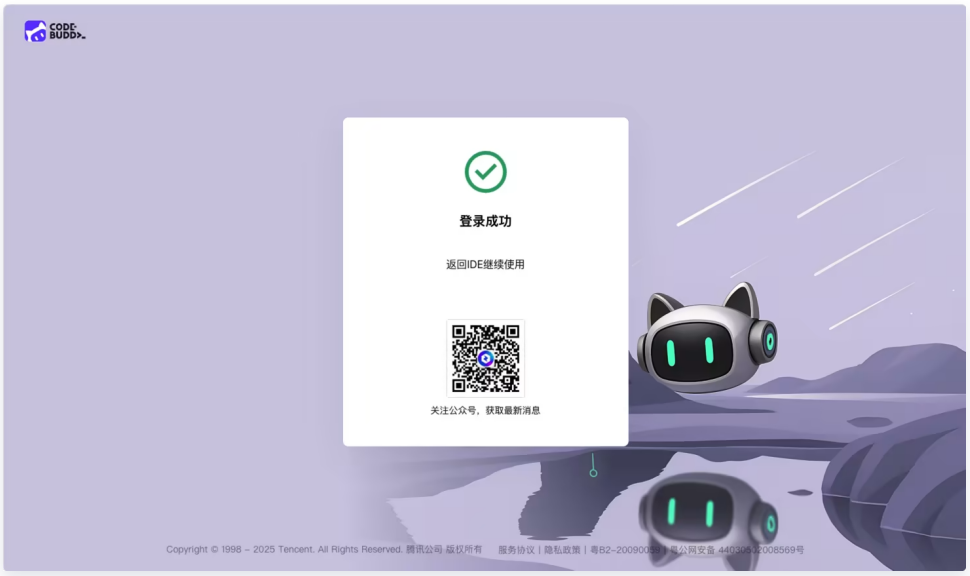


- **SSO 登录：输入企业域名，进入 SSO 登录页完成登录。**



> **说明：**
> - 如果企业仅开通了**企微认证源**，将直接跳转企业微信登录页面，使用企业微信**扫码登录**即可。
> - 如果同时开通了多个认证源，将跳转到腾讯统一身份 SSO 登录页面，使用用户名密码登录。

4. 登录成功后，将自动重定向回 CodeBuddy，获取登录状态。



专有云企业版（专享版）登录

一、安装专享版插件

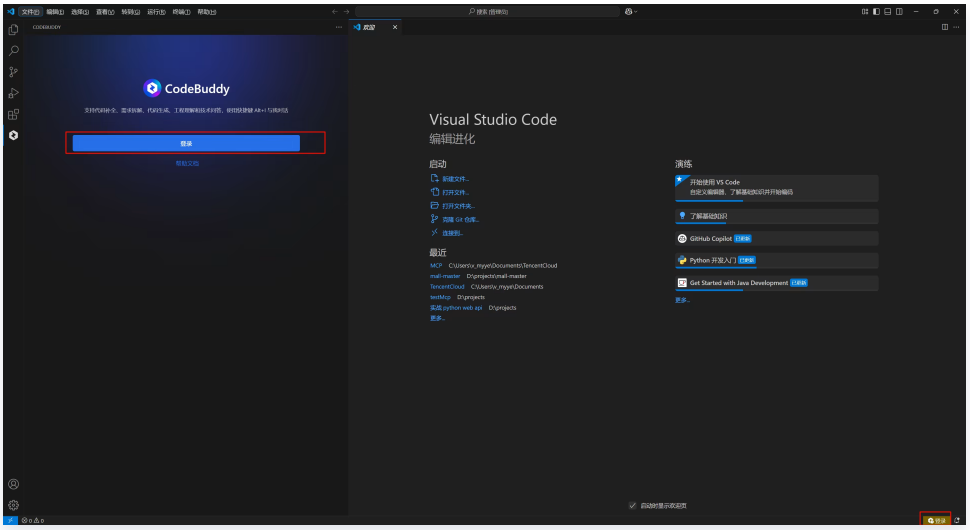
专享版插件请联系企业管理员获取，详情请参见 专享版快速入门 。

二、获取账号信息

从企业管理员处获得账号信息，包括账号、密码。

三、登录插件

1. 以 VSCode 为例，单击**底部 icon** 触发登录，单击**登录**。



2. 跳转浏览器登录页，输入账号密码。



3. 登录成功，即可返回 IDE 使用插件。

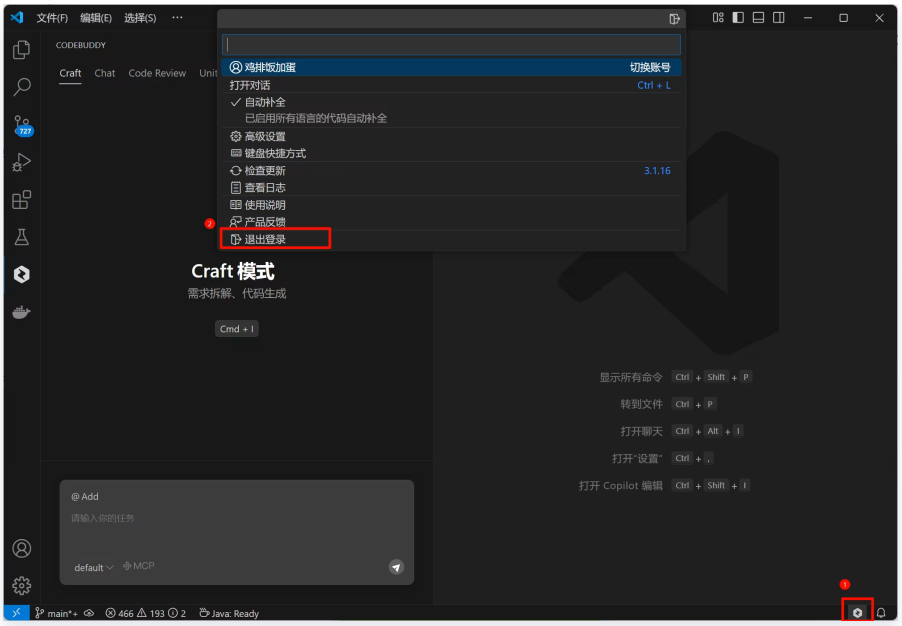


退出登录

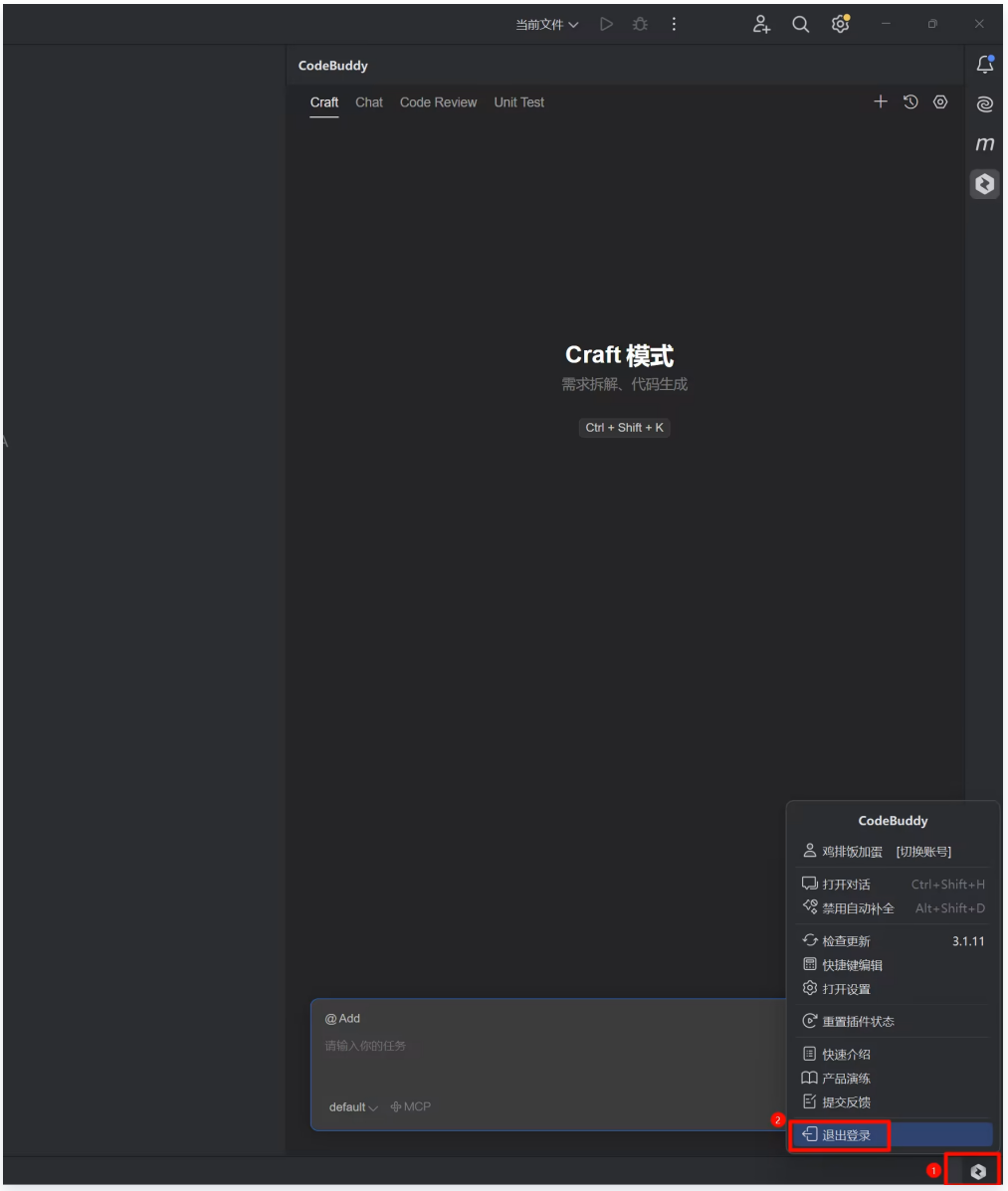
1. 单击**底部 icon** 触发退出登录。

VS Code

如下图所示：

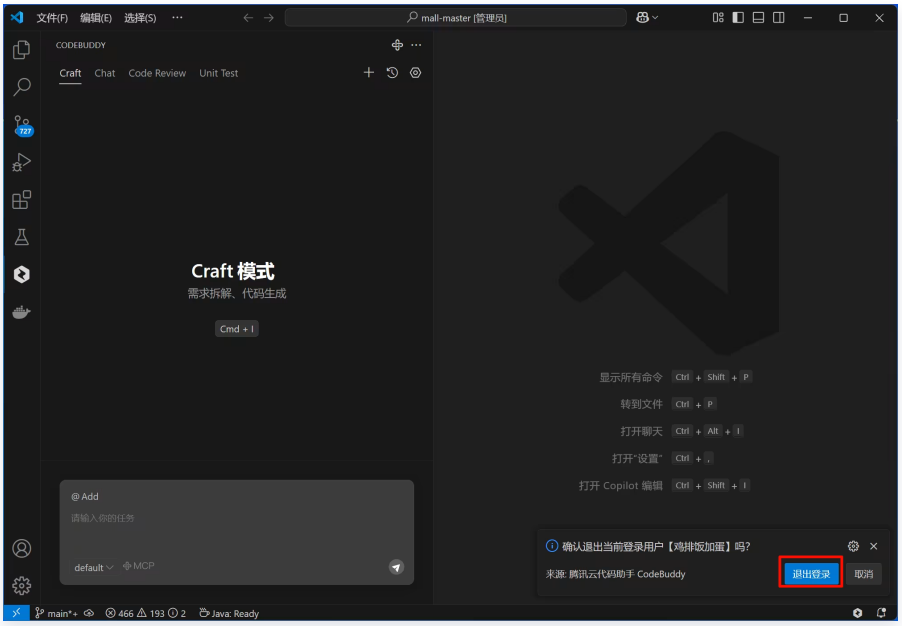


JetBrains IDEs

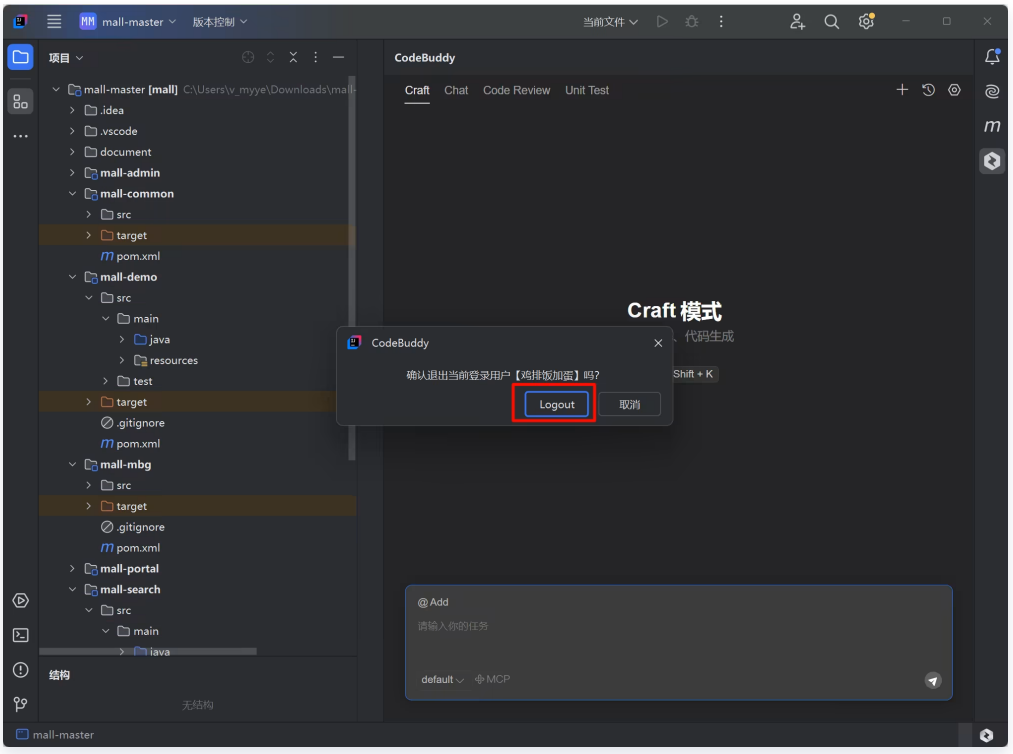


2. 单击**退出登录/Logout** 即可。

VS Code



JetBrains IDEs



功能入门

最近更新时间：2026-07-20 19:49:31

CodeBuddy 插件

VS Code

腾讯云代码助手主要提供两类功能：技术对话功能和代码补全功能。

技术对话功能

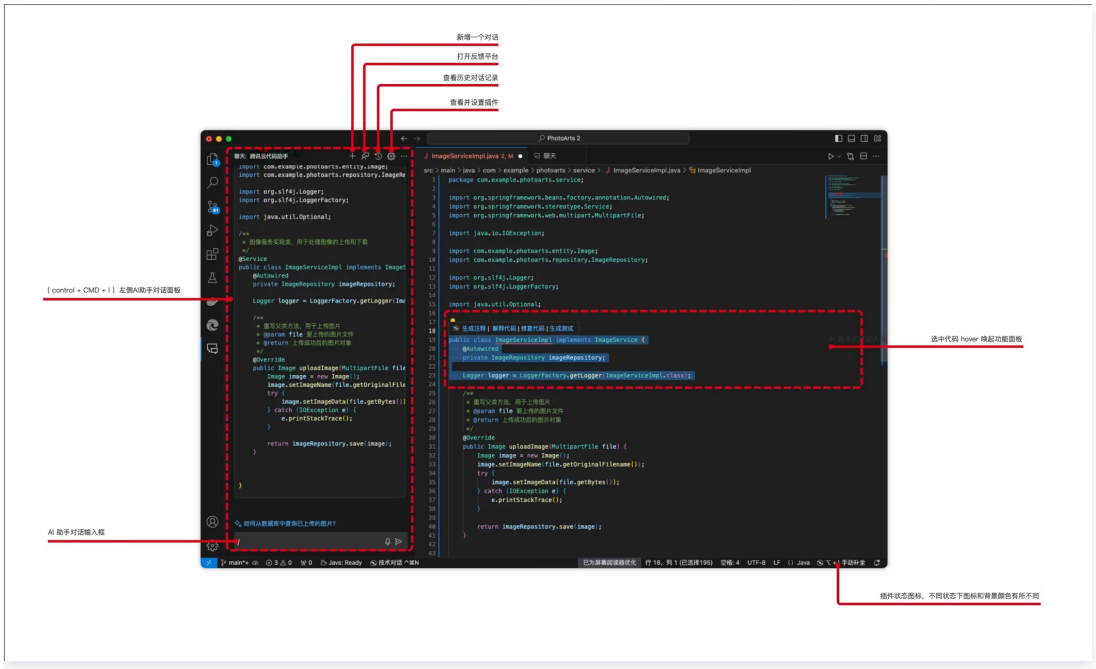
代码助手对话面板包括：左侧对话面板和编辑器内功能对话面板。

- 左侧对话面板。

  和传统的聊天软件类似，但是开发者只能与代码助手进行对话，并且只能咨询有关编程技术的问题。答案不一定有代码，可能是代码 + 文字说明等各种各样的形式。操作相对灵活自由，开发者自由选择。
- 编辑器内功能对话面板。

  编辑器内功能对话面板是选中代码后进行交互，要求代码助手对选中的代码做处理，处理的方式可以通过自然语言交互。对选中的代码处理后的结果，是明确要求开发者选择是接受，还是取消。

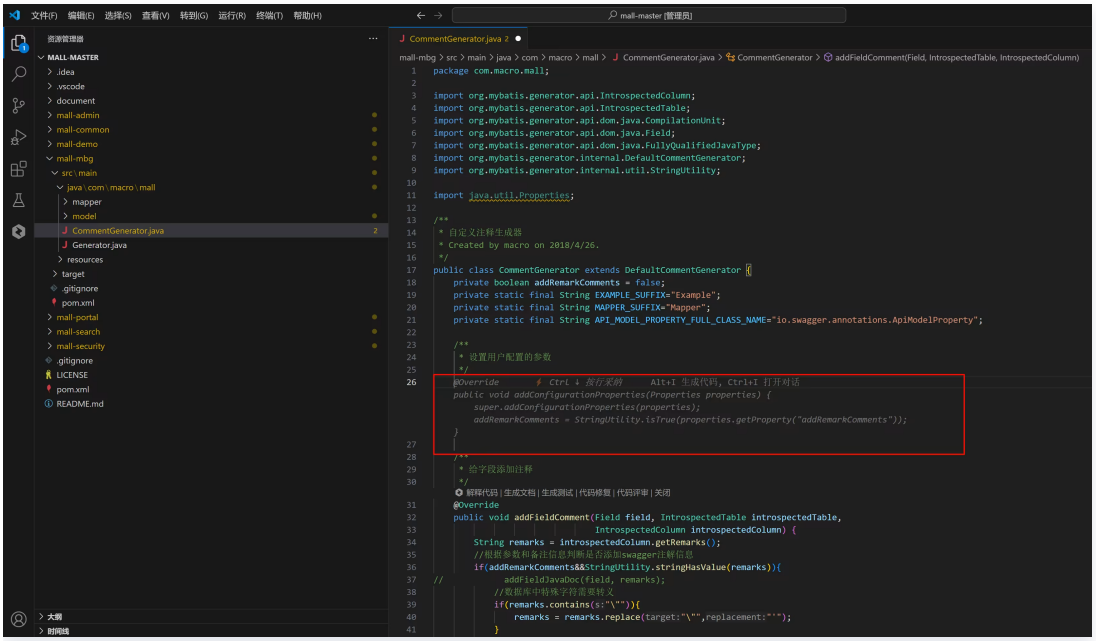
面板的概览图如下所示：



代码补全功能

目前支持的主流开发语言的补全，例如：C、C++、C#、CSS、Go、HTML、Java、JavaScript、Kotlin、TypeScript、React、Python、SQL、PHP、Objective-C、Shell、SystemC、Verilog、MATLAB、Markdown 等。

只需要使用编辑器打开代码文件，编辑代码文件时，代码助手会在合适时机和位置自动触发智能代码补全提示，如下图所示：



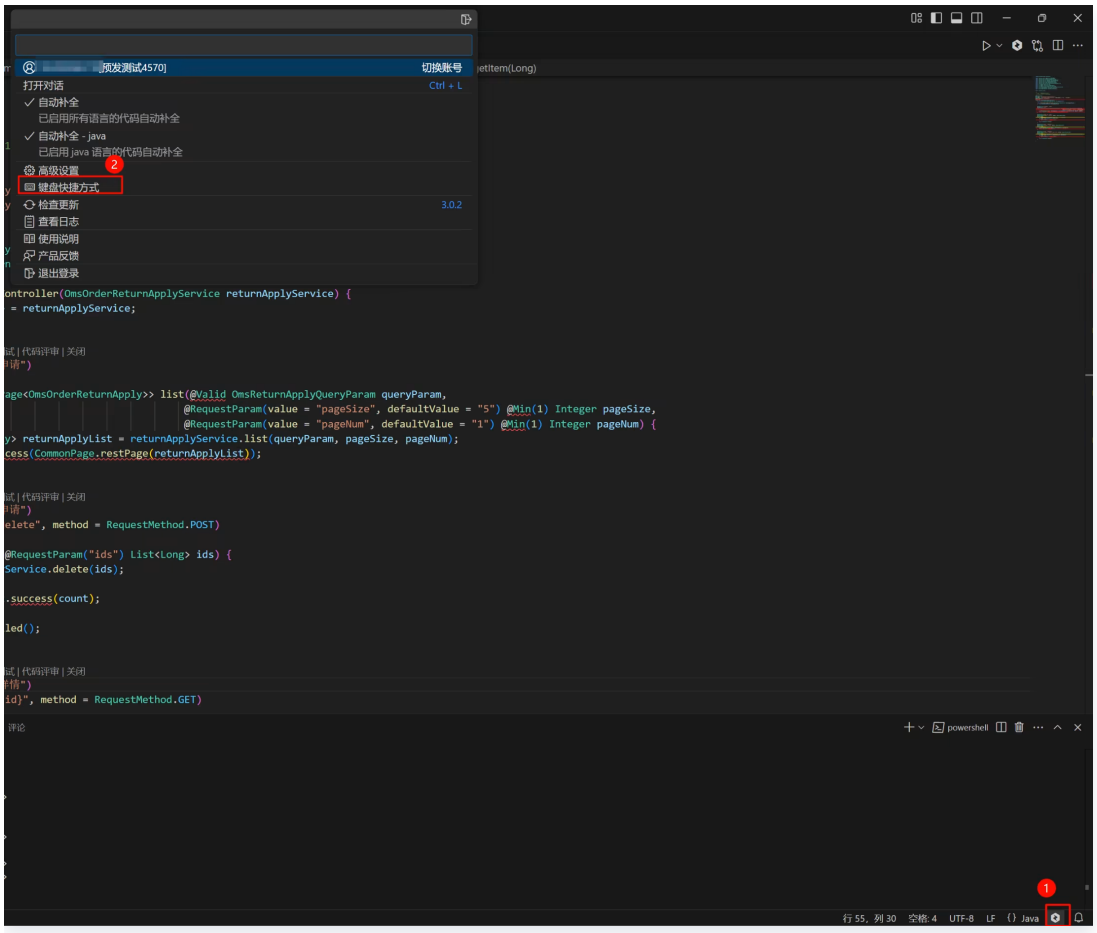
快捷键

默认快捷键示例如下：

操作	macOS	Windows
进行代码解释	⌥ + ⇧ + X	Shift + Alt + X
进行代码修复	⌥ + ⇧ + Y	Shift + Alt + Y
进行代码注释	⌥ + ⇧ + M	Shift + Alt + M
生成测试单元	⌥ + ⇧ + T	Shift + Alt + T
询问 AI 问题	⌘ + ^ + I	Ctrl + Windows + I
在代码区域打开技术对话	⌘ + ^ + N	Ctrl + Windows + N

> **说明：**
> 实际快捷键配置请在应用内查看。

支持按下图操作，进行快捷键配置：单击**插件菜单**，选择**键盘快捷方式**，即可对快捷键进行自定义。



对话快捷指令

在对话输入框中，输入 / 或 @ 调用预置的快捷指令：

- /clear：清空当前会话。
- /comments：为所选的代码添加文档注释。
- /new-notebook：创建一个新的 Jupyter 笔记本。
- /explain：解释所选代码的工作原理。
- /fix：针对所选代码中的问题提出修复方案。
- /tests：为所选代码生成单元测试。
- /name-var：变量命名。
- /cr：为所选代码和本地提交的 Diff 提出评审方案。
- /help：查看使用指南。
- @vscode：询问 VS Code。
- @terminal：询问如何在终端中执行某些操作。
- @workspace：询问您的工作空间，将自动引用当前代码。

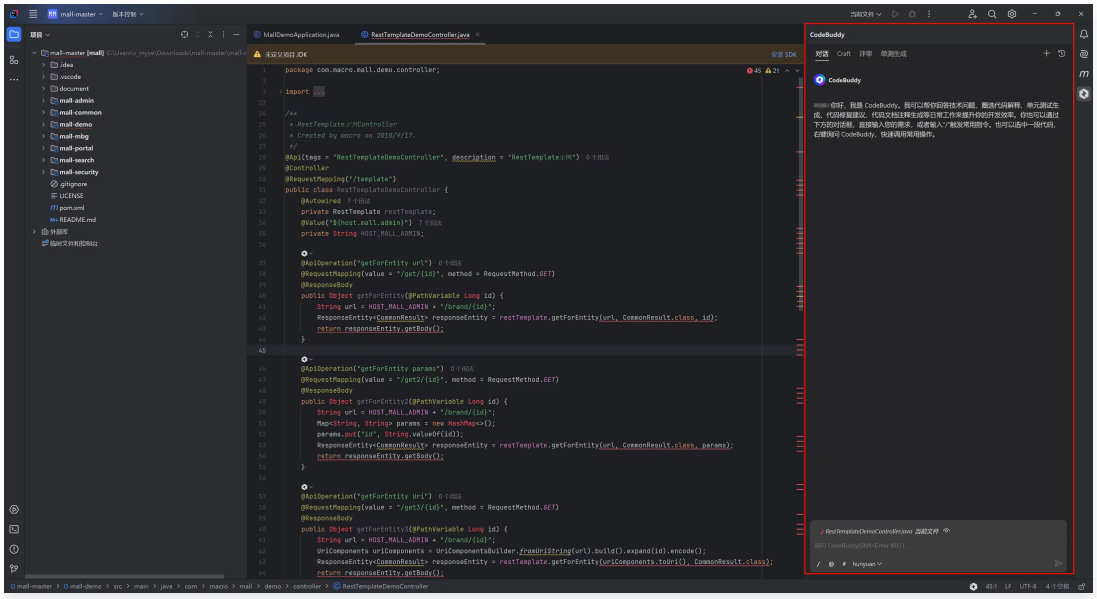
Visual Studio

具体详情请参见Visual Studio-使用指引 。

IDEA

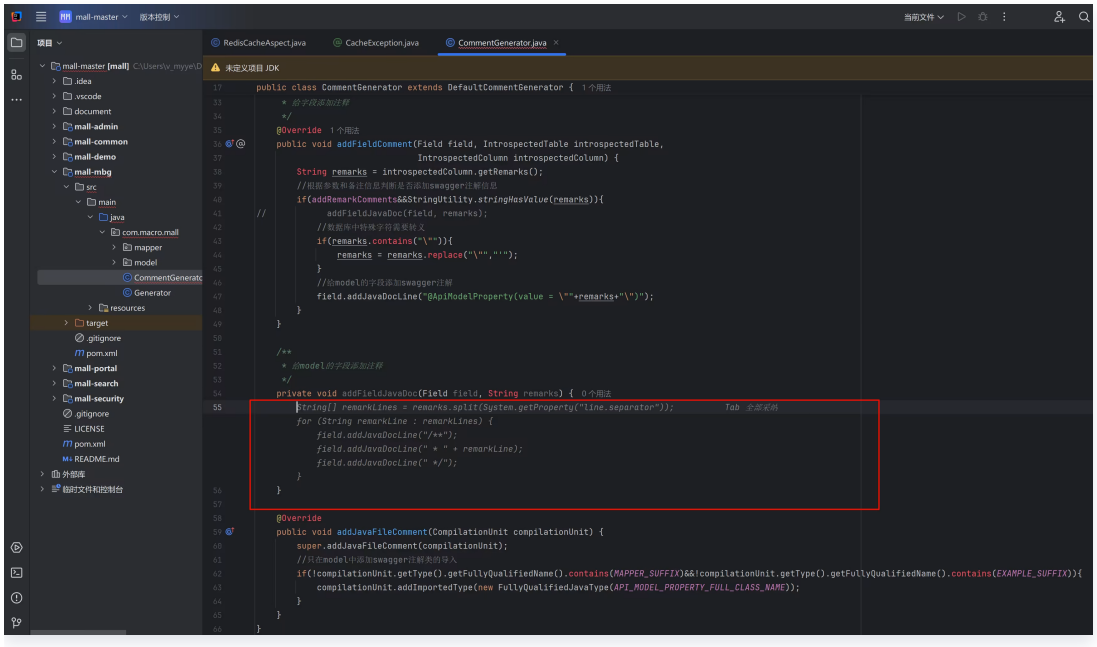
技术对话功能

代码助手对话面板与传统的聊天软件类似，只是开发者只能与代码助手进行对话，并且只能咨询一些有关编程技术的问题。对话面板适合咨询一些比较宽泛的问题，一些复杂的问题，需要开发者多回合沟通才能得到最终答案，并且答案的形式为纯文字或者代码 + 文字说明等形式。如下所示：



代码补全功能

目前支持的主流开发语言的补全，例如：C、C++、CSS、Go、HTML、Java、JavaScript、Kotlin、TypeScript、React、Python、SQL、Markdown 等。只需要使用编辑器打开代码文件，编辑代码文件时，代码助手会在合适时机和位置自动触发智能代码补全提示，如下图所示：



快捷键

示例快捷键如下：

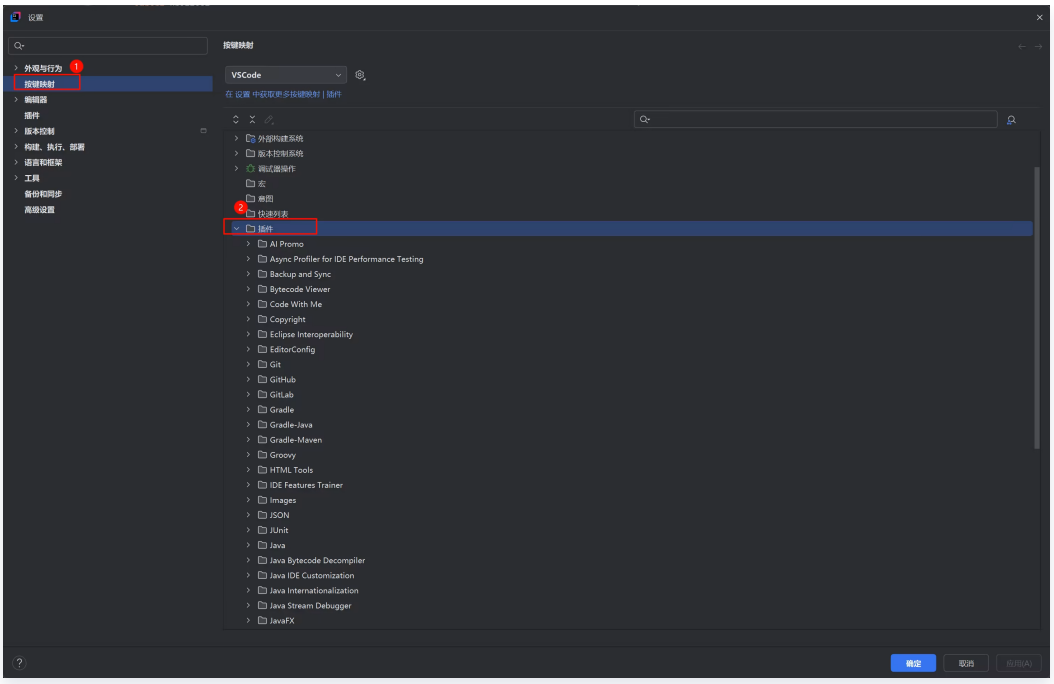
操作	macOS	Windows
触发代码补全	Enter	Enter
采纳所有推荐代码	Tab	Tab
按行采纳推荐代码	Ctrl + ⌘ + →	Ctrl + Alt + →
按词采纳推荐代码	⌘ + →	Ctrl + →
切换至下一个推荐结果	⌥ +]	Alt +]
切换至上一个推荐结果	⌥ + [	Alt + [
手动触发推荐	⌥ + \	Alt + \
撤销当前推荐状态	Esc	Esc

> **说明：**
> 实际快捷键配置请在应用内查看。

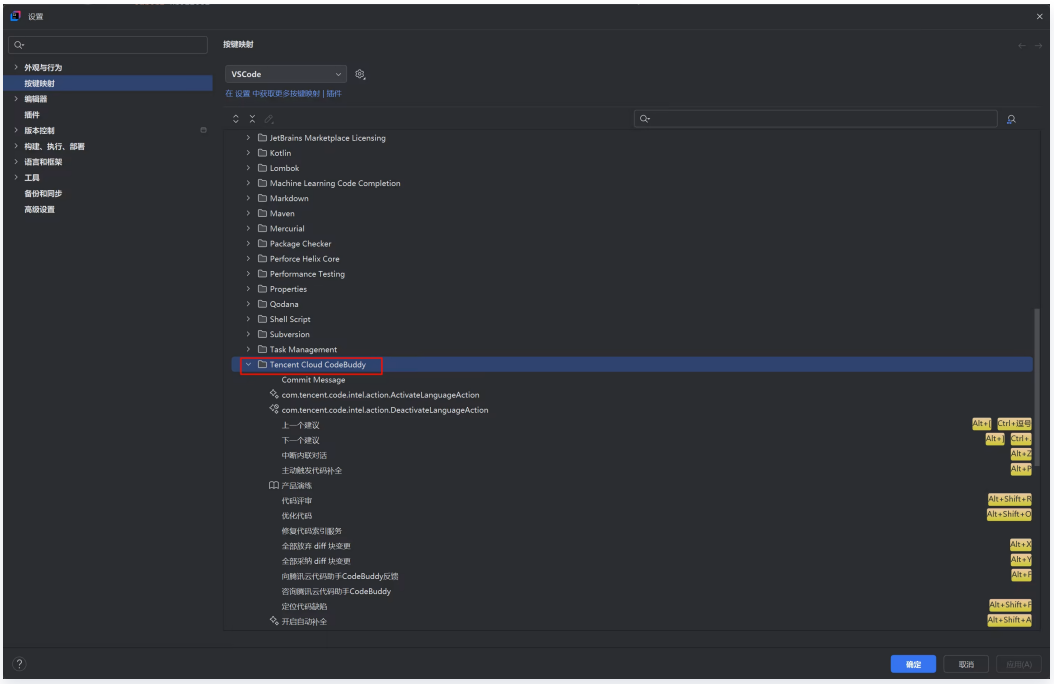
1. 支持按下图操作，进行快捷键配置：单击**插件菜单**，选择**快捷键编辑**。



2. 在弹出的设置弹窗中，选择**按键映射**，并在右侧找到“插件”文件夹。



3. 在“插件”文件夹中，找到“Tencent Cloud CodeBuddy”，即可对快捷键进行自定义。



对话快捷指令

在对话输入框中，输入 / 或 @ 调用预置的快捷指令：

- /comments：为所选的代码添加文档注释。
- /explain：解释所选代码的工作原理。
- /fix：针对所选代码中的问题提出修复方案。
- /tests：为所选代码生成单元测试。
- /clear：清空当前会话。

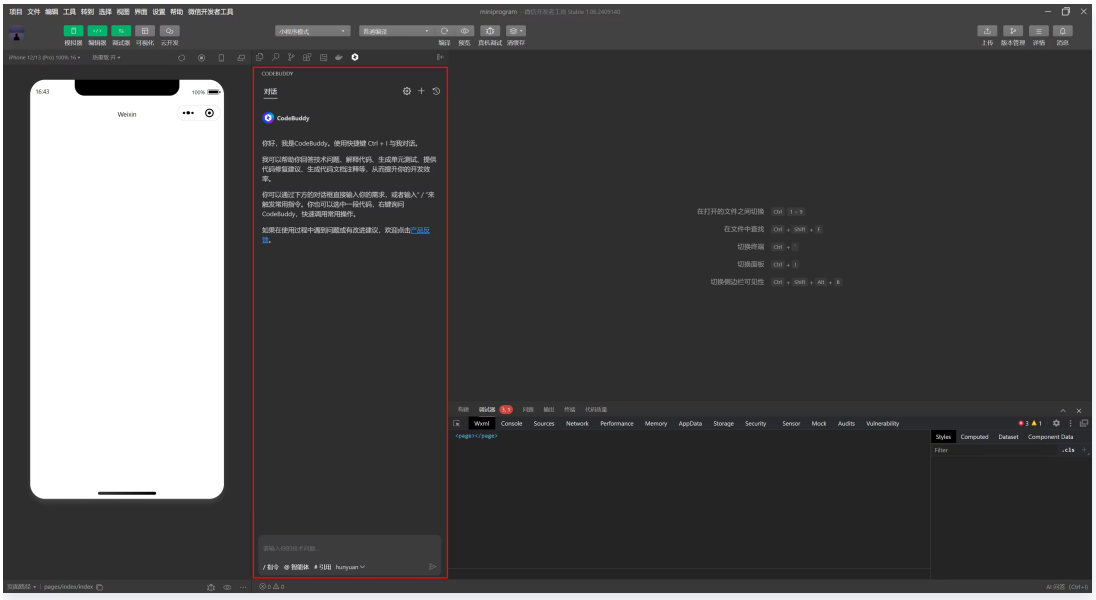
- /cr：AI 代码评审。
- /help：使用指南。
- @workspace：询问您的工作空间，将自动引用当前代码。
- @terminal：询问如何在终端中执行某些操作。

微信开发者平台 IDE

目前，腾讯云代码助手在微信开发者平台中主要提供技术对话功能，包括左侧对话面板和编辑器内功能对话面板。

左侧对话面板

在左侧对话面板中，您可以在对话框中输入编程技术相关的问题，进行技术对话，当生成的结果不满意时，可以进行多轮对话，一步步生成您想要的结果。

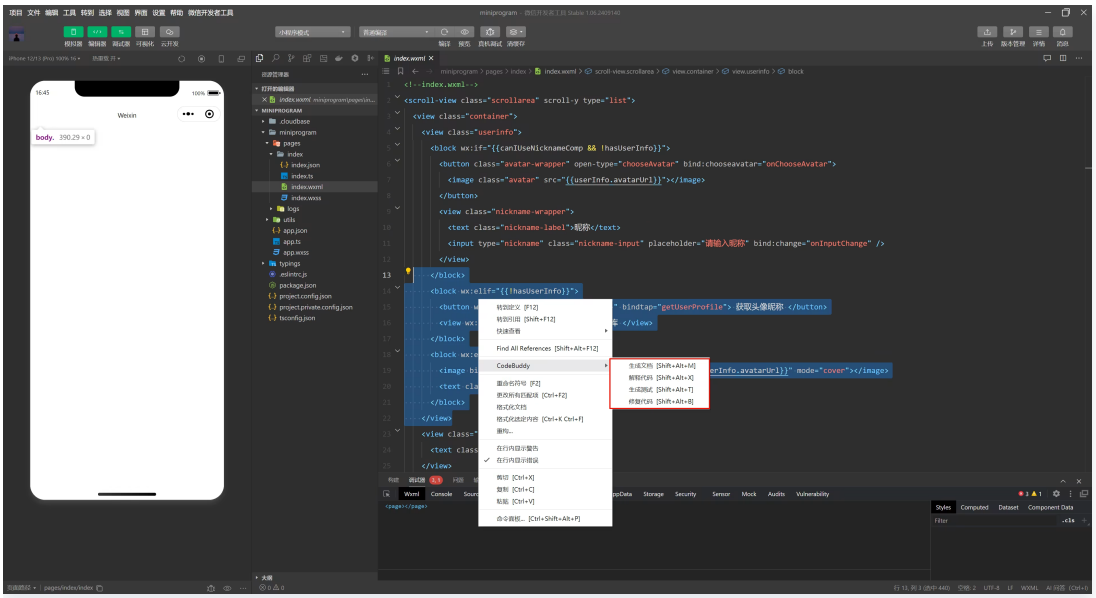


同时，目前代码助手也提供了如下指令，帮助您进行快捷对话：

- /clear：清空当前会话。
- /comments：为所选代码添加文档注释。
- /explain：解释所选代码的工作原理。
- /fix：提出针对所选代码问题的修复方案。
- /tests：为所选代码生成单元测试。
- /help：查看使用指南。
- @workspace：自动引用当前工作空间代码。
- @vscode：询问 VS Code。
- @terminal：询问如何在终端中执行某些操作。

编辑器内功能对话面板

编辑器内功能对话面板是选中代码后进行对话，目前暂时提供生成注释文档、解释代码、生成测试、修复代码功能。



CodeBuddy IDE

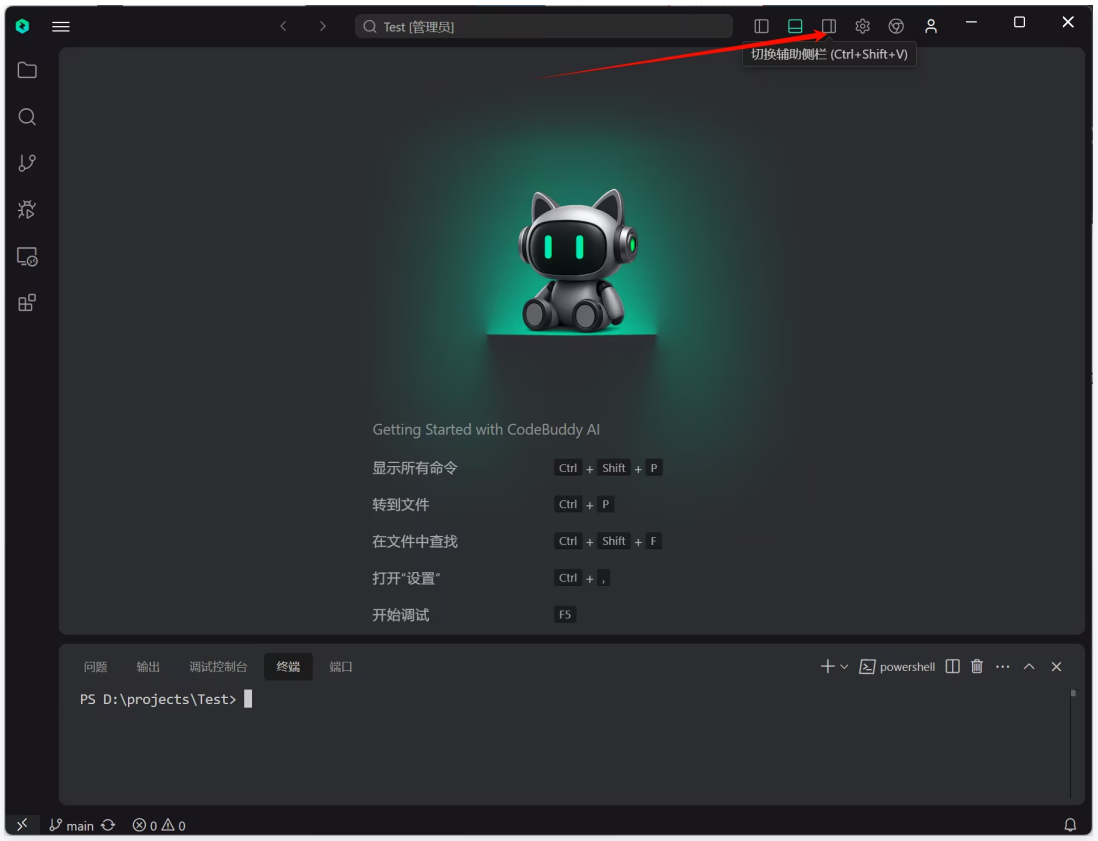
在 CodeBuddy IDE 中，同样可以分为两大类功能，即对话功能和代码补全 Tab 功能。

对话功能

对话功能，即通过自然语言与 Agent 进行对话，Agent 将根据您选择的对话模式（Chat/Craft）帮您提供解决方案，或自动帮您完成复杂编码任务。对话功能提供两种交互方式：侧栏对话和编辑器内的内联对话。

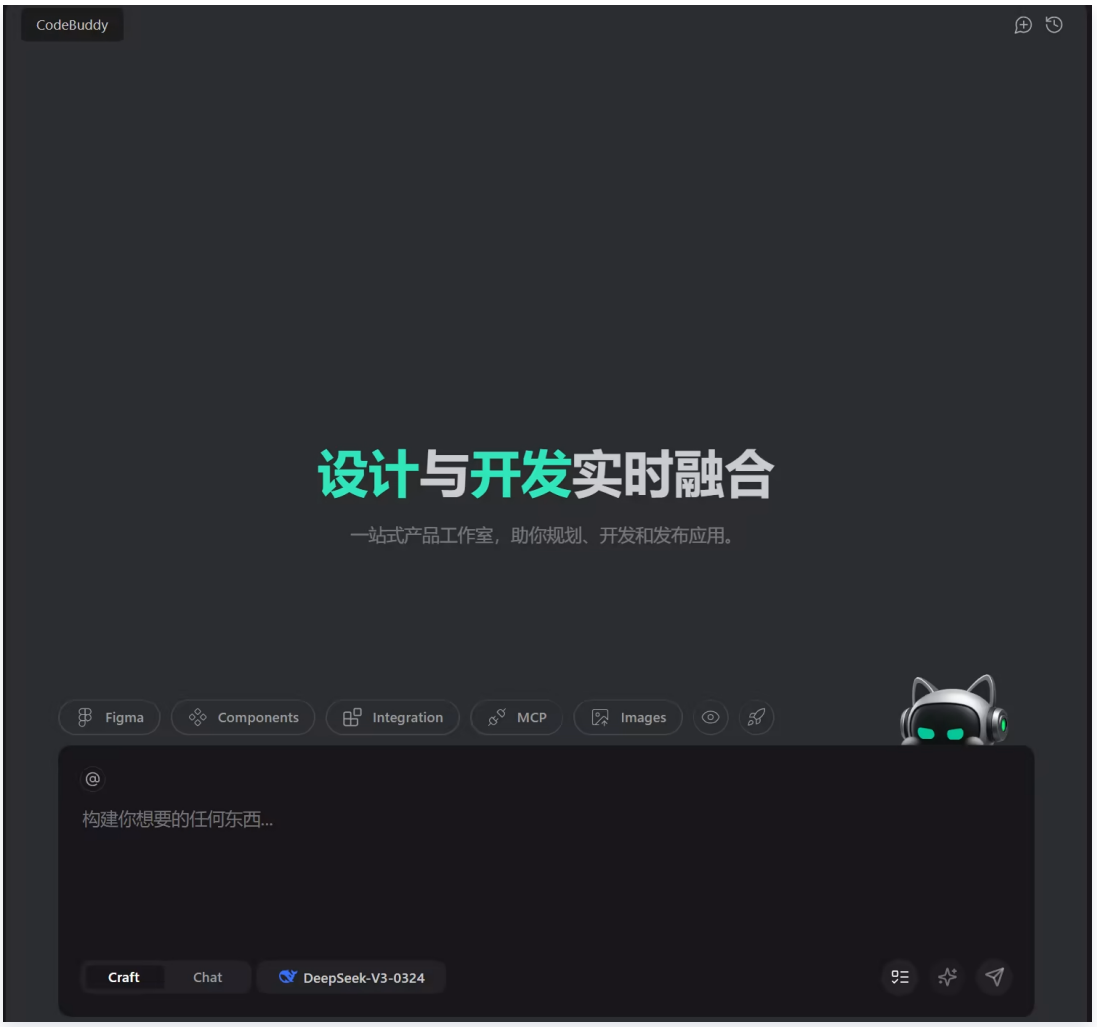
侧栏对话

1. 打开侧栏对话可以通过以下两种方式：
    - 点击右上方的**切换辅助侧栏**按钮，如下图所示：



- 直接通过快捷键打开（Windows：**Ctrl + Shift + V**，Mac：⌘ **Command** + ⇧ **Shift** + **V**）。

2. **在侧栏对话中，您可以根据实际需求使用不同的模式与 Agent 进行对话，即 Craft 和 Chat 模式。**

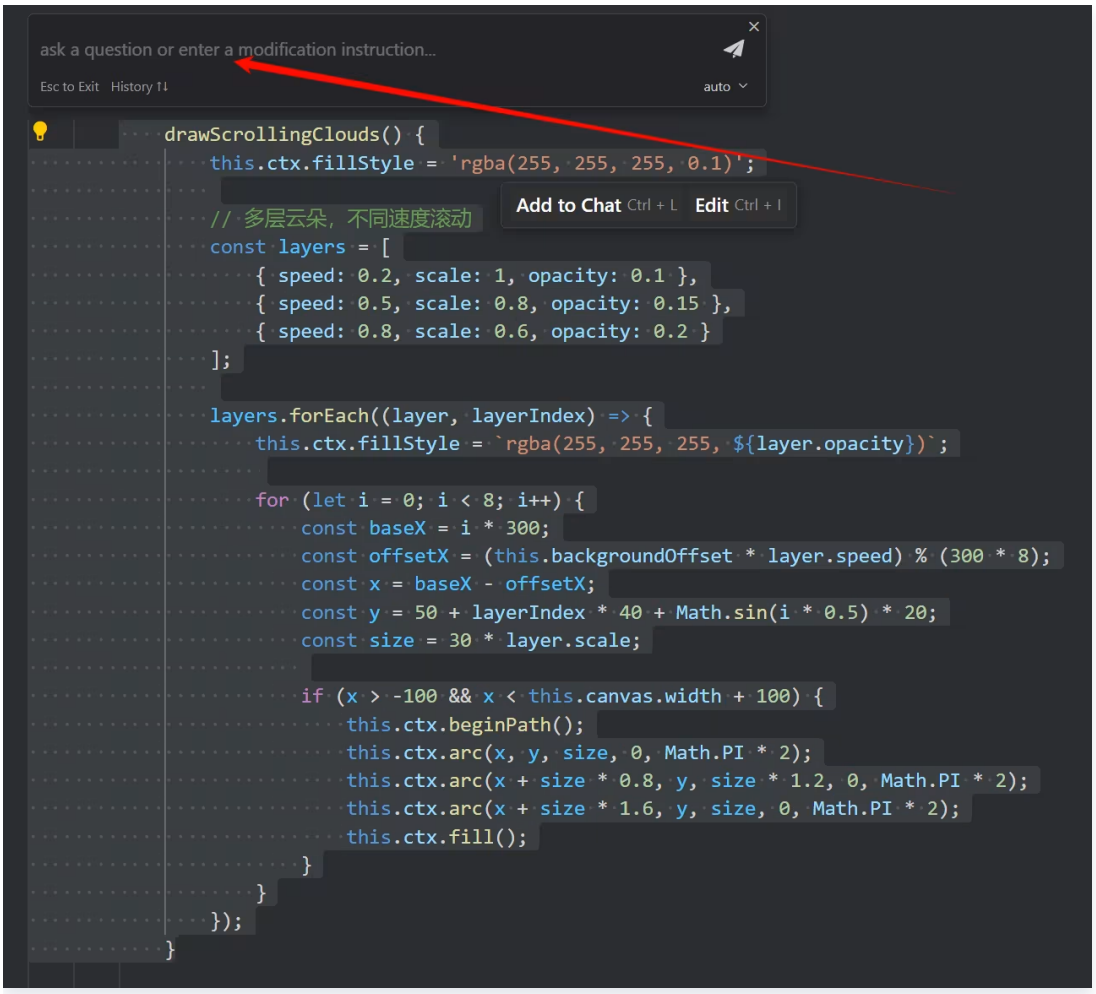


- **Chat 模式**：类似聊天机器人，通过自然语言进行技术对话，询问代码库问题或其它编程技术问题，支持多轮追问和上下文感知。
- **Craft 模式**：相比 Chat 模式， Craft 模式意味着它不再是一个对话机器人工具，而是一个具有 “独立思考” 和 “自主执行” 的编程伙伴，可以通过自然语言指令，AI 深度理解需求，独立自主完成多文件代码编写与修改。



内联对话

可以通过快捷键的方式打开内联对话，Windows：**Ctrl + I**，Mac：⌘ **Command + I**。在内联对话中，您可以圈选代码后，通过自然语言的方式进行对话，Agent 将帮您完成任务。



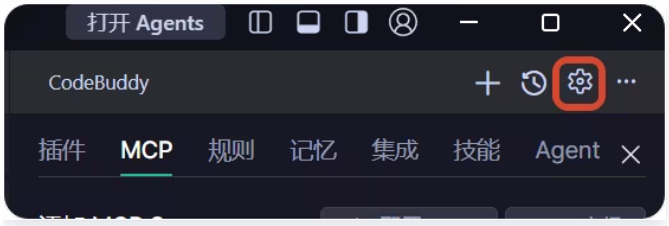
Tab

支持多行编辑的智能代码补全。基于当前代码内容和项目上下文分析，提供多行代码补全建议，并支持基于历史编辑记录的“下一步编辑建议”功能，可通过 Tab 键接受补全建议。

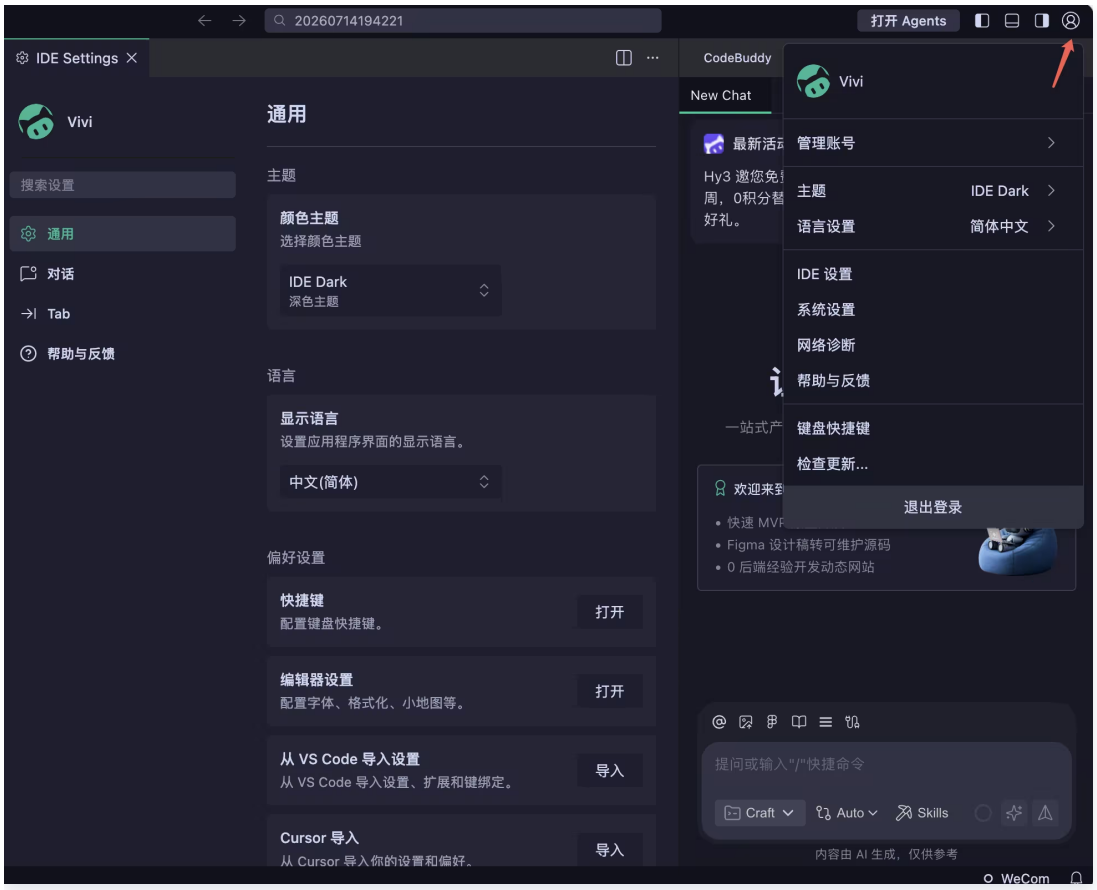
```
// 绘制滚动云朵
drawScrollingClouds() {
    this.ctx.fillStyle = 'rgba(255, 255, 255, 0.1)';

    // 多层云朵，不同速度滚动
    const layers = [
        { speed: 0.2, scale: 1, opacity: 0.1 },
        { speed: 0.5, scale: 0.8, opacity: 0.15 },
        { speed: 0.8, scale: 0.6, opacity: 0.2 }
    ];

    layers.forEach((layer, layerIndex) => {
        this.ctx.fillStyle = `rgba(255, 255, 255, ${layer.opacity})`;

        for (let i = 0; i < 8; i++) {
            const baseX = i * 300;
            const offsetX = (this.backgroundOffset * layer.speed) % (300 * 8);
            const x = baseX - offsetX;
            const y = 50 + layerIndex * 40 + Math.sin(i * 0.5) * 20;
            const size = 30 * layer.scale;

            if (x > -100 && x < this.canvas.width + 100) {
                this.ctx.beginPath();
                this.ctx.arc(x, y, size, 0, Math.PI * 2);
                this.ctx.fill();
            }
        }
    });
}
```

设置及快捷键

在 CodeBuddy IDE 中，可以单击右上方的**设置**按钮打开设置页进行设置，在设置页中，分为**插件、MCP、规则、记忆、集成、技能、Agent** 这7个分页，供您选择设置。



在右上角头像处进入 **IDE 设置**中，可按如下位置查看/配置快捷键。


20260714194221
打开 Agents
IDE Settings
Vivi
搜索设置
通用
对话
Tab
帮助与反馈
通用
主题
颜色主题
选择颜色主题
IDE Dark
深色主题
语言
显示语言
设置应用程序界面的显示语言。
中文(简体)
偏好设置
快捷键
配置键盘快捷键。
打开
编辑器设置
配置字体、格式化、小地图等。
打开
从 VS Code 导入设置
从 VS Code 导入设置、扩展和键绑定。
导入
Cursor 导入
从 Cursor 导入你的设置和偏好。
导入
CodeBuddy
New Chat
Vivi
管理账号
主题
IDE Dark
语言设置
简体中文
IDE 设置
系统设置
网络诊断
帮助与反馈
键盘快捷键
检查更新...
退出登录
• Figma 设计稿转可维护源码
• 0 后端经验开发动态网站
提问或输入"/"快捷命令
Craft
Auto
Skills
内容由 AI 生成，仅供参考
WeCom


搜索设置
通用
对话
Tab
帮助与反馈
主题
颜色主题
选择颜色主题
IDE Dark
深色主题
语言
显示语言
设置应用程序界面的显示语言。
中文(简体)
偏好设置
快捷键
配置键盘快捷键。
打开
编辑器设置
配置字体、格式化、小地图等。
打开
从 VS Code 导入设置
从 VS Code 导入设置、扩展和键绑定。
导入
Cursor 导入
从 Cursor 导入你的设置和偏好。
导入

操作指南
代码补全

最近更新时间：2026-07-20 19:49:31

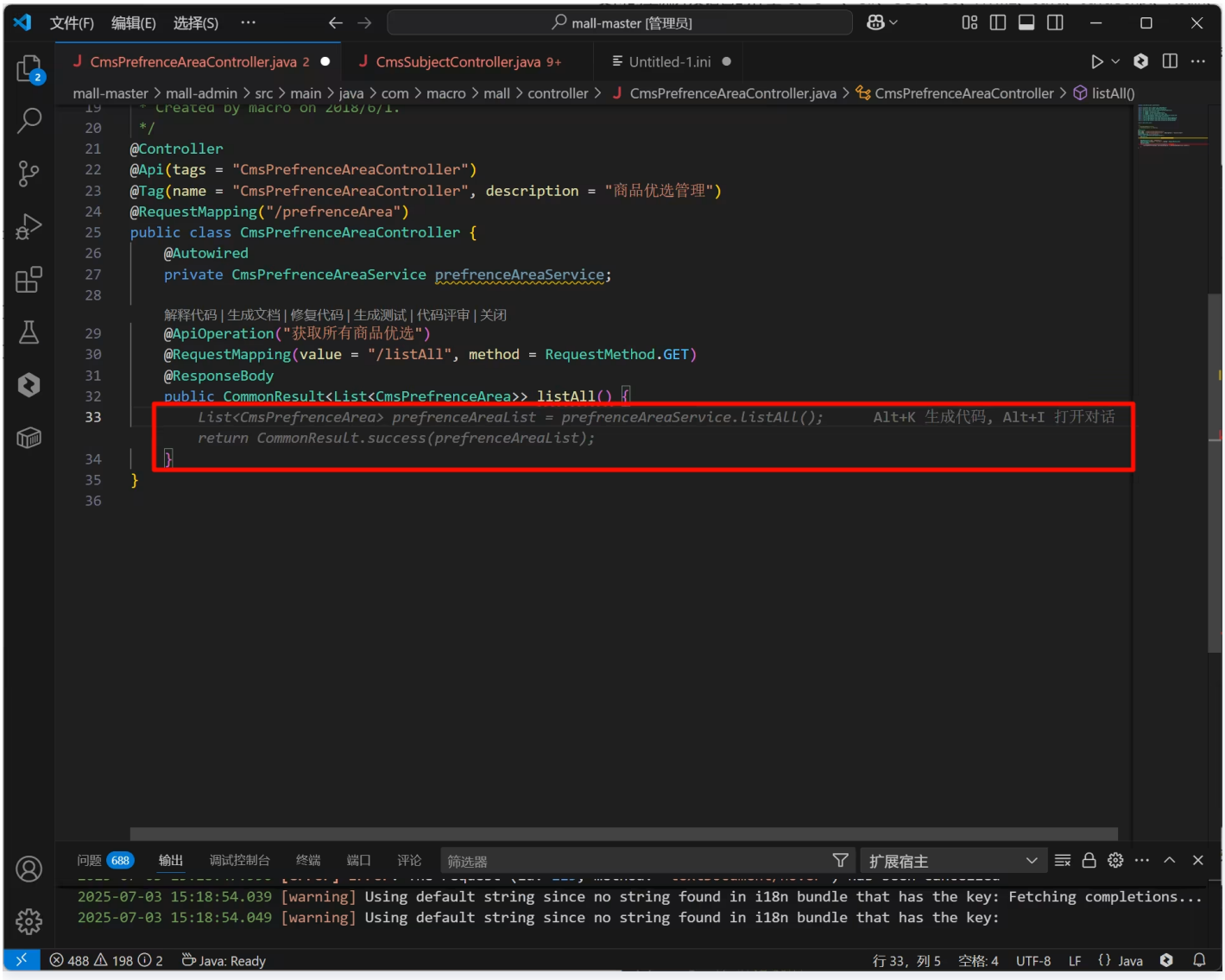
支持的主流开发语言的补全 C、C++、C#、CSS、Go、HTML、Java、JavaScript、Kotlin、TypeScript、React、Python、SQL、PHP、Objective-C、Shell、System-C、Verilog、MATLAB、Markdown 等。

只需要使用编辑器打开代码文件，在编辑代码文件时，代码助手会在合适时机和位置自动触发智能代码补全提示，而且会有代码补全推荐时的文字引导，可以按照文字引导进行操作，如下所示。

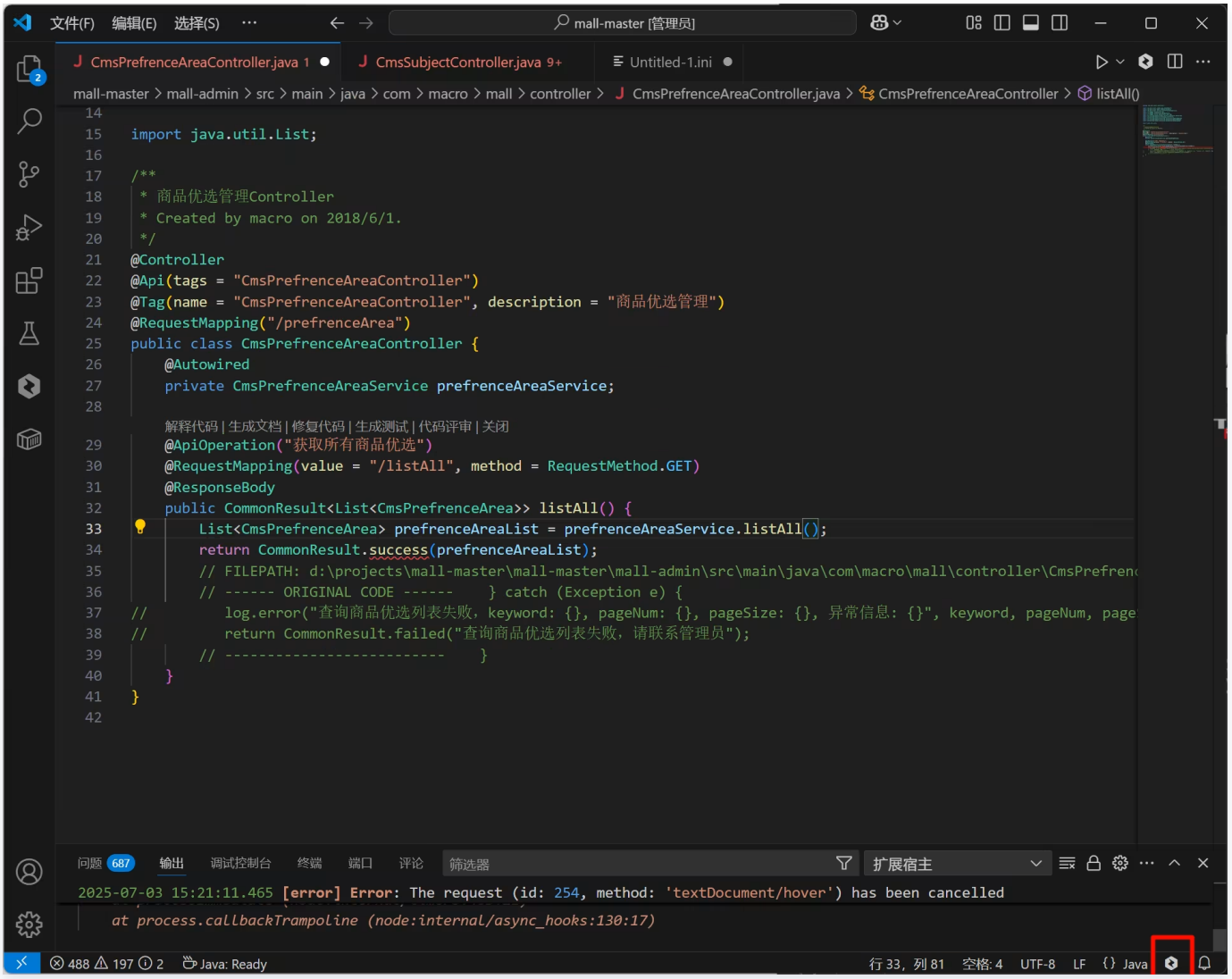
CodeBuddy 插件

安装指引参考：产品安装 。

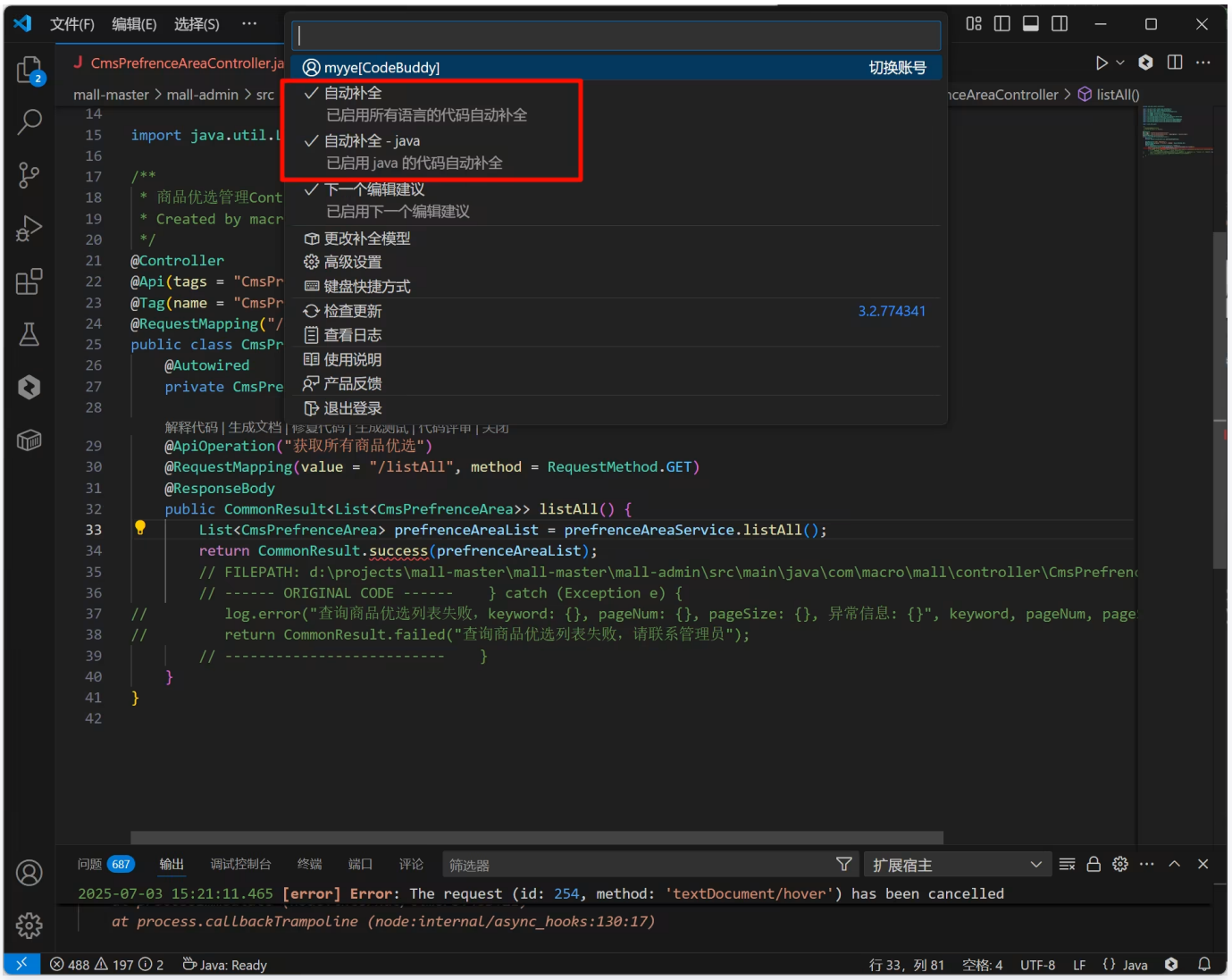
VS Code



- 禁用和启用补全功能：单击状态栏的插件小图标，可以启用或者禁用插件代码补全功能，如下图所示。



禁用的时候，可以全局禁用，也可以选择对特定的代码语言进行禁用，如下图所示。

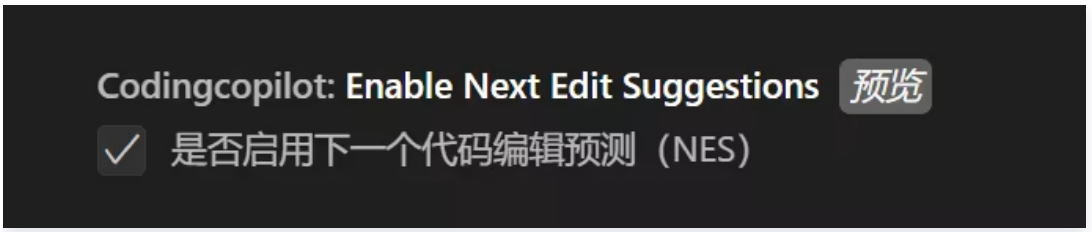


- **补全默认快捷键如下：**

操作	macOS	Windows
采纳所有推荐代码	Tab	Tab
按行采纳推荐代码	⌘ + ↓	Ctrl + ↓
按词采纳推荐代码	⌘ + →	Ctrl + →
手动触发推荐	⌥ + \	Alt + \
撤销当前推荐状态	Escape	Escape

补全 Pro 能力

补全 Pro 能力，不仅可以对当前光标处的代码进行补全，也可以对其它相关地方的代码进行补全，即预测开发者接下来要写的代码，并提前给出建议，具有“前瞻性”的核心特点，实现更好的补全效果。默认为开启状态，支持自动触发和手动触发，手动触发的快捷键为 Ctrl + Shift + Enter。



- **修改变量名的补全 Pro 场景**

  修改变量名后，根据光标的移动，以及代码情况，捕捉到上下文信息后，在合理的光标位置出现补全推荐。

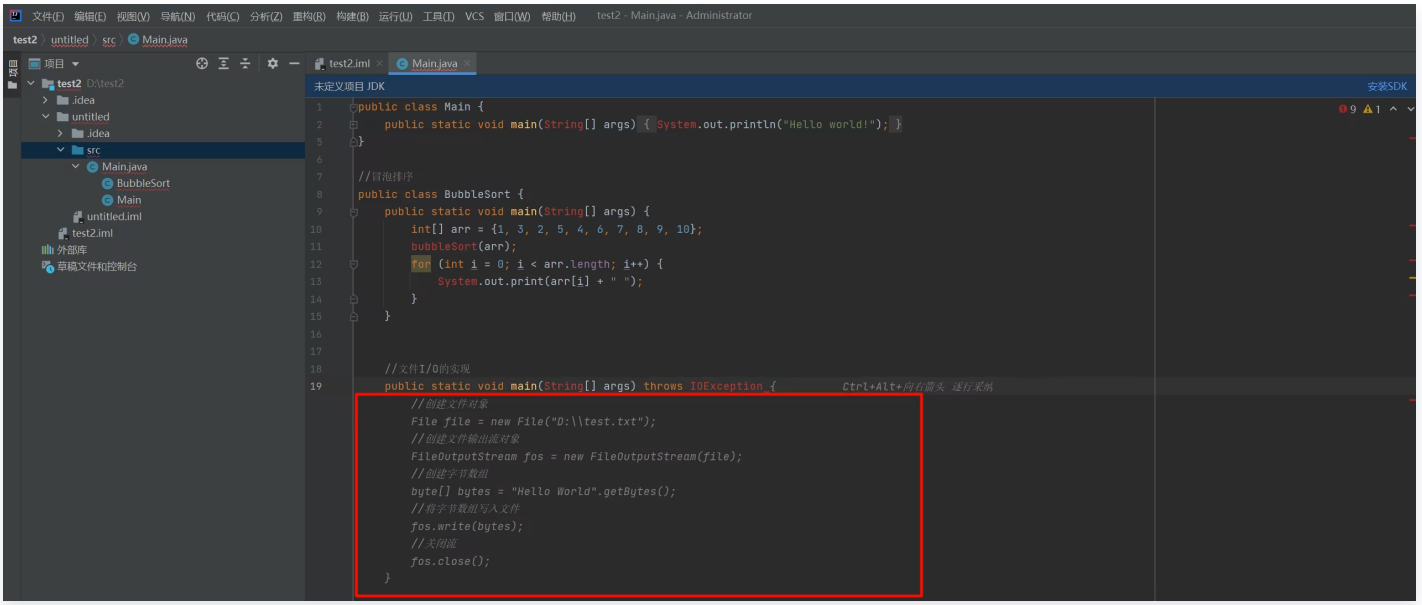
```python
# 示例：多个函数间的相互调用与变量的多次调用

# 定义全局变量
global_var = 10

# 函数1：计算两个数的和
解释代码 | 生成文档 | 修复代码 | 生成测试 | 代码评审 | 关闭
def add_numbers(a, b):
    return a + b

# 函数2：计算两个数的乘积
解释代码 | 生成文档 | 修复代码 | 生成测试 | 代码评审 | 关闭
def multiply_numbers(a, b):
    return a * b

# 函数3：调用函数1和函数2，并打印结果
解释代码 | 生成文档 | 修复代码 | 生成测试 | 代码评审 | 关闭
def calculate_and_print(x, y):
    aaaaaa = add_numbers(x, y)  # 调用函数1
    product_result = multiply_numbers(x, y)  # 调用函数2
    print(f"Sum of {x} and {y} is: {sum_resultaaaaaa}")  → Tab
    print(f"Product of {x} and {y} is: {product_result}")
    return sum_result, product_result

# 函数4：修改全局变量并调用其他函数
解释代码 | 生成文档 | 修复代码 | 生成测试 | 代码评审 | 关闭
def modify_global_and_calculate():
    global global_var
    global_var += 5  # 修改全局变量
    print(f"Updated global_var: {global_var}")
    result = calculate_and_print(global_var, 3)  # 调用函数3
    return result

# 主函数
解释代码 | 生成文档 | 修复代码 | 生成测试 | 代码评审 | 关闭
def main():
    # 调用函数3
    sum_result, product_result = calculate_and_print(5, 7)
    print(f"Results from calculate_and_print: Sum={sum_result}, Product={product_result}")

    # 调用函数4
    final_result = modify_global_and_calculate()
    print(f"Final results from modify_global_and_calculate: {final_result}")

# 执行主函数
if __name__ == "__main__":
    main()
```

- **修改函数或方法名的补全 Pro 场景**

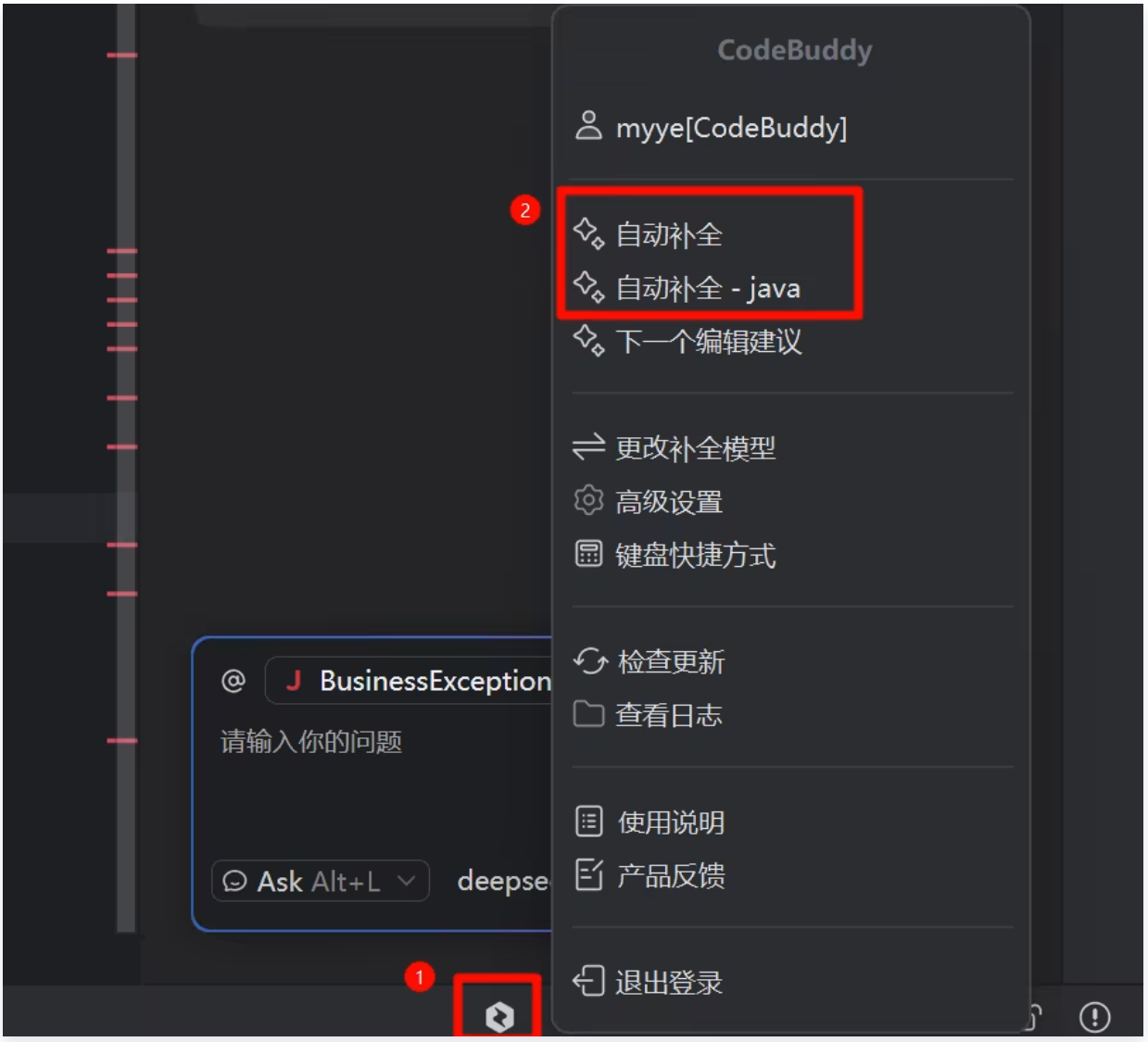
  函数或方法名修改后，在引用处，通过光标的定位，自动触发补全 Pro 。

```python
# 定义全局变量
global_var = 10

# 函数1：计算两个数的和
解释代码 | 生成文档 | 修复代码 | 生成测试 | 代码评审 | 关闭
def abdcd(a, b):
    return a + b

# 函数2：计算两个数的乘积
解释代码 | 生成文档 | 修复代码 | 生成测试 | 代码评审 | 关闭
def multiply_numbers(a, b):
    return a * b

# 函数3：调用函数1和函数2，并打印结果
解释代码 | 生成文档 | 修复代码 | 生成测试 | 代码评审 | 关闭
def calculate_and_print(x, y):
    sum_result = add_numbersabdcd(x, y)  # 调用函数1  → Tab
    product_result = multiply_numbers(x, y)  # 调用函数2
    print(f"Sum of {x} and {y} is: {sum_result}")
    print(f"Product of {x} and {y} is: {product_result}")
    return sum_result, product_result

# 函数4：修改全局变量并调用其他函数
解释代码 | 生成文档 | 修复代码 | 生成测试 | 代码评审 | 关闭
def modify_global_and_calculate():
    global global_var
    global_var += 5  # 修改全局变量
    print(f"Updated global_var: {global_var}")
    result = calculate_and_print(global_var, 3)  # 调用函数3
    return result

# 主函数
解释代码 | 生成文档 | 修复代码 | 生成测试 | 代码评审 | 关闭
def main():
    # 调用函数3
    sum_result, product_result = calculate_and_print(5, 7)
    print(f"Results from calculate_and_print: Sum={sum_result}, Product={product_result}")

    # 调用函数4
    final_result = modify_global_and_calculate()
    print(f"Final results from modify_global_and_calculate: {final_result}")

# 执行主函数
if __name__ == "__main__":
    main()
```

JetBrains IDE



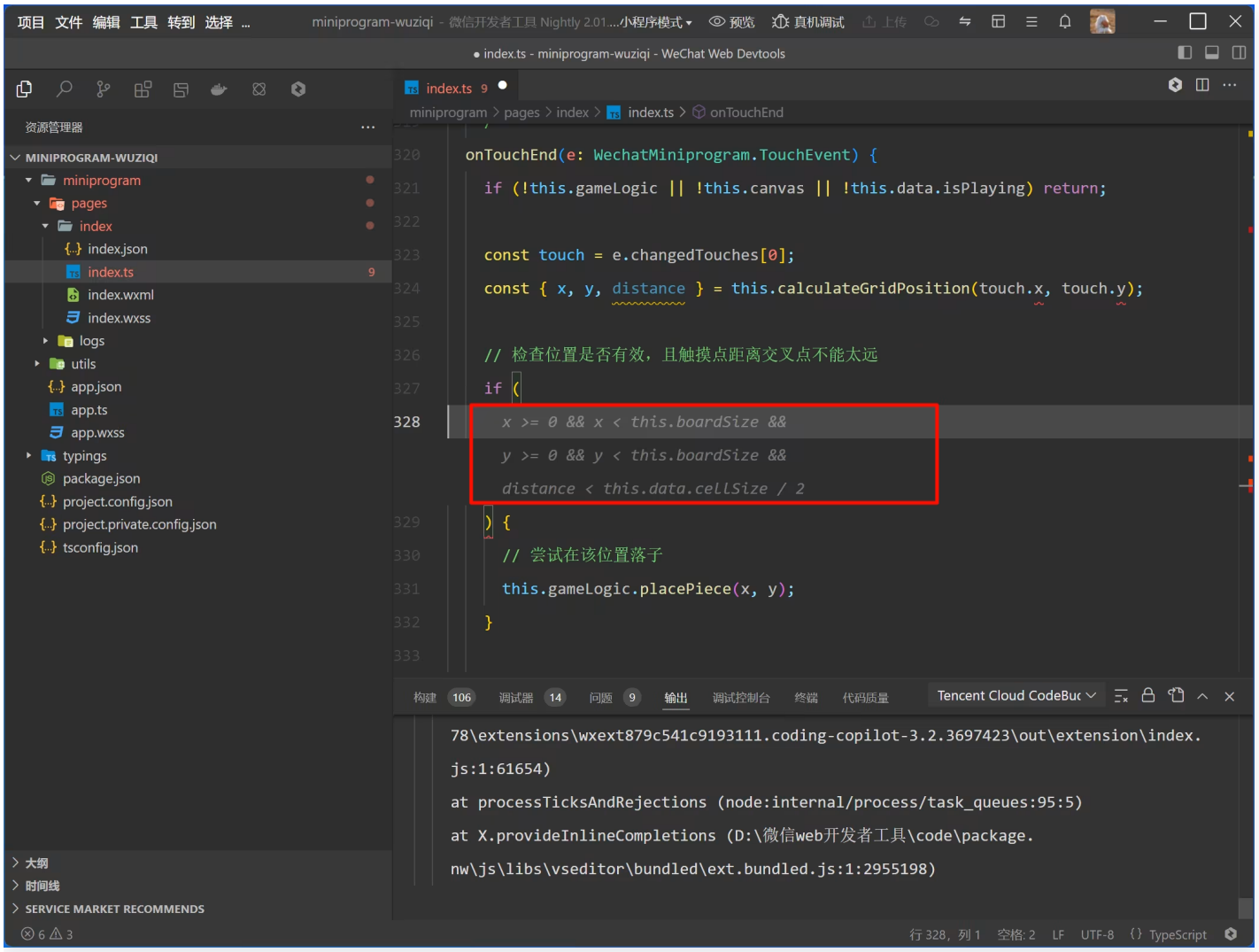
- 禁用和启用补全功能：单击状态栏的插件小图标，可以启用或者禁用插件代码自动补全功能，如下图所示。



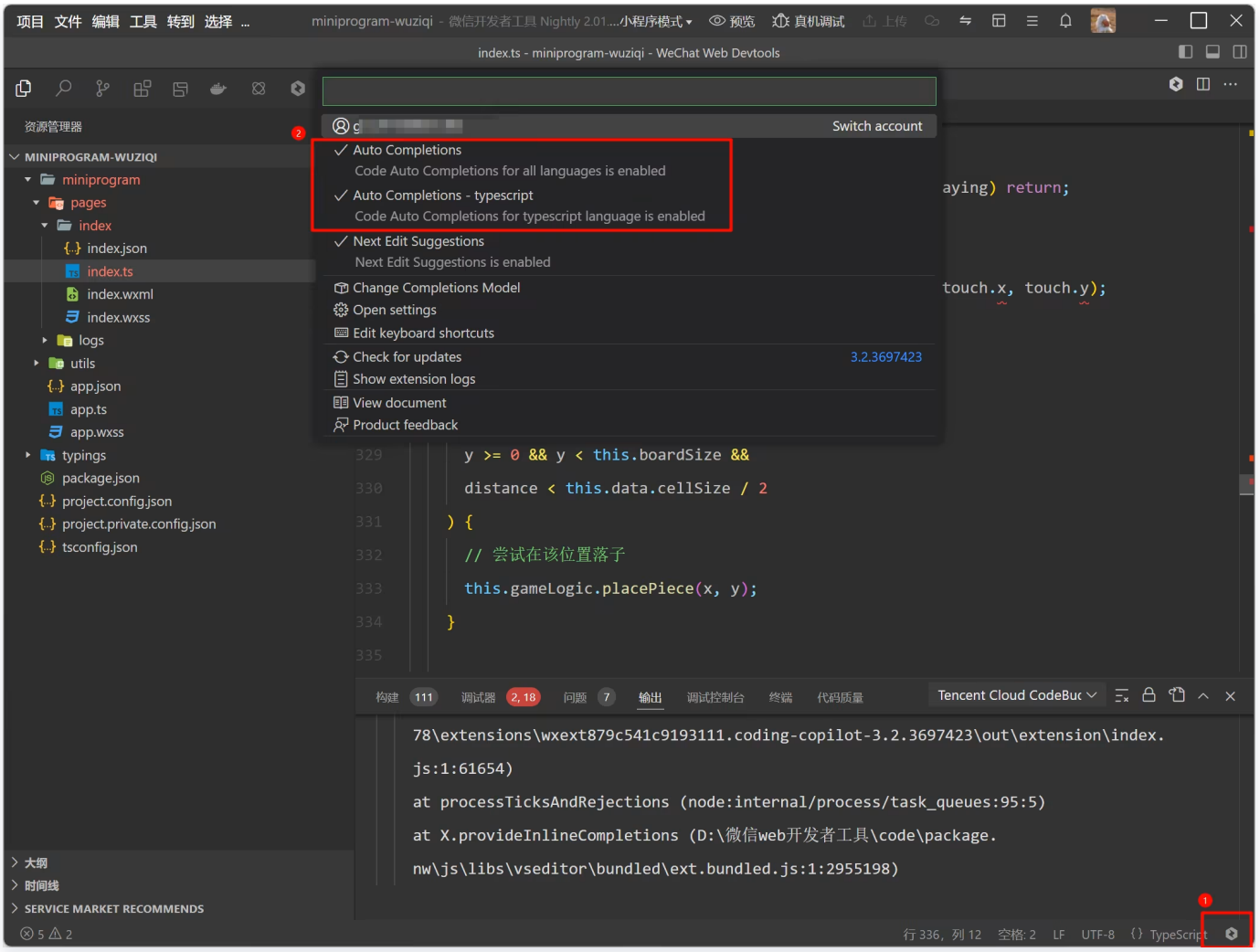
- 默认快捷键如下：

操作	macOS	Windows
触发代码补全	Enter	Enter
采纳所有推荐代码	Tab	Tab
切换至下一个推荐结果	⌥ +]	Alt +]
切换至上一个推荐结果	⌥ + [	Alt + [
手动触发推荐	暂无	Alt + P
撤销当前推荐状态	Esc	Esc

微信开发者工具



- **禁用和启用智能补全功能：单击状态栏的插件小图标，在弹出的菜单选项里可以启用或者禁用插件代码补全功能，如下图所示。**

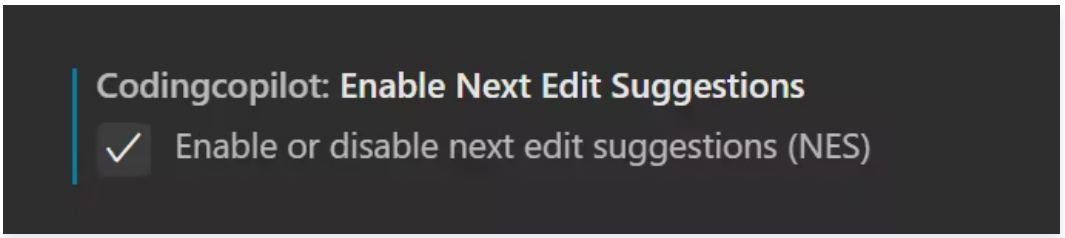


- 补全相关默认快捷键：

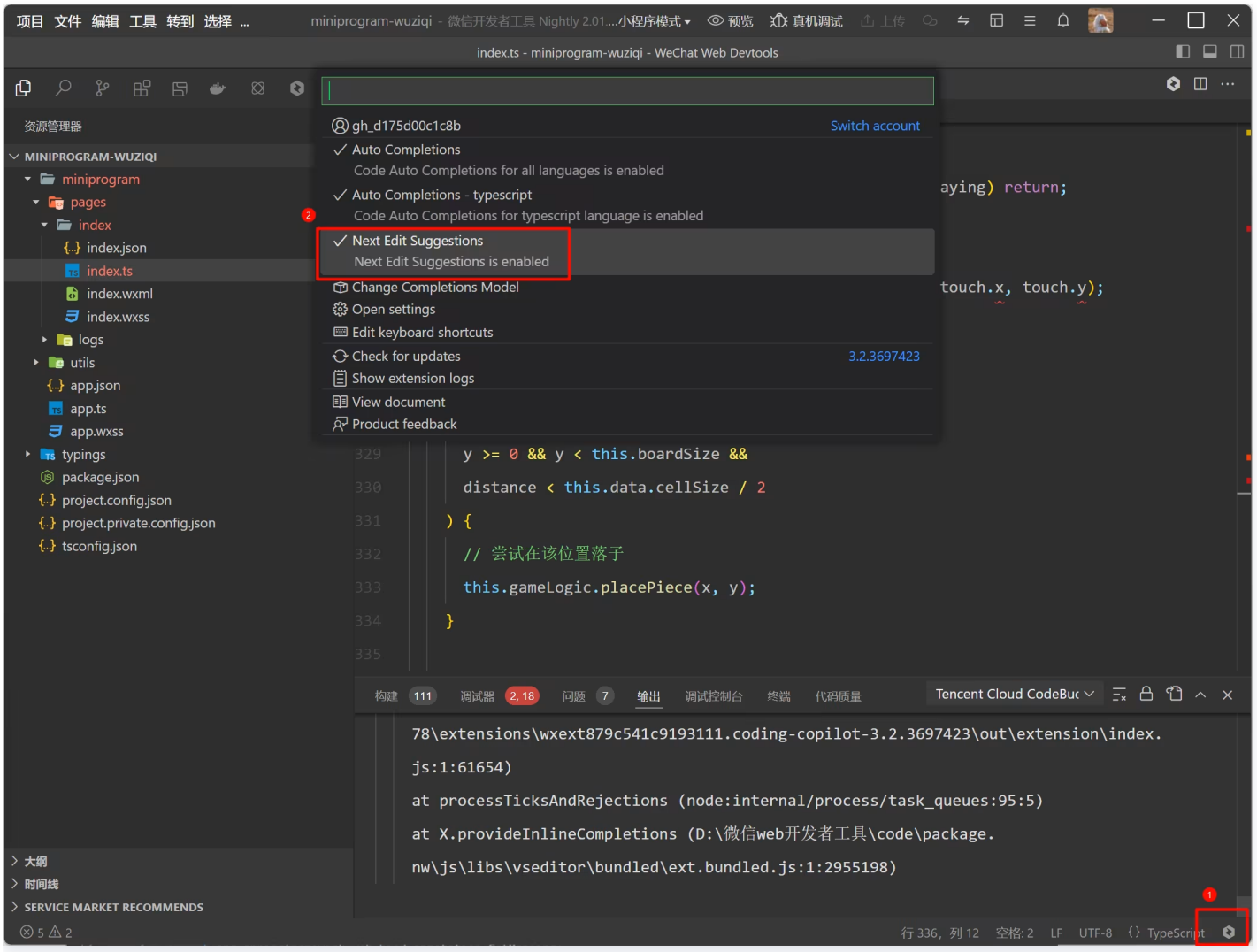
操作	macOS	Windows
采纳所有推荐代码	Tab	Tab
按行采纳推荐代码	⌘ + ↓	Ctrl + ↓
按词采纳推荐代码	⌘ + →	Ctrl + →
手动触发推荐	⌥ + \	Alt + \
撤销当前推荐状态	Escape	Escape

补全 Pro 能力

补全 Pro 能力，即 NES（Next Edit Suggestions），不仅可以对当前光标处的代码进行补全，也可以对其它相关地方的代码进行补全，即预测开发者接下来要写的代码，并提前给出建议，具有“前瞻性”的核心特点，实现更好的补全效果。默认为开启状态，支持自动触发和手动触发，手动触发的快捷键为 **Ctrl + Shift + Enter**。

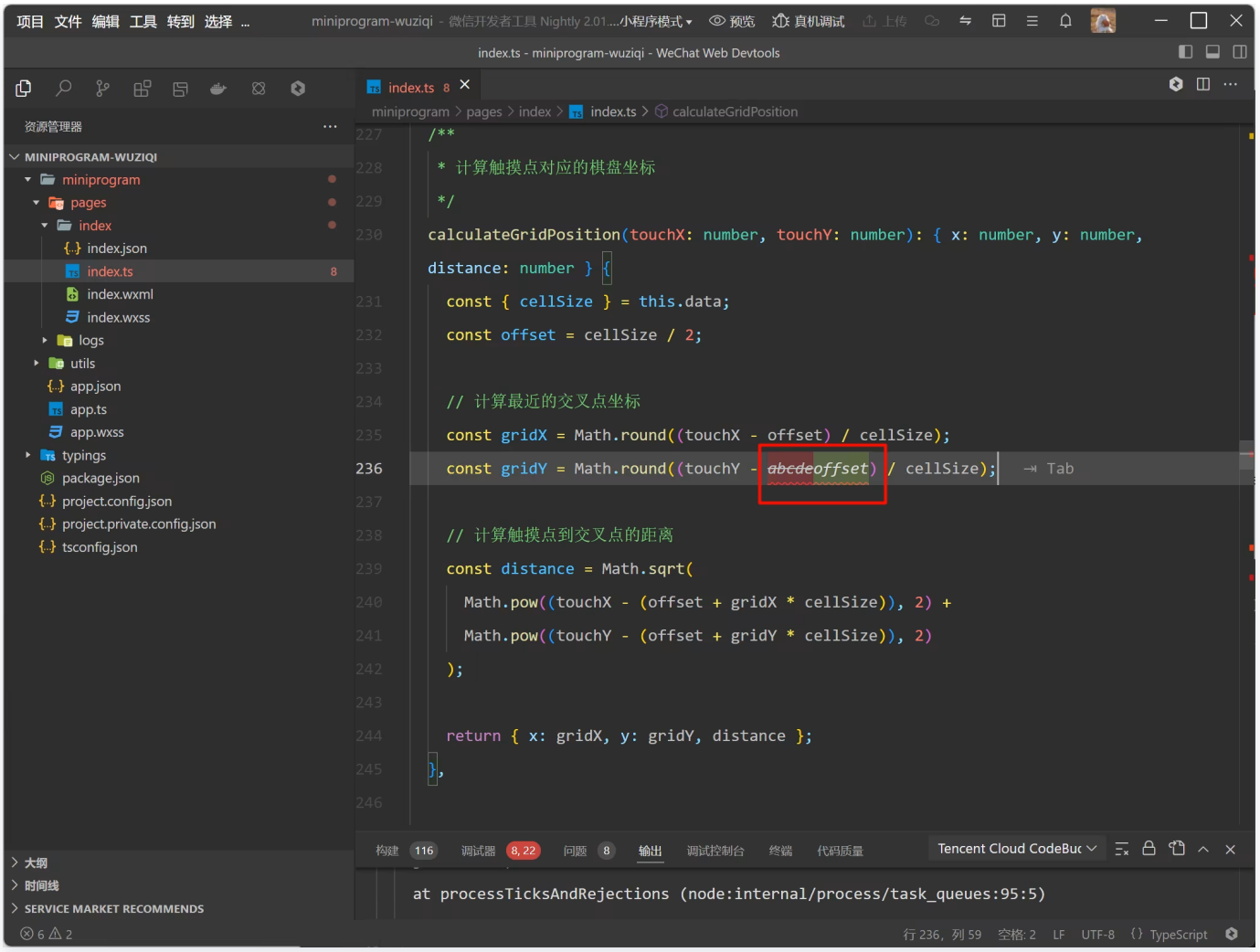


您可以在底部状态栏中单击插件小图标，在弹出的菜单选项里可以选择启用或禁用 NES，如下图所示：



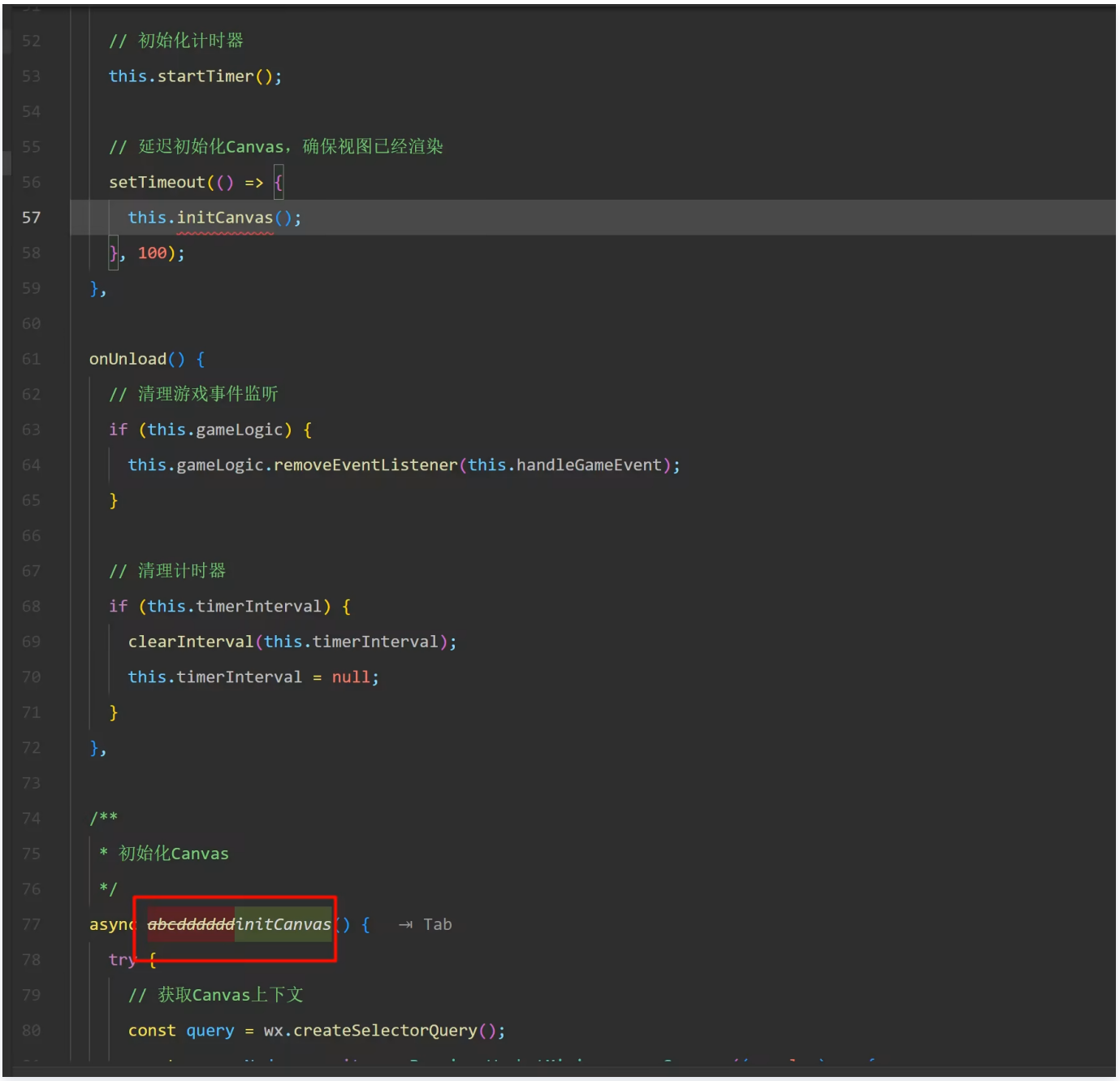
- **修改函数或方法名的补全 Pro 场景**

  修改变量名后，根据光标的移动位置，以及代码情况，捕捉到上下文信息后，在合理的光标位置出现补全推荐。



- **修改函数或方法名的补全 Pro 场景**

  函数或方法名修改后，在引用处，通过光标的定位，自动触发 NES。



```
    // 初始化计时器
    this.startTimer();

    // 延迟初始化Canvas，确保视图已经渲染
    setTimeout(() => {
      this.initCanvas();
    }, 100);
  },

  onUnload() {
    // 清理游戏事件监听
    if (this.gameLogic) {
      this.gameLogic.removeEventListener(this.handleGameEvent);
    }

    // 清理计时器
    if (this.timerInterval) {
      clearInterval(this.timerInterval);
      this.timerInterval = null;
    }
  },

  /**
   * 初始化Canvas
   */
  async abcdddddinitCanvas() {    ⇥ Tab
    try {
      // 获取Canvas上下文
      const query = wx.createSelectorQuery();
```

CodeBuddy IDE

代码补全 Tab，是支持多行编辑的智能代码补全。基于当前代码内容和项目上下文分析，提供多行代码补全建议，并支持基于历史编辑记录的“下一步编辑建议”功能，可通过 Tab 键接受补全建议。

```
// 绘制滚动云朵
drawScrollingClouds() {
    this.ctx.fillStyle = 'rgba(255, 255, 255, 0.1)';

    // 多层云朵，不同速度滚动
    const layers = [
        { speed: 0.2, scale: 1, opacity: 0.1 },
        { speed: 0.5, scale: 0.8, opacity: 0.15 },
        { speed: 0.8, scale: 0.6, opacity: 0.2 }
    ];

    layers.forEach((layer, layerIndex) => {
        this.ctx.fillStyle = `rgba(255, 255, 255, ${layer.opacity})`;

        for (let i = 0; i < 8; i++) {
            const baseX = i * 300;
            const offsetX = (this.backgroundOffset * layer.speed) % (300 * 8);
            const x = baseX - offsetX;
            const y = 50 + layerIndex * 40 + Math.sin(i * 0.5) * 20;
            const size = 30 * layer.scale;

            if (x > -100 && x < this.canvas.width + 100) {
                this.ctx.beginPath();
                this.ctx.arc(x, y, size, 0, Math.PI * 2);
                this.ctx.fill();
            }
        }
    });
}
```

补全 Pro 能力

补全 Pro 能力，即下一个编辑建议，NES（Next Edit Suggestions），不仅可以对当前光标处的代码进行补全，也可以对其它相关地方的代码进行补全，即预测开发者接下来要写的代码，并提前给出建议，具有“前瞻性”的核心特点，实现更好的补全效果。默认为开启状态，支持自动触发和手动触发，手动触发的快捷键为 Windows：**Ctrl + Shift + Enter**，Mac：**⌘ Command + ⇧ Shift + Enter**。

- 修改函数或方法名的补全 Pro 场景

  在函数或方法名的定义处或引用处修改后，通过光标的定位，自动触发 NES。

```
// 调整画布大小
abbbbbbbresizeCanvas() {    ⇥ Tab
    const container = document.getElementById('gameContainer');
    const containerWidth = container.clientWidth;
    const containerHeight = container.clientHeight;

    // 保持16:9比例
    const aspectRatio = 16 / 9;
    let canvasWidth = containerWidth * 0.9;
    let canvasHeight = canvasWidth / aspectRatio;

    if (canvasHeight > containerHeight * 0.9) {
        canvasHeight = containerHeight * 0.9;
        canvasWidth = canvasHeight * aspectRatio;
    }

    this.canvas.style.width = canvasWidth + 'px';
    this.canvas.style.height = canvasHeight + 'px';
}
```

- 修改变量名的补全 Pro 场景

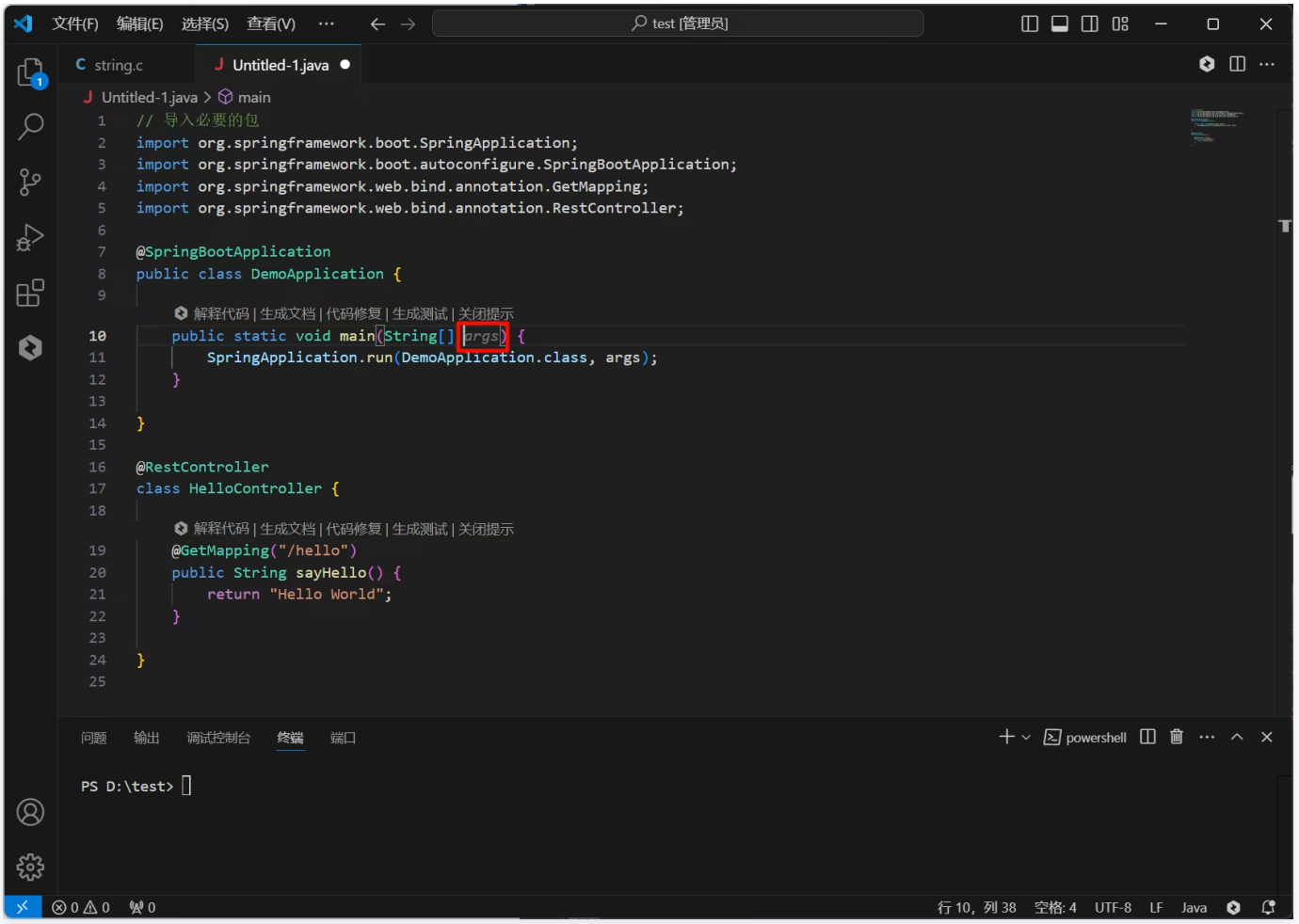
  修改变量名后，根据光标的移动，以及代码情况，捕捉到上下文信息后，在合理的光标位置出现补全推荐。

```
// 保持16:9比例
const aspectRatio = 16 / 9;
let canvasWidth = containerWidth * 0.9;
let canvasHeight = canvasWidth / bbbbbbbbaspectRatio;    ⇥ Tab

if (canvasHeight > containerHeight * 0.9) {
    canvasHeight = containerHeight * 0.9;
    canvasWidth = canvasHeight * aspectRatio;
}

this.canvas.style.width = canvasWidth + 'px';
this.canvas.style.height = canvasHeight + 'px';
```

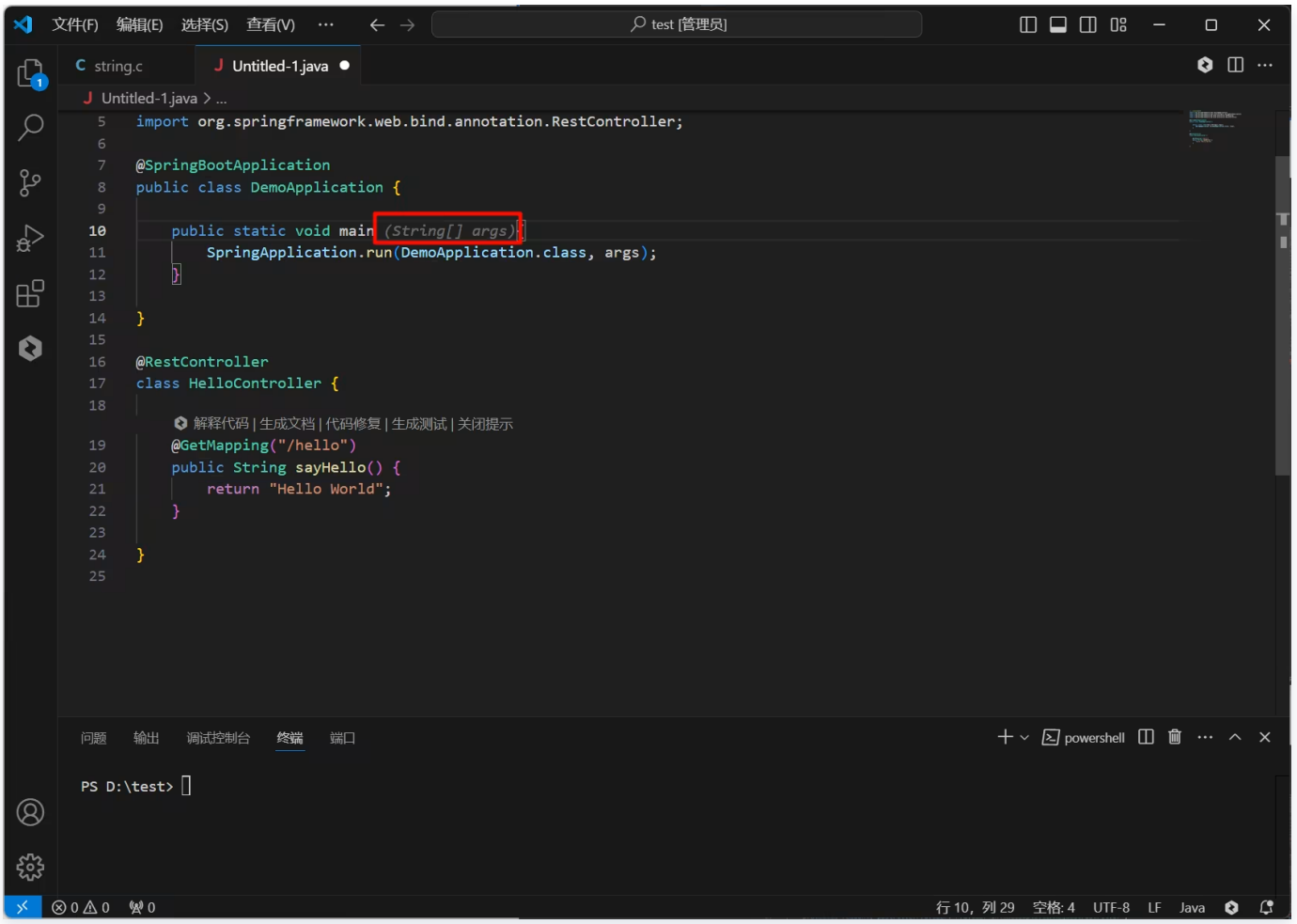
代码补全触发时机

插件会根据光标所在位置合理选择合适的位置进行代码补全，例如当光标位置在代码行尾附近位置，或者换行的时候才会触发代码补全。以 VS Code 为例（JetBrains IDE 同理）。

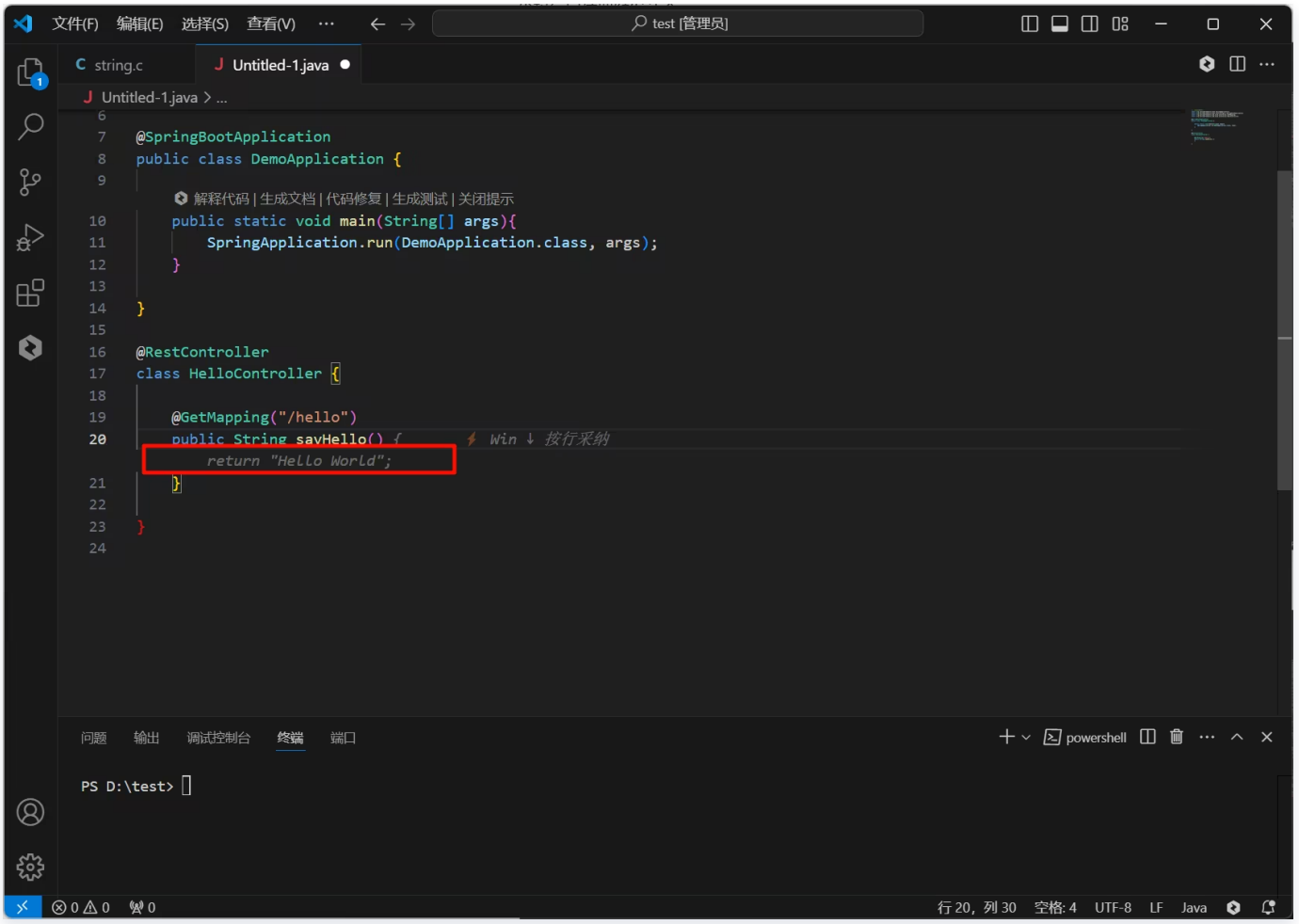
- 行尾右边小括号前面位置触发补全。



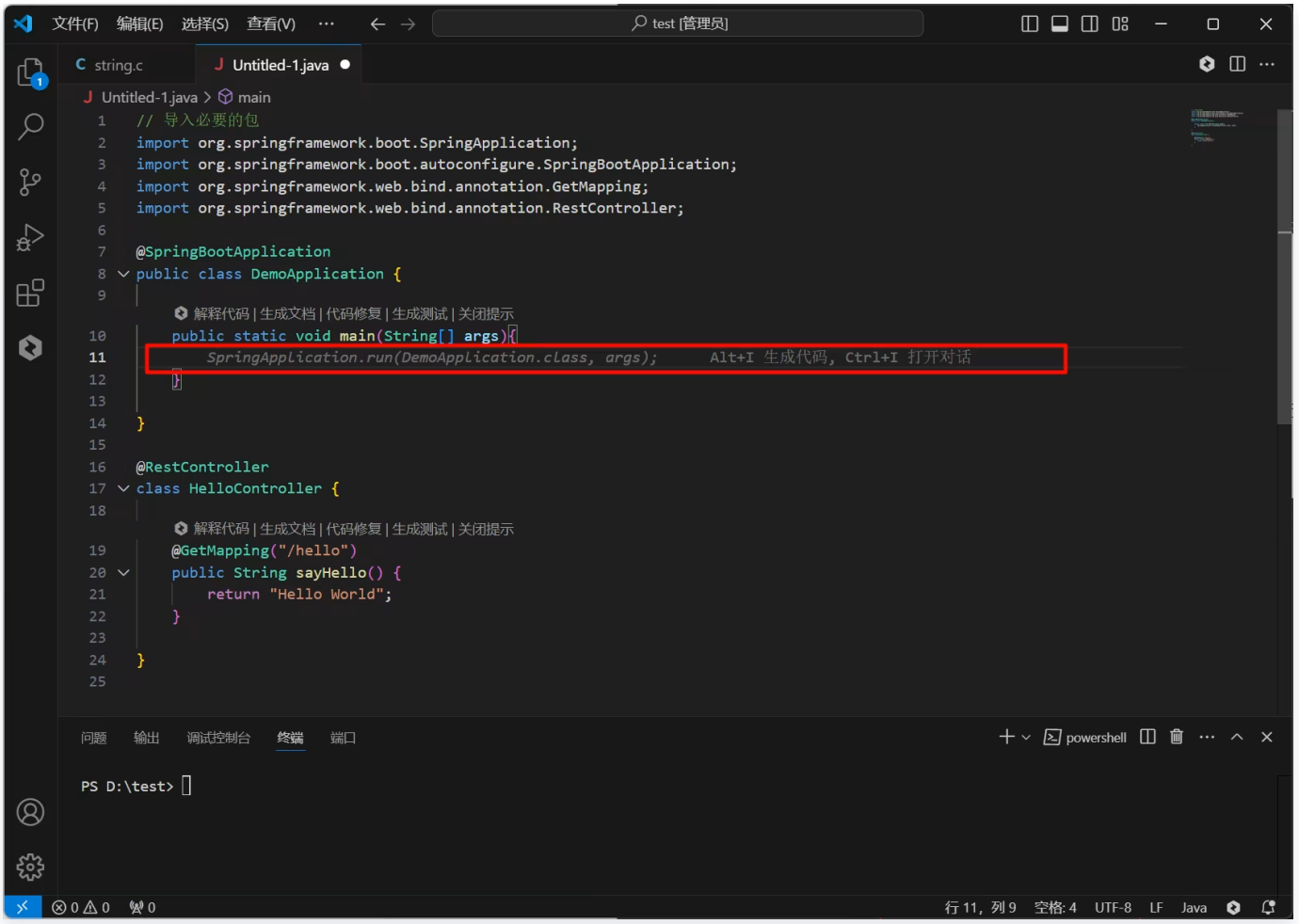
- 行尾右边大括号前面位置触发补全。


文件(F) 编辑(E) 选择(S) 查看(V) ···
test [管理员]
string.c
Untitled-1.java
Untitled-1.java > ...
import org.springframework.web.bind.annotation.RestController;
@SpringBootApplication
public class DemoApplication {
public static void main (String[] args) {
SpringApplication.run(DemoApplication.class, args);
}
}
@RestController
class HelloController {
解释代码 | 生成文档 | 代码修复 | 生成测试 | 关闭提示
@GetMapping("/hello")
public String sayHello() {
return "Hello World";
}
}
问题 输出 调试控制台 终端 端口
powershell
PS D:\test>
行 10，列 29 空格: 4 UTF-8 LF Java

- 代码行尾位置触发补全。


文件(F) 编辑(E) 选择(S) 查看(V) ···
test [管理员]
string.c
Untitled-1.java
Untitled-1.java > ...
@SpringBootApplication
public class DemoApplication {
解释代码 | 生成文档 | 代码修复 | 生成测试 | 关闭提示
public static void main(String[] args){
SpringApplication.run(DemoApplication.class, args);
}
}
@RestController
class HelloController {
@GetMapping("/hello")
public String sayHello() {
Win ↓ 按行采纳
return "Hello World";
}
}
问题 输出 调试控制台 终端 端口
powershell
PS D:\test>
行 20，列 30 空格: 4 UTF-8 LF Java

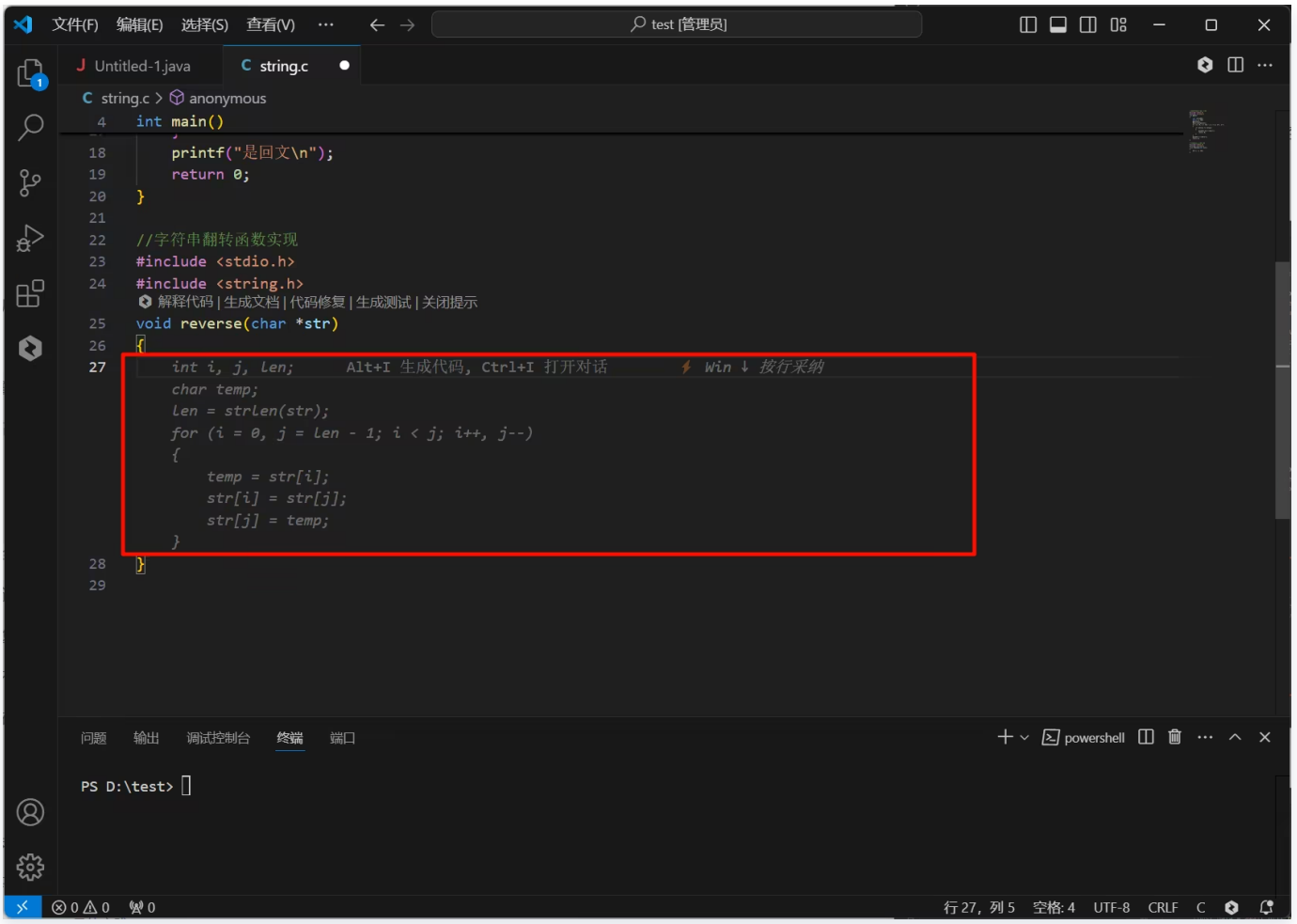
- 代码换行位置触发补全。



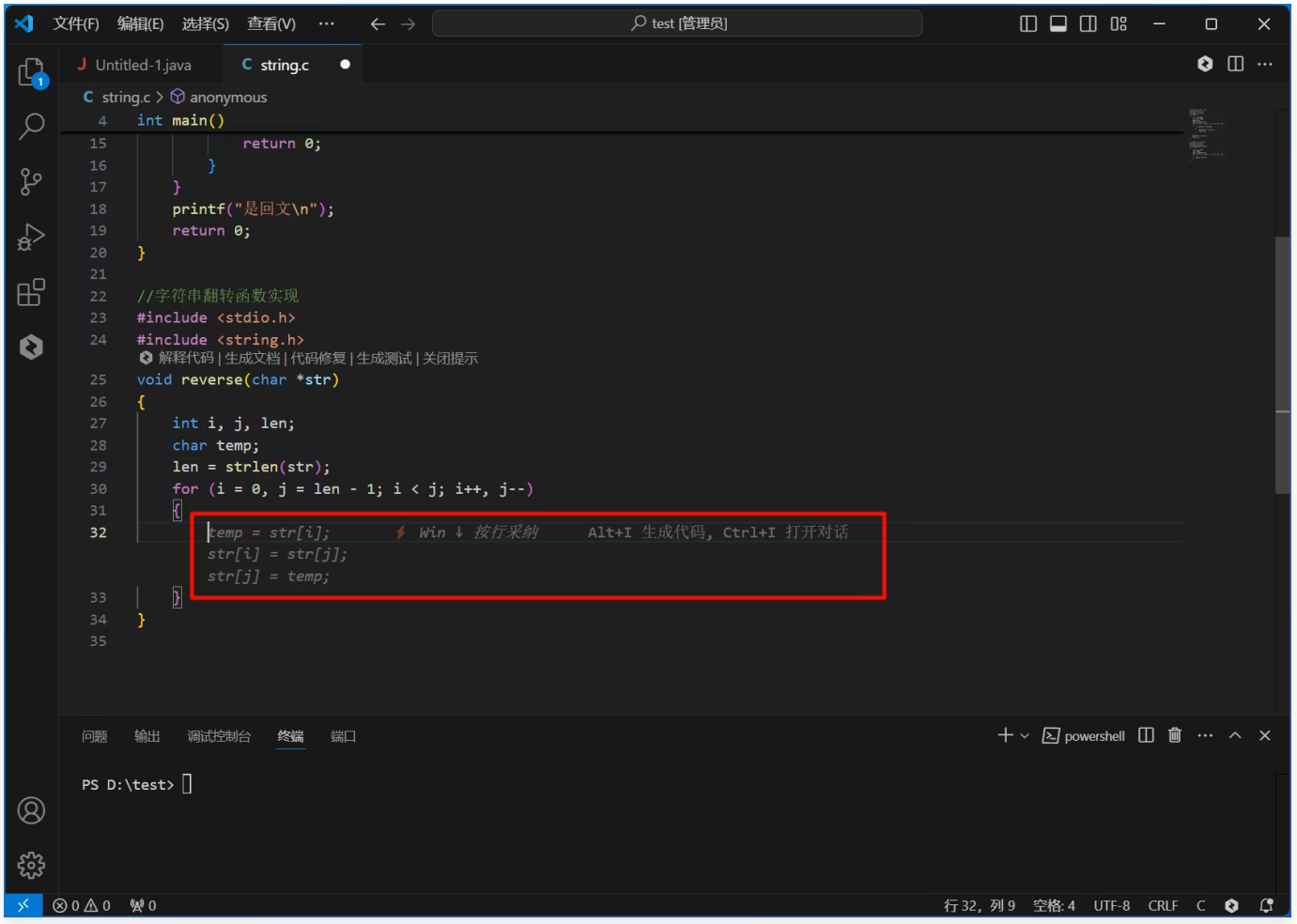
代码补全内容的粒度

实际使用过程中，插件补全内容的粒度在不同的代码位置是不一样的。实验发现，并不是补全的内容越多越好（可能补全的越多，错的越多，而且速度也会变慢，补全内容的长度与补全所需要的时间成正比），在不同代码位置，补全的内容的多少（粒度）是不一样的。例如在一个空的代码块中，插件则按代码块粒度进行补全；在非空代码块中，大部分情况按单行粒度进行补全；在类定义块中，或者顶级块中，就算块内不为空，也按代码块粒度进行补全。以 VS Code 为例（JetBrains IDE 同理）。

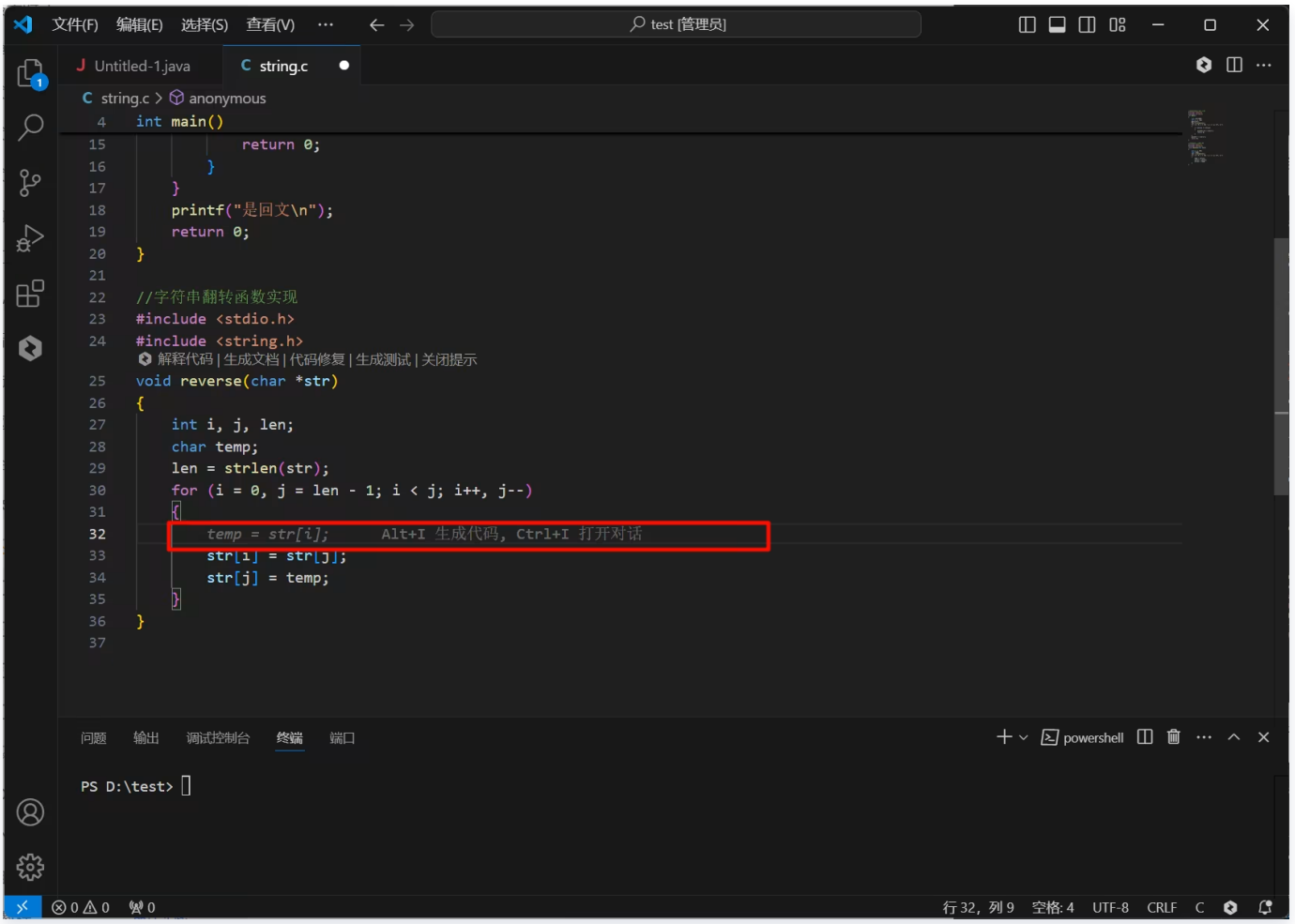
- 光标在方法或者函数体内，且体内为空，按块补全。



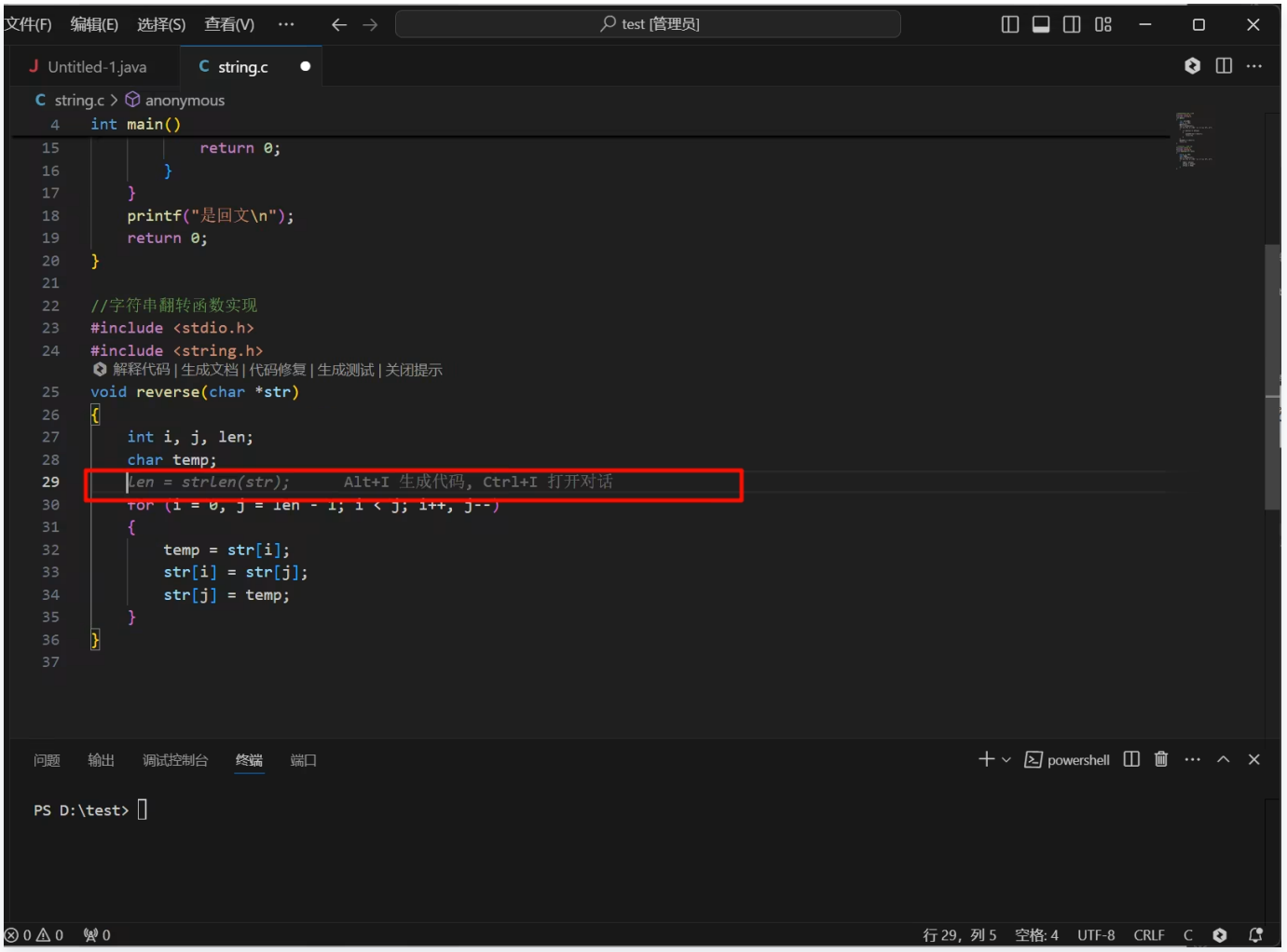
- 光标在语句内，例如 for 语句、try 语句等，且体内为空，按块补全。



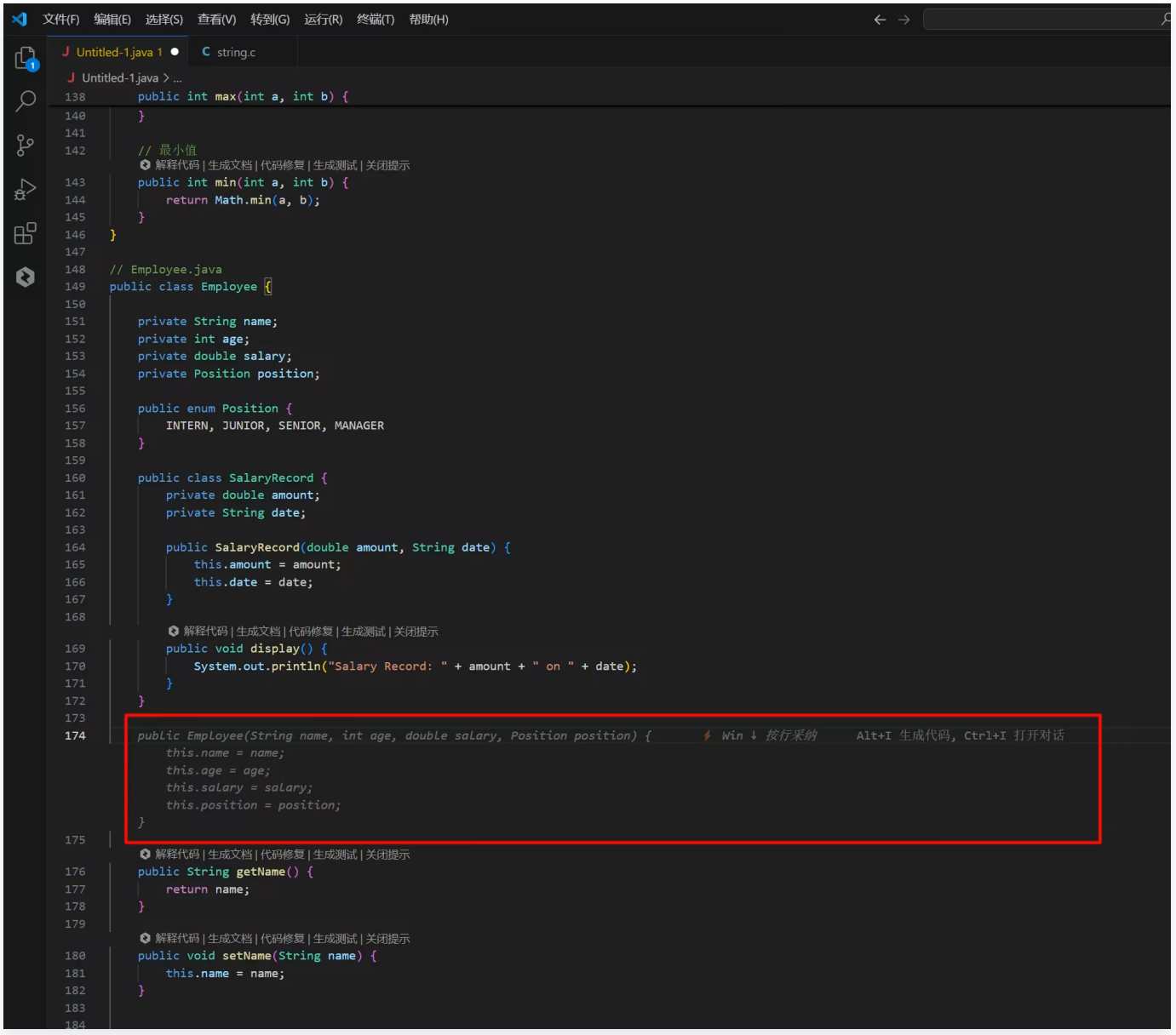
- 光标在语句内，例如 for 语句、try 语句等，且体内不为空，则按单行补全。



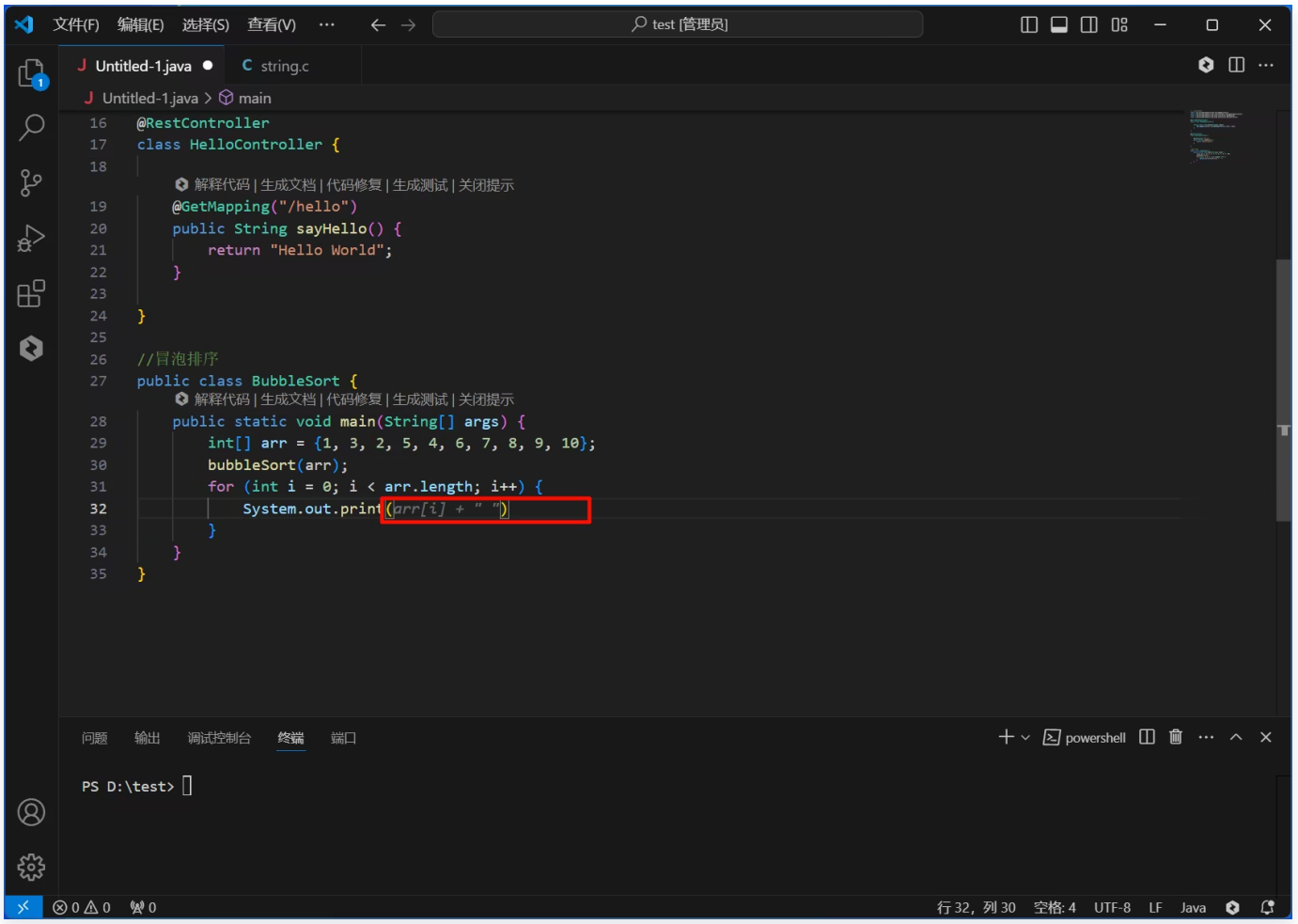
- 光标在方法或者函数体内，且体内不为空，按单行补全。



- 光标在类定义体内，按块补全。



- 光标在一行代码尾部范围，则按单行补全。



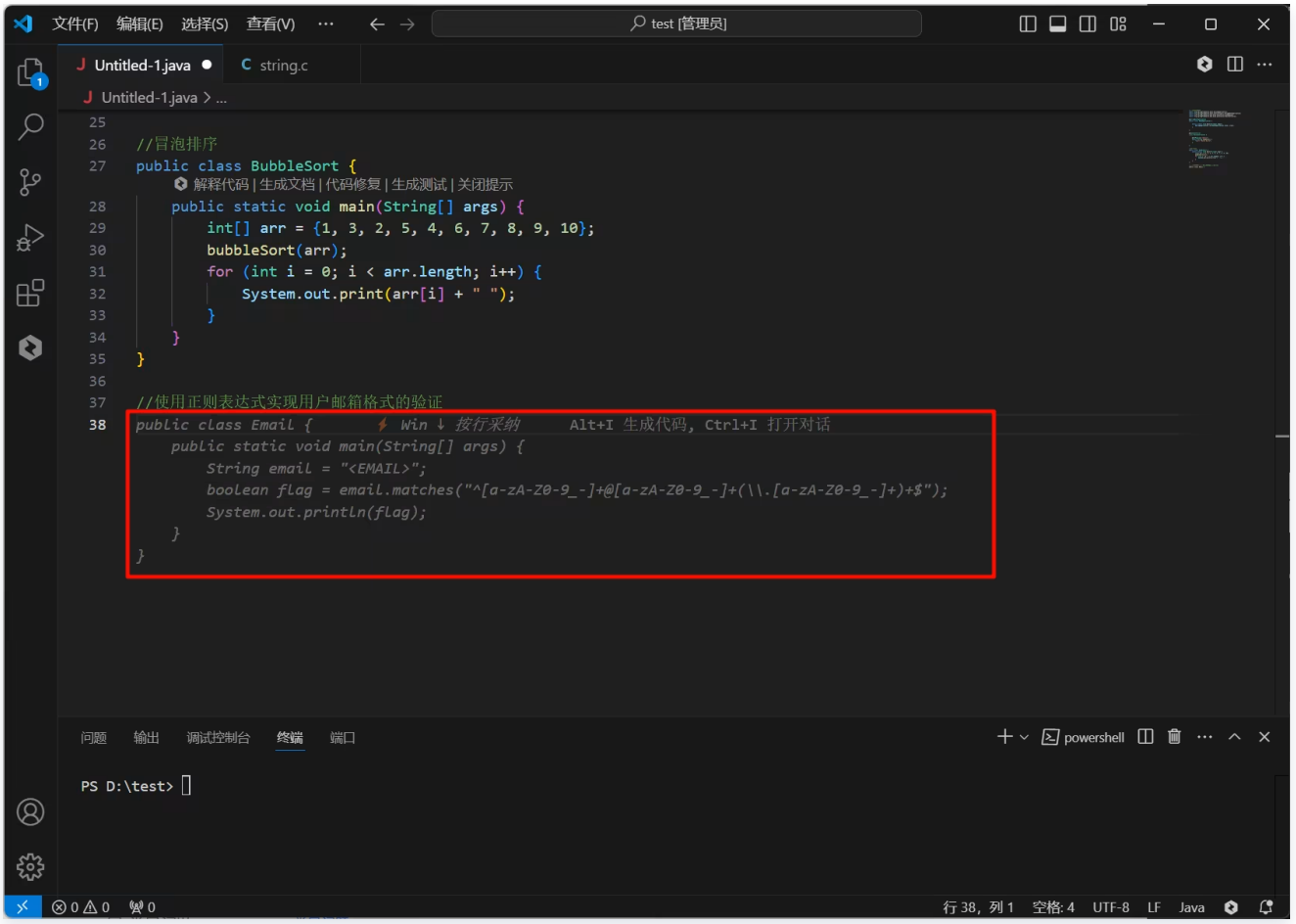
如何提升代码补全结果的正确性

代码补全结果的正确性是相对当前开发者来说的，对插件来说可能是正确合理的，但是对当前开发者来说可能不正确、不合理。为了帮助您理解，可以把代码补全功能，想象成一个隐藏在背后的程序员：

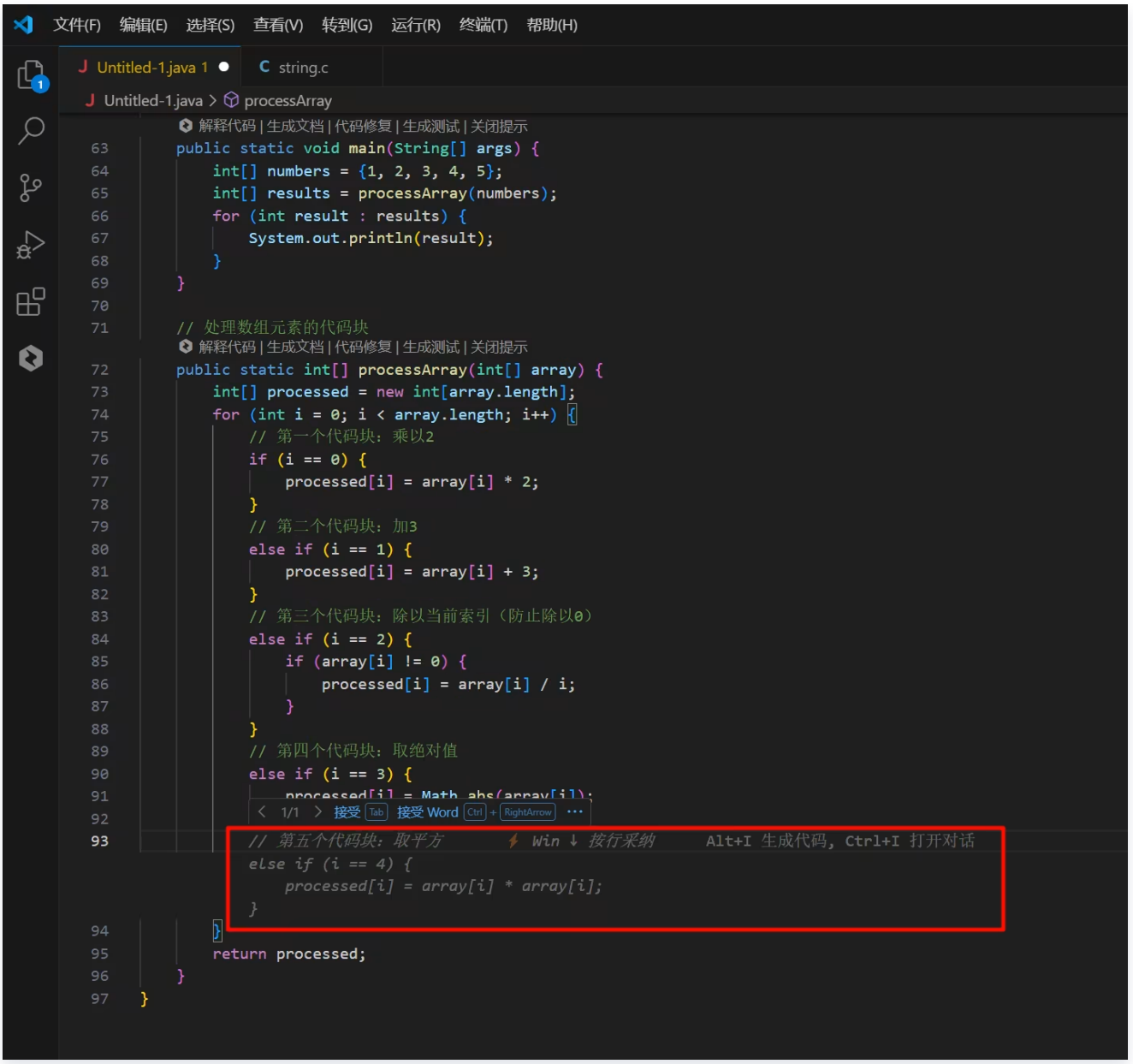
- 它的眼睛几乎只能看到当前光标之前的代码（代码行数和字符数也是有一定限制的）。
- 具有较强的逻辑推理和模仿能力。
- 它只能接着当前光标位置继续完成后续的代码。

所以，从上面三个特性就能很容易推出可以通过使用光标之前的代码**引导**它帮忙我们写出更为正确的后续代码。引导的方式有很多种，如下所示：

1. 通过代码注释直接告诉它，我想要做什么，例如实现一个冒泡排序。
2. 通过已有平级同类型代码，引导出可能的补全内容，例如用户模块的 CRUD 功能实现，通过实现 CRU，引导它模仿 CRU 的实现方式，实现 D 功能。
3. 让自己的代码**有迹可循**，例如代码风格统一、变量和方法等命名具体且有规律、代码结构具有规律。它很擅长挖掘已有代码潜在的规律，然后去模仿推导。以 VS Code 为例（JetBrains IDE 同理）。
    - 通过代码注释引导补全。



- 通过已有平级同类型代码引导补全。



- 通过潜在代码规律引导补全。


文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) 终端(T) 帮助(H)
Untitled-1.java 1
string.c
Untitled-1.java > ...
72 public static int[] processArray(int[] array) {
89 // 第四个代码块：取绝对值
90 else if (i == 3) {
91 processed[i] = Math.abs(array[i]);
92 }
93 // 第五个代码块：取平方
94 else if (i == 4) {
95 processed[i] = array[i] * array[i];
96 }
97 }
98 return processed;
99 }
100 }
101
102 public class RegularMathOperations {
103 // 加法
解释代码 | 生成文档 | 代码修复 | 生成测试 | 关闭提示
104 public int add(int a, int b) {
105 return a + b;
106 }
107
108 // 减法
解释代码 | 生成文档 | 代码修复 | 生成测试 | 关闭提示
109 public int subtract(int a, int b) {
110 return a - b;
111 }
112
1/1 接受 Tab 接受 Word Ctrl + RightArrow
113
114 public int multiply(int a, int b) {
return a * b;
}
Win ↓ 按行采纳 Alt+I 生成代码, Ctrl+I 打开对话
115
116 // 除法
解释代码 | 生成文档 | 代码修复 | 生成测试 | 关闭提示
117 public double divide(int a, int b) {
118 if (b == 0) {
119 throw new IllegalArgumentException("Divider cannot be zero.");
120 }
121 return (double) a / b;
122 }
123
124 // 求余
解释代码 | 生成文档 | 代码修复 | 生成测试 | 关闭提示
125 public int remainder(int a, int b) {
126 return a % b;
127 }
128
129
130 // 绝对值
解释代码 | 生成文档 | 代码修复 | 生成测试 | 关闭提示
131 public int absolute(int a) {
132 return Math.abs(a);
133 }
134
135 // 最大值
解释代码 | 生成文档 | 代码修复 | 生成测试 | 关闭提示
136 public int max(int a, int b) {
137 return Math.max(a, b);
138 }
139
问题 1 输出 调试控制台 终端 端口
PS D:\test>

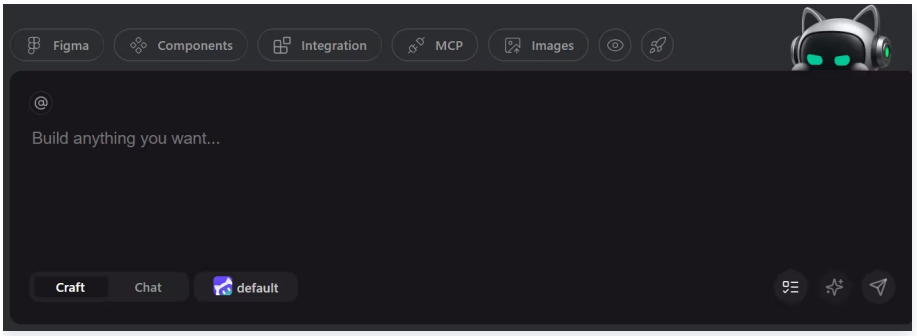
技术对话

最近更新时间：2026-07-20 19:49:31

开发者可以与 AI 助手进行对话，并且咨询和编程技术相关的问题。通过多轮对话生成代码，或者由 AI 帮助解答编码过程中遇到的问题。

CodeBuddy IDE

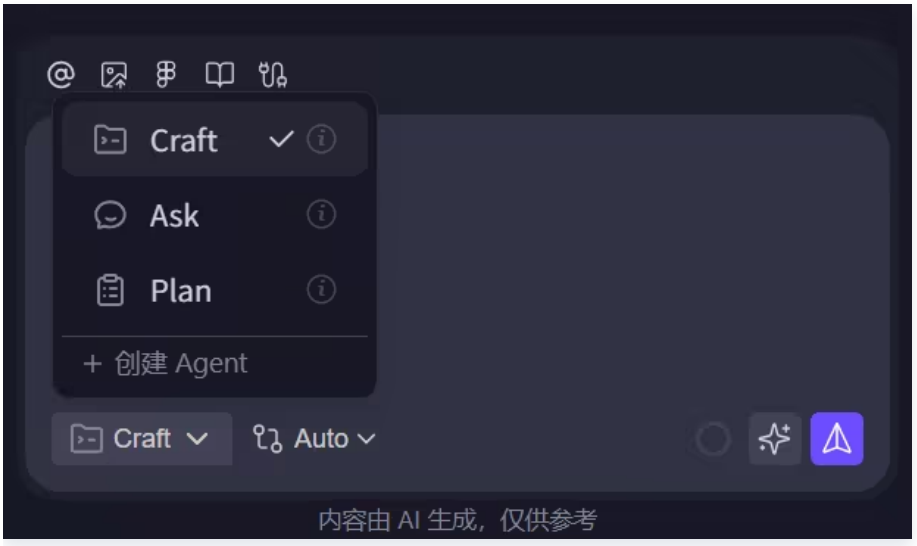
您可以通过自然语言与 Agent 进行对话，Agent 将根据您选择的对话模式（Chat/Craft）帮您提供解决方案，或自动帮您完成复杂编码任务。此外，您可以根据实际场景需求，合理使用内置的功能/工具，例如 Figma、Components、Integration、MCP 等等。



对话模式选择

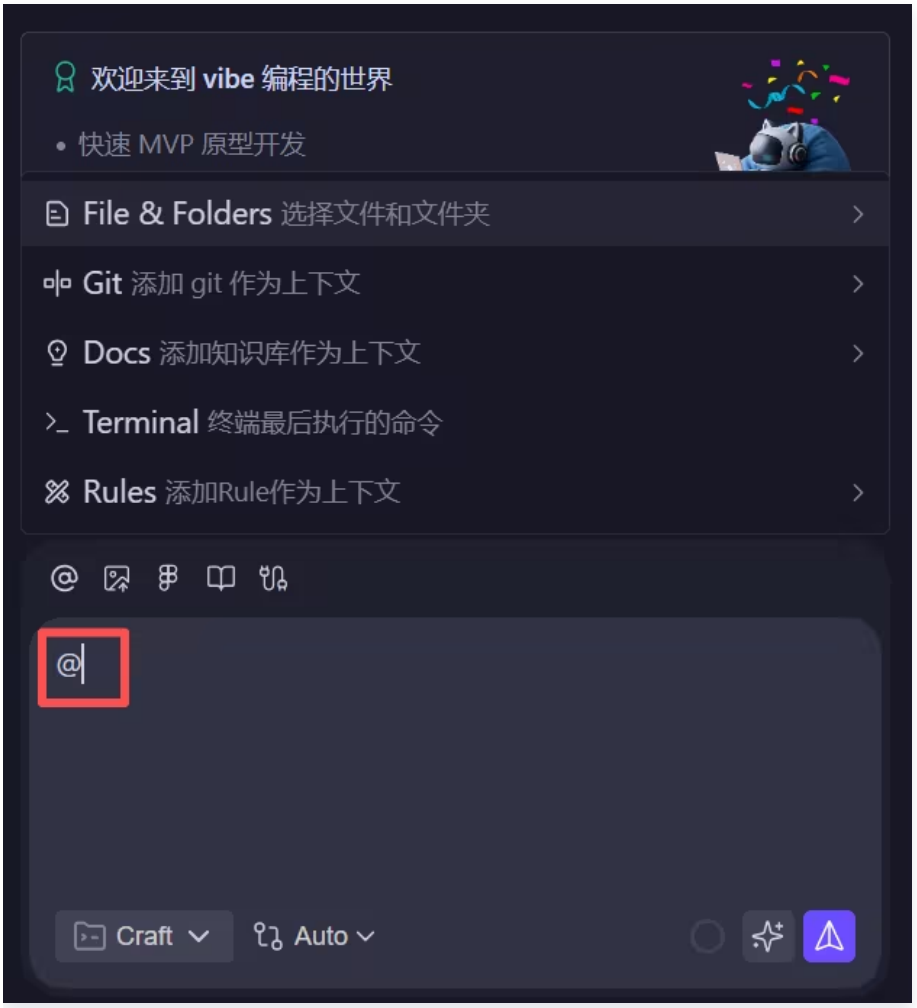
Craft、Ask 和 Plan 三种模式支持您自由切换，适配您的工作流。

- **Craft 模式**：Craft 模式是一个具有 “独立思考” 和 “自主执行” 的编程伙伴，通过自然语言指令，AI 深度理解需求，独立自主完成多文件代码编写与修改。
- **Ask 模式**：类似聊天机器人，主要用于对话式技术问答，解决编码过程中的具体问题或快速生成片段代码。
- **Plan 模式**：即计划模式，启用后，Craft 将根据需求拆解任务，制定计划并澄清需求。如果需求不明确 Craft 会进行询问，您只需回复直到需求清晰后，单击开始按钮或者回复请生成代码，即可开始根据执行计划完成任务。



添加上下文

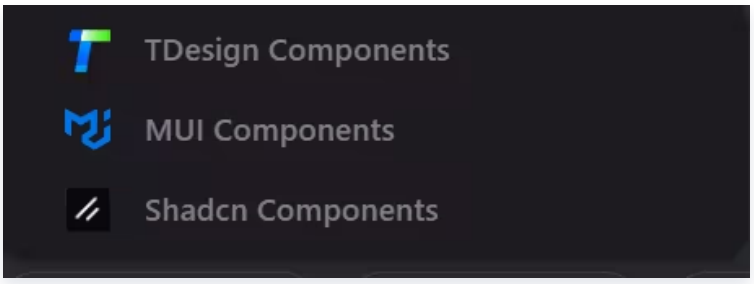
可通过 @ 符号选择添加文件、知识库等资料作为对话上下文，将更多信息提供给模型，提高模型输出的质量。



CodeBuddy IDE 内置了众多的功能/工具供您选择使用，您可以根据实际需求进行选择。



- **Figma**：支持通过内置浏览器打开 Figma 官网，将您的设计稿添加到对话中，提供给 Agent 还原成可交互式的前端代码。
- **Components**：内置了 TDesign Components、MUI Components、Shadcn Components 三种组件库，可快速拼装生成符合规范的设计稿代码。



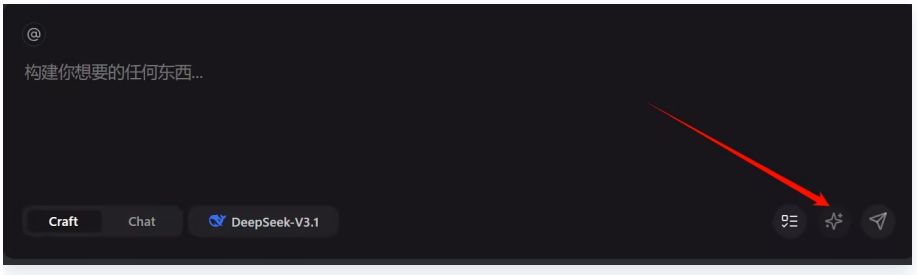
- **Integration**：内置了 Supabase / Tencent CloudBase 后端服务，无需手动配置，自动处理数据库、用户认证或腾讯云生态等需求。



- **部署**：可将您的项目部署至云端，生成可公开访问的链接，方便团队快速验证反馈。

Boost Prompt

即 Prompt 优化，将您模糊的 Prompt 进行优化，更好地完成您的任务，可在 Chat 或 Craft 模式中使用。



对话管理

对话管理包括创建新对话和历史对话记录管理。

- 创建新对话

  如果您有其它不同的任务，或者当前对话中上下文对话内容过多，不希望被干扰，您可以单击 **New Task** 按钮开启一个新对话。



- 历史对话记录管理

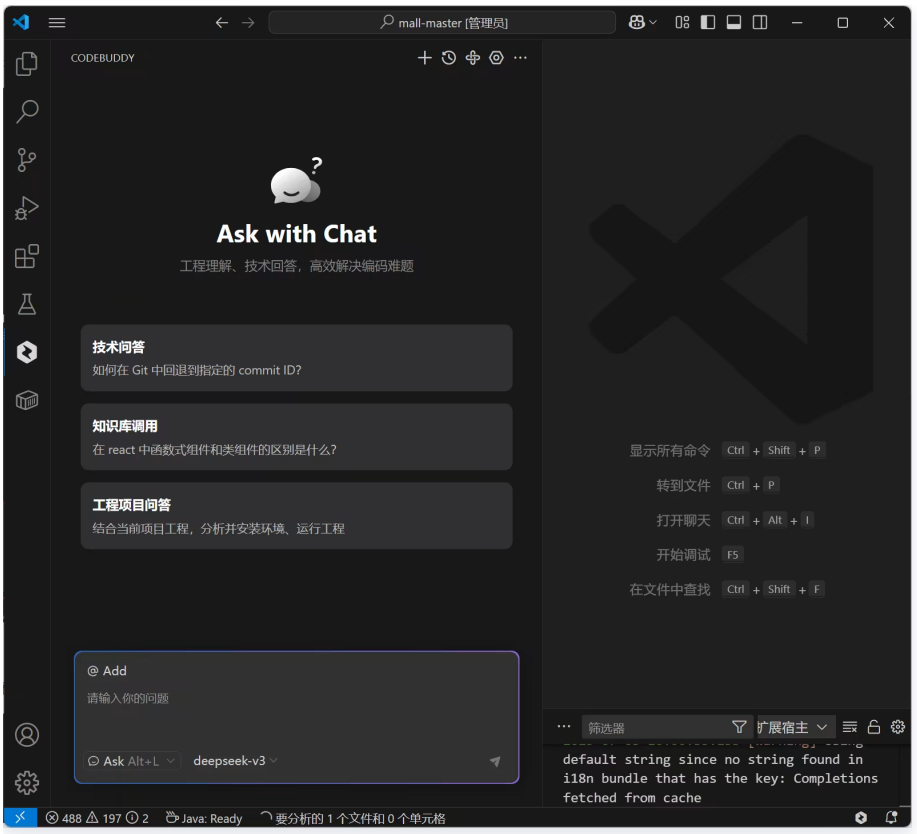
  单击 **History** 按钮后，在这里，您可以查看或切换历史对话记录，并支持对历史对话记录进行删除和导出的操作。



CodeBuddy 插件

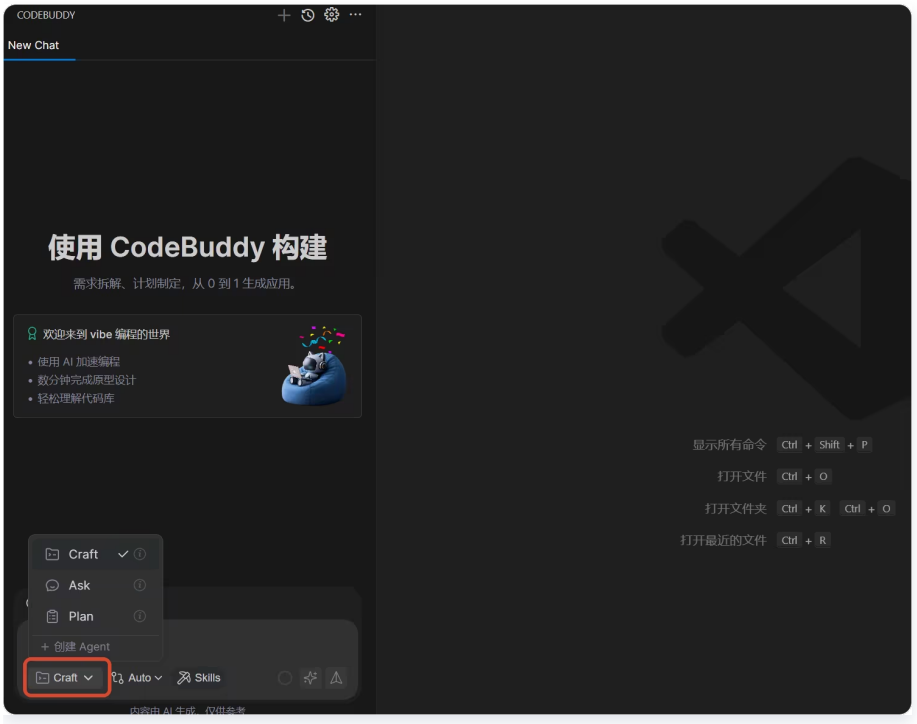
VS Code

开发者可以在对话输入框中通过 `@` 和 `/` 发起对话场景的指令。



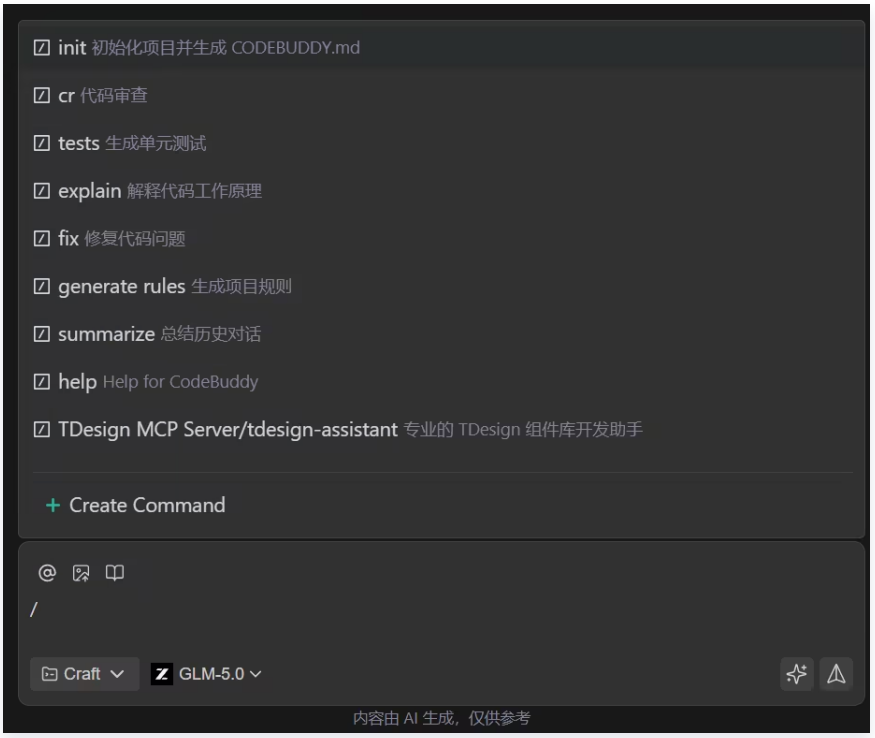
Ask 和 Craft 模式切换

代码助手在对话输入框中支持 Ask 和 Craft 两种模式的切换，满足不同开发场景的需求。Ask 模式主要用于对话式技术问答，解决编码过程中的具体问题或快速生成片段代码；而 Craft 模式用于实现从自然语言需求到生成完整应用的复杂需求，实现“需求到交付”的自动化，属于项目级的软件开发智能体。

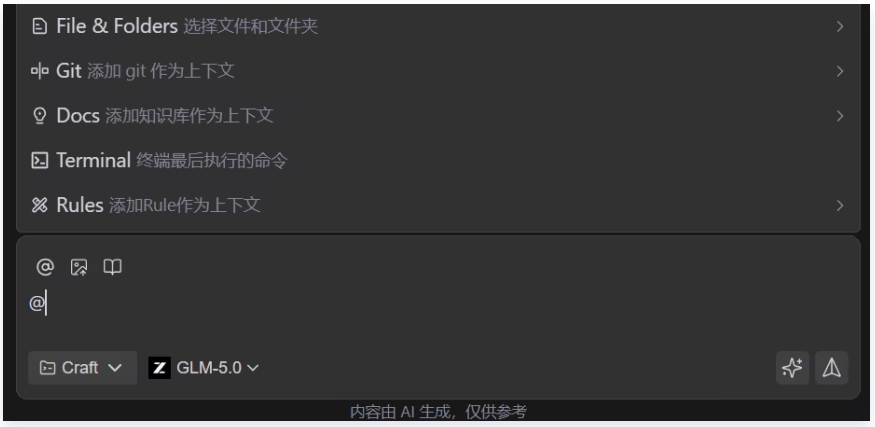


对话快捷指令

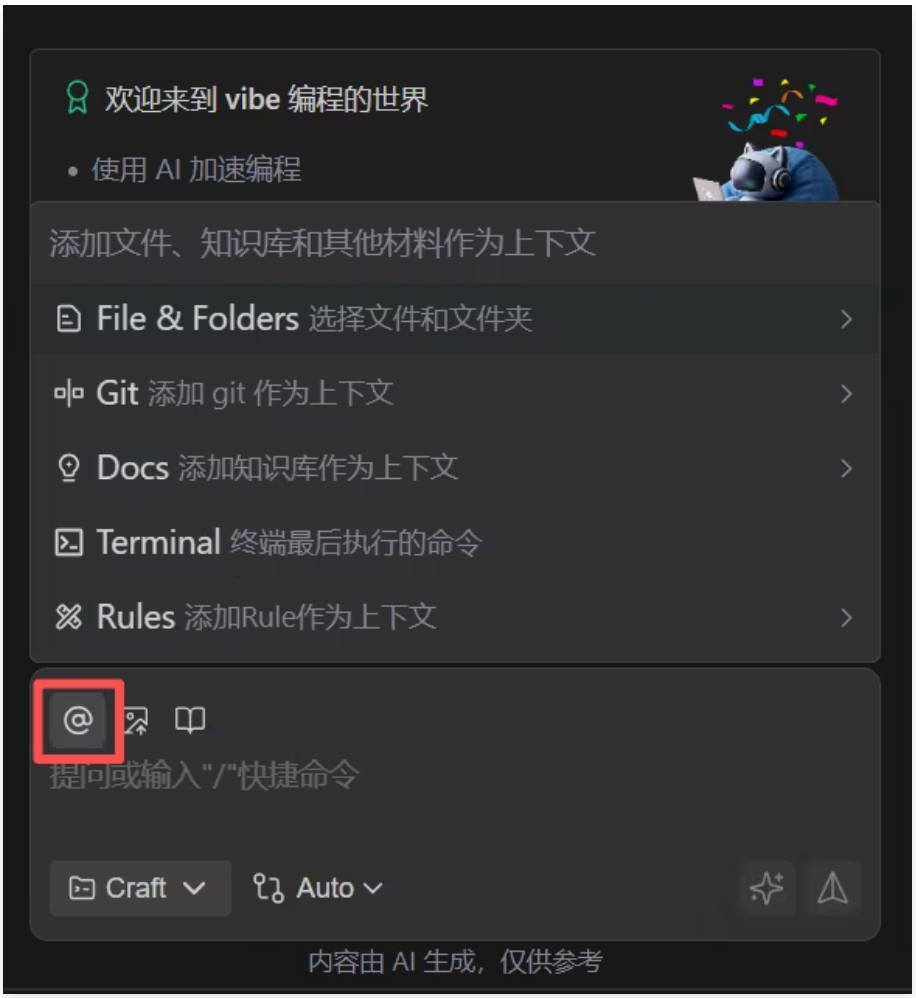
- 在对话输入框中，可输入 `/` 调用预置的快捷指令，以及自定义指令。



- 在对话输入框中，可输入 `@` 选择添加文件、知识库等资料作为对话上下文。



单击 @ 可以选择添加文件、知识库等资料作为上下文，除了支持搜索文件之外，其它的与在对话输入框中输入 @ 添加上下文一致。



模型切换

支持以下内置模型，详情请参见 模型选择 。

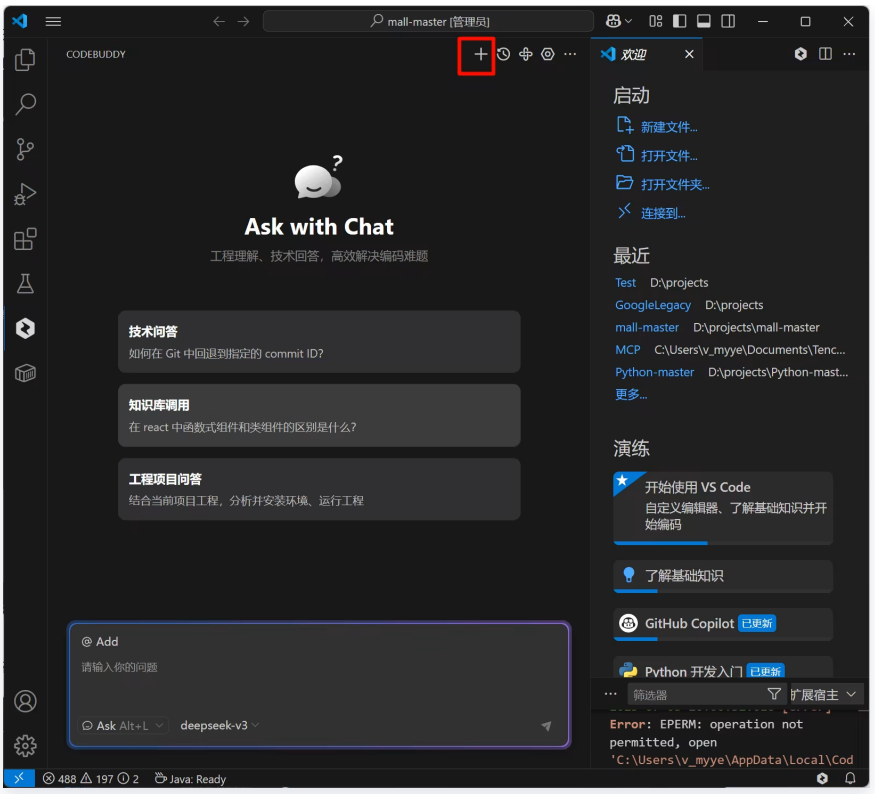


对话管理

对话管理包括创建新对话和历史对话记录管理。

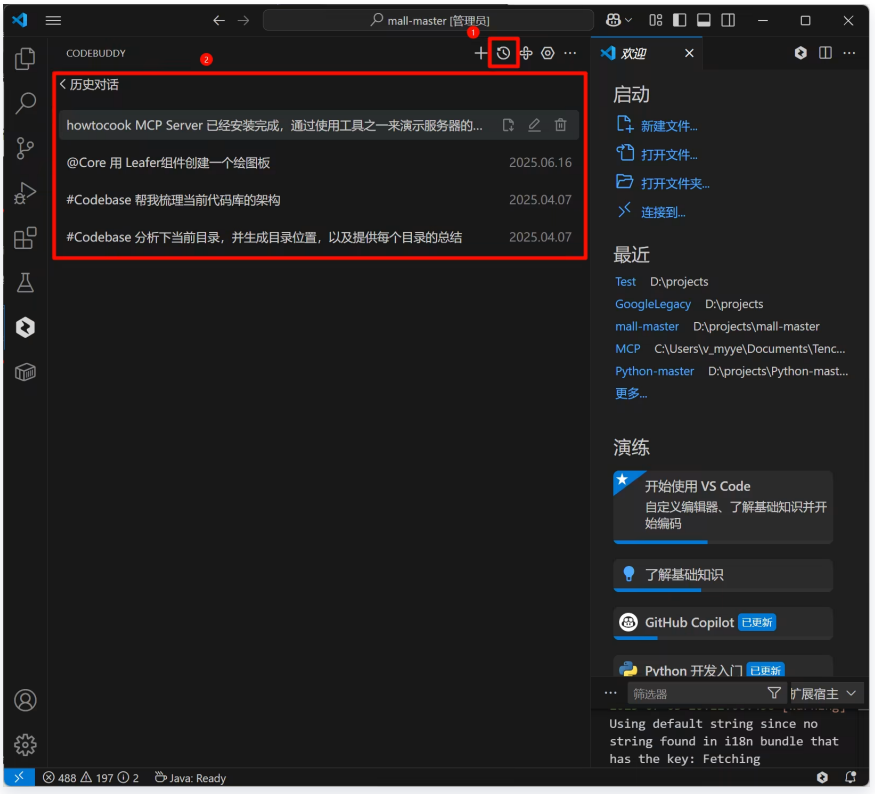
- 创建新对话。

  如果您想重新开启对话或者开启一个新的技术话题，您可以单击**创建新对话**。



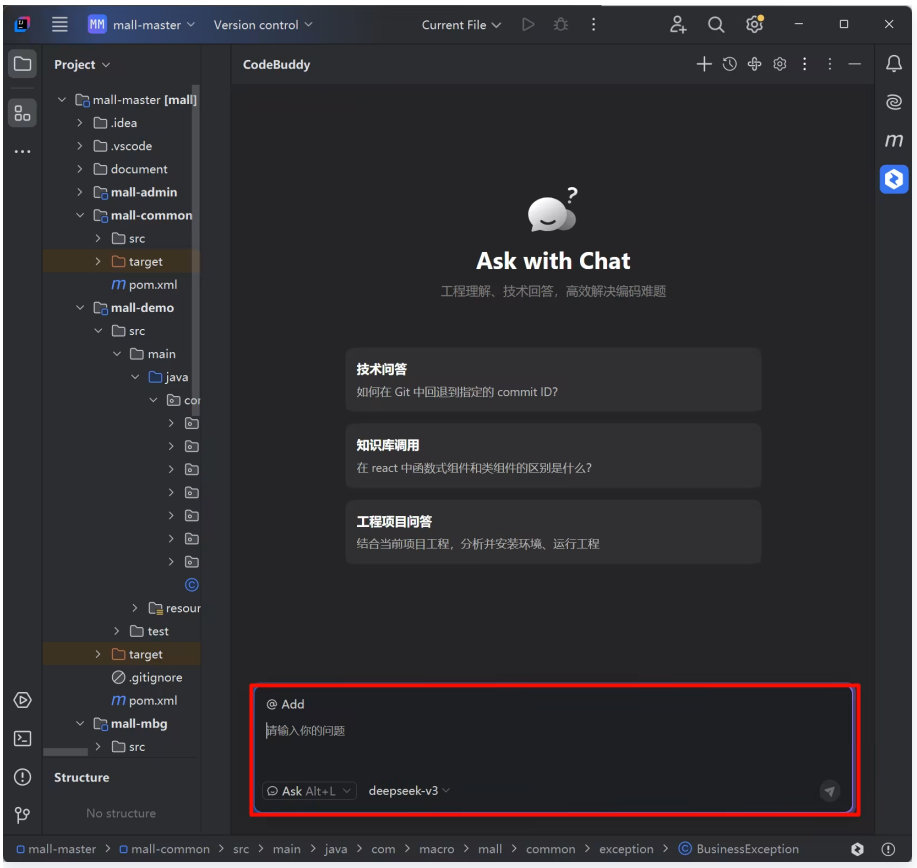
- **历史对话记录管理。**

  单击进入历史对话记录后，在这里，您可以查看历史对话记录，并支持对历史对话进行编辑、导出和删除操作。



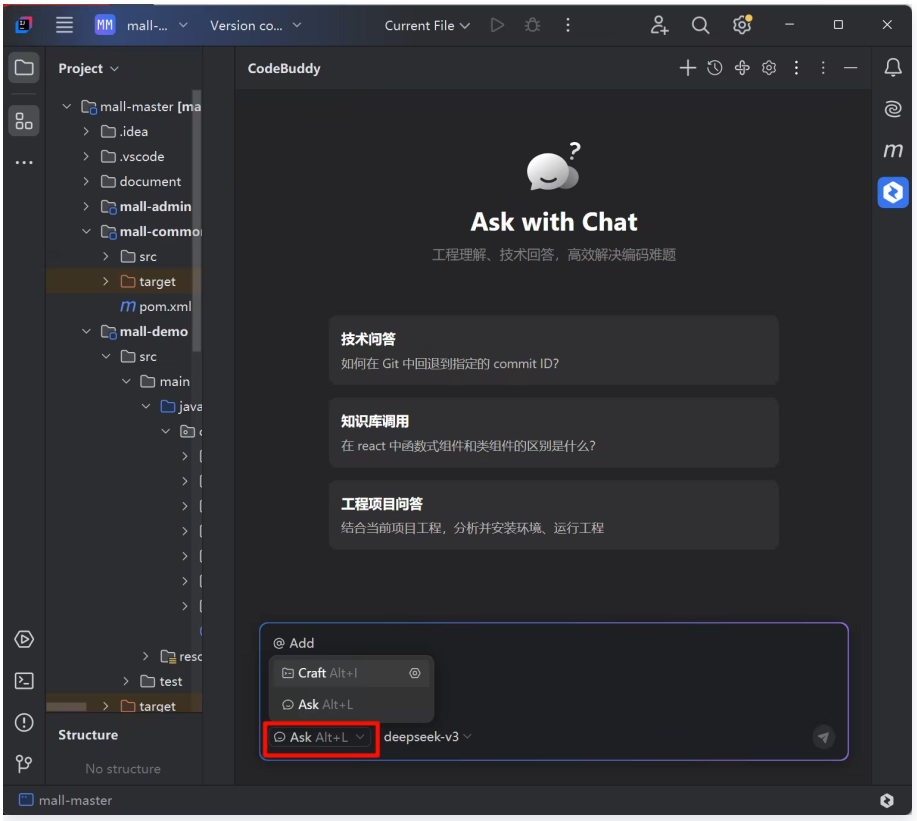
JetBrains IDE

JetBrains IDE 的技术对话使用方式，与 VS Code 基本一致。



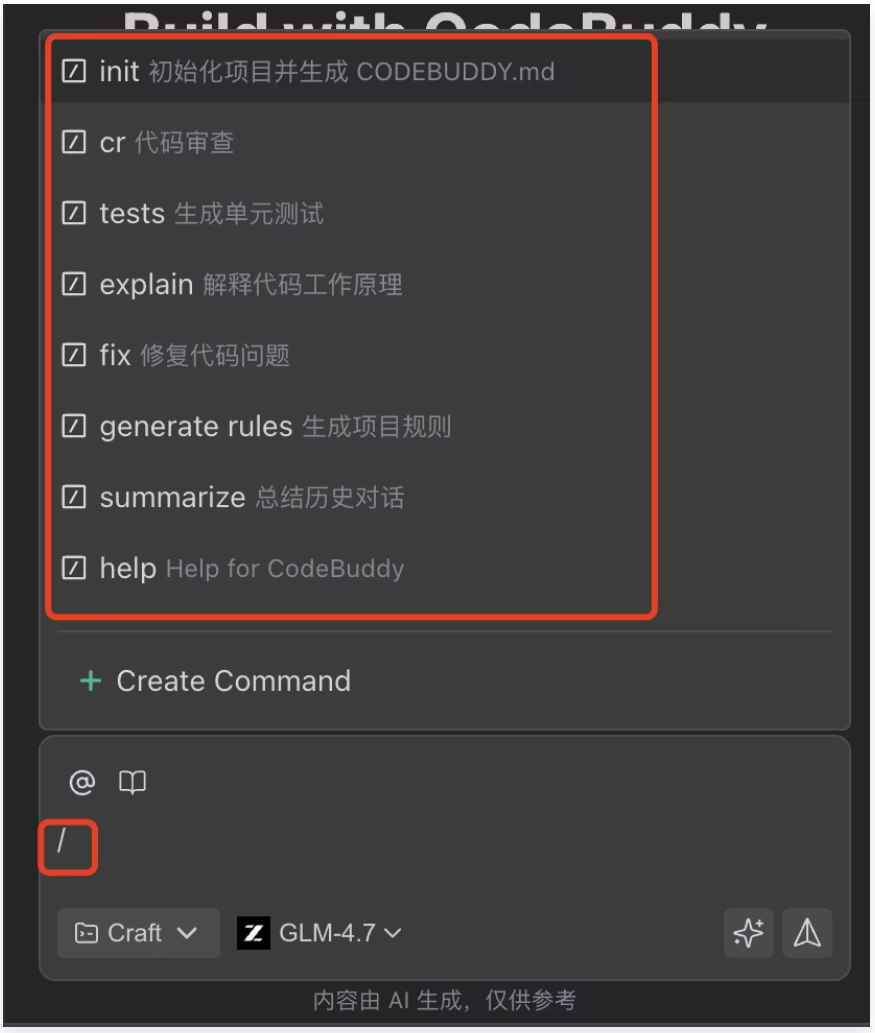
Ask 和 Craft 模式切换

与 VS Code 相同，在 JetBrains IDEs 中，代码助手在对话输入框中同样支持 Ask 和 Craft 两种模式的切换，满足不同开发场景的需求。

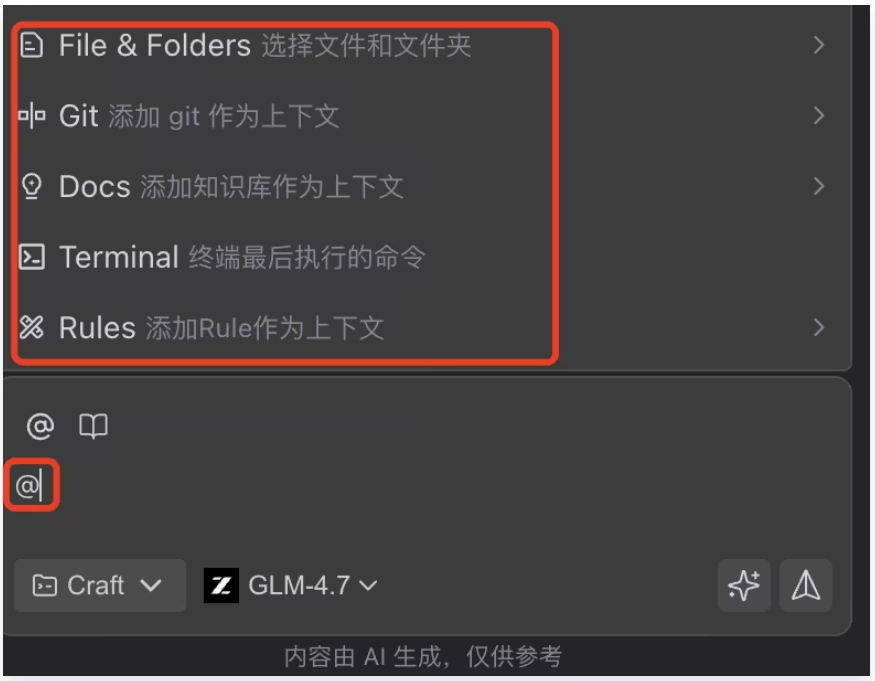


对话快捷指令

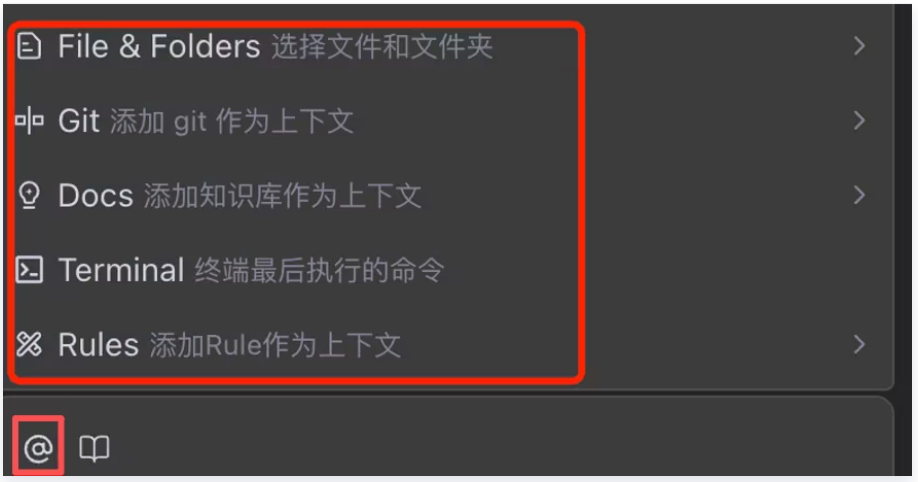
- 在对话输入框中，可输入 `/` 调用预置的快捷指令以及自定义指令。



- 在对话输入框中，可输入 `@` 选择添加文件、知识库等资料作为对话上下文。

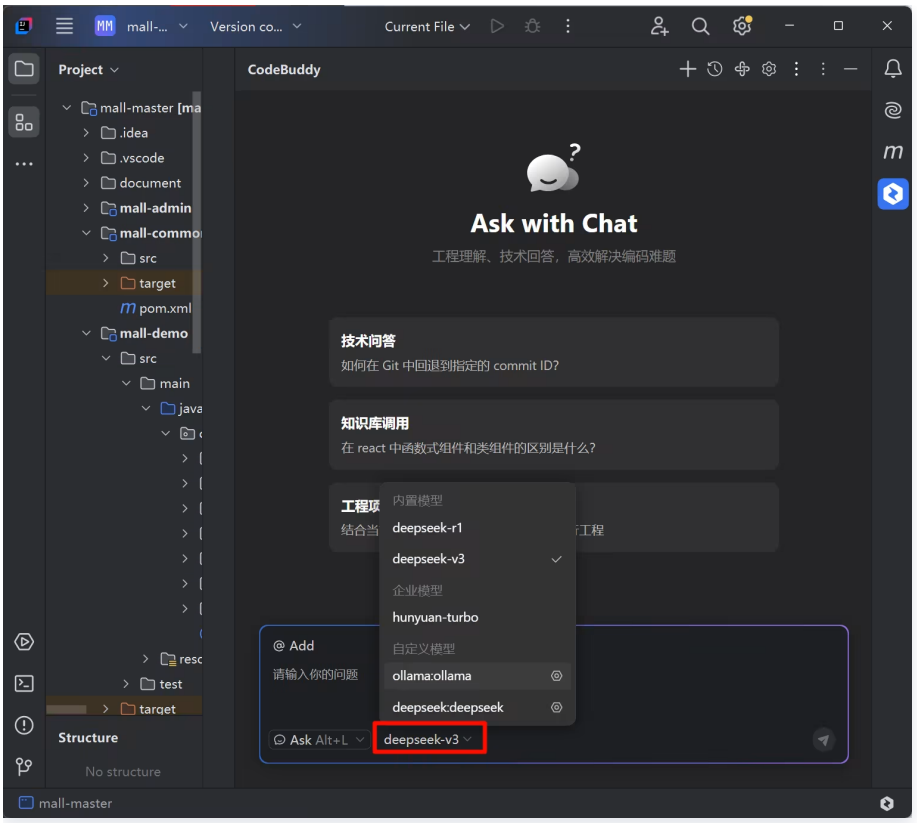


单击 @ 可以选择添加文件、知识库等资料作为上下文，除了支持搜索文件之外，其它的与在对话输入框中输入 @ 添加上下文一致。



模型切换

支持内置的对话模型，详情请参见 模型选择 。

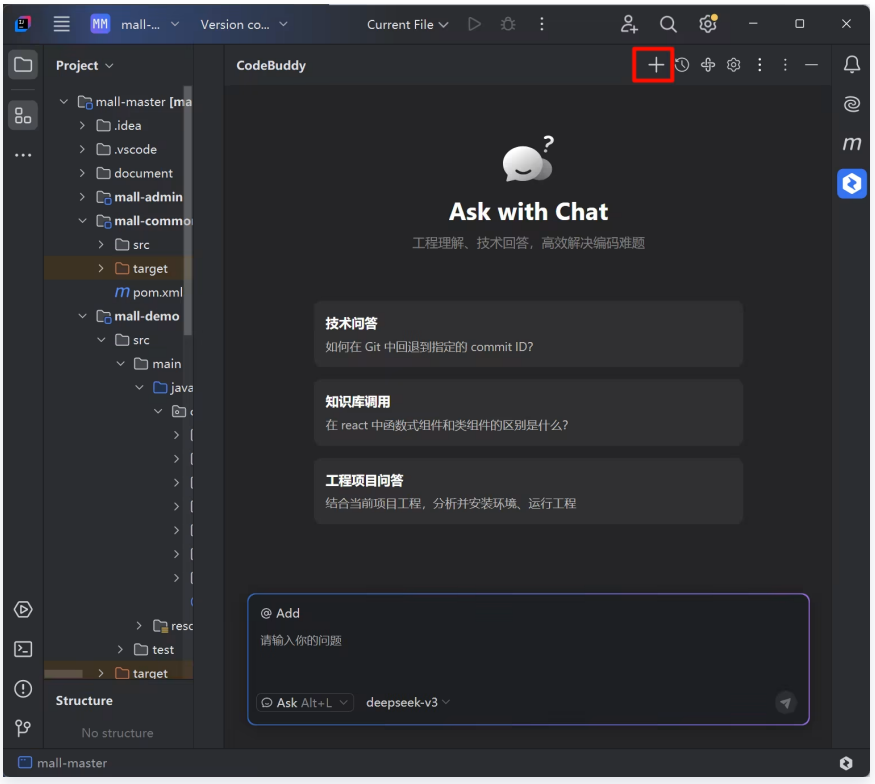


对话管理

对话管理包括创建新对话和历史对话记录管理。

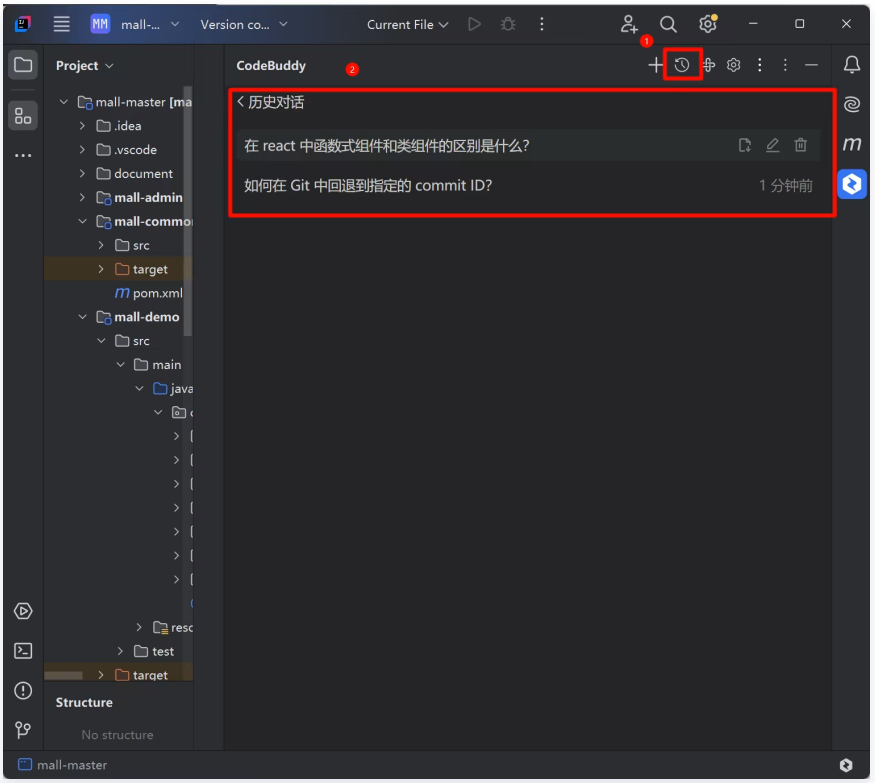
- 创建新对话

  如果您想重新开启对话或者开启一个新的技术话题，您可以单击**创建新对话**。



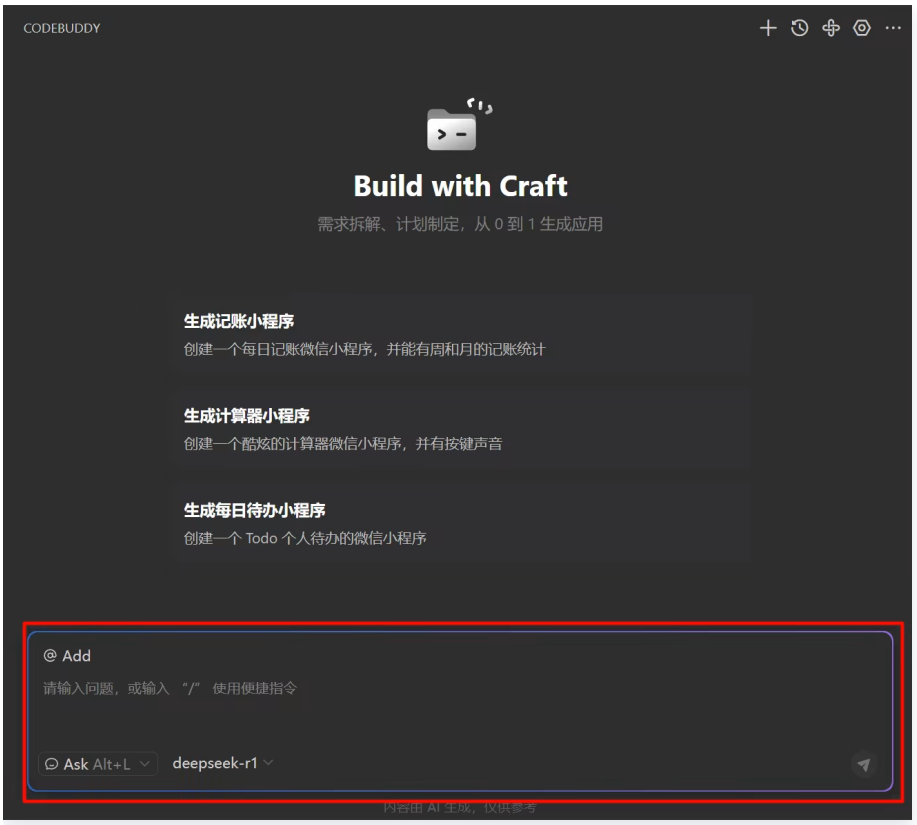
- 历史对话记录管理

  单击进入**历史对话记录**后，在这里，您可以查看历史对话记录，并支持对历史对话进行编辑、导出和删除操作。



微信开发者工具 IDE

微信开发者工具 IDE 的技术对话使用方式，与 VS Code 对齐。您可以在对话输入框中通过 `@` 和 `/` 符号发起对话场景的指令。



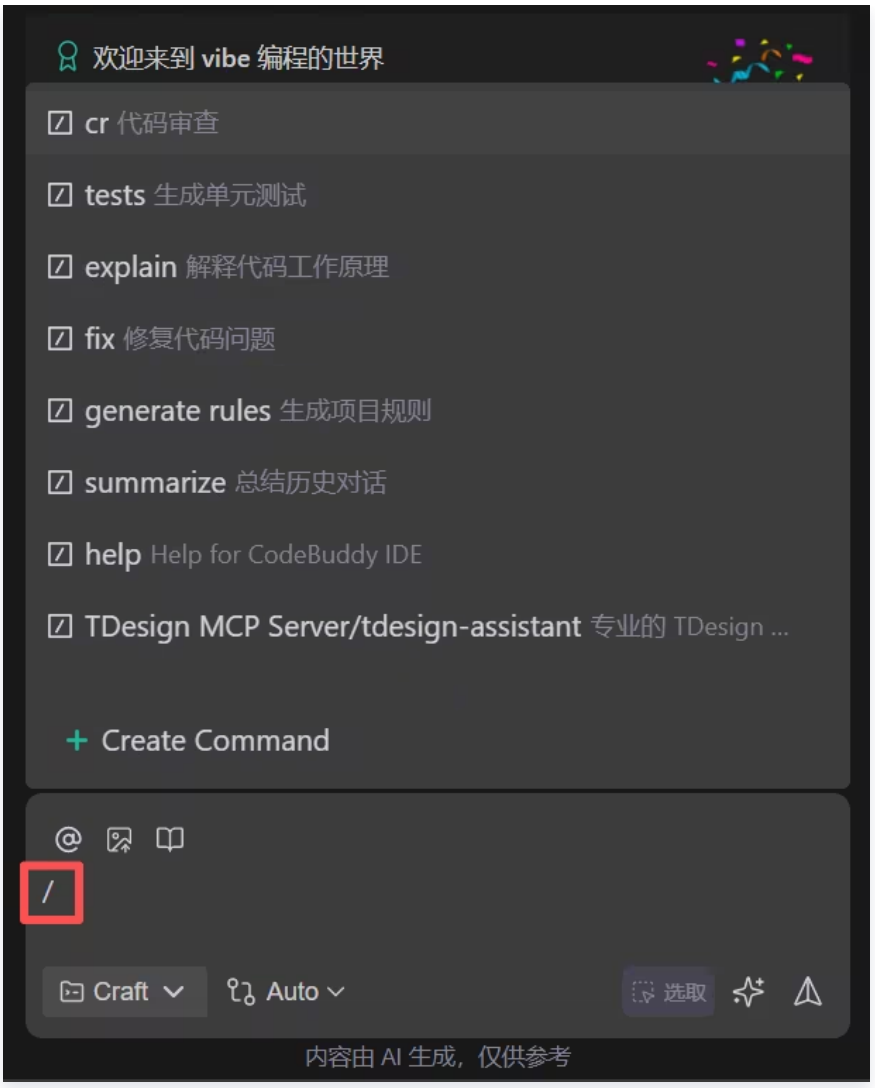
Ask 和 Craft 模式切换

代码助手在对话输入框中支持 Ask 和 Craft 两种模式的切换，满足不同开发场景的需求。如果您希望解决编码过程中的具体问题或快速生成片段代码，可以选择 Ask 模式；如果您希望实现从自然语言到生成完整应用的复杂需求，实现“需求到交付”的自动化，可以选择 Craft 模式。



对话快捷指令

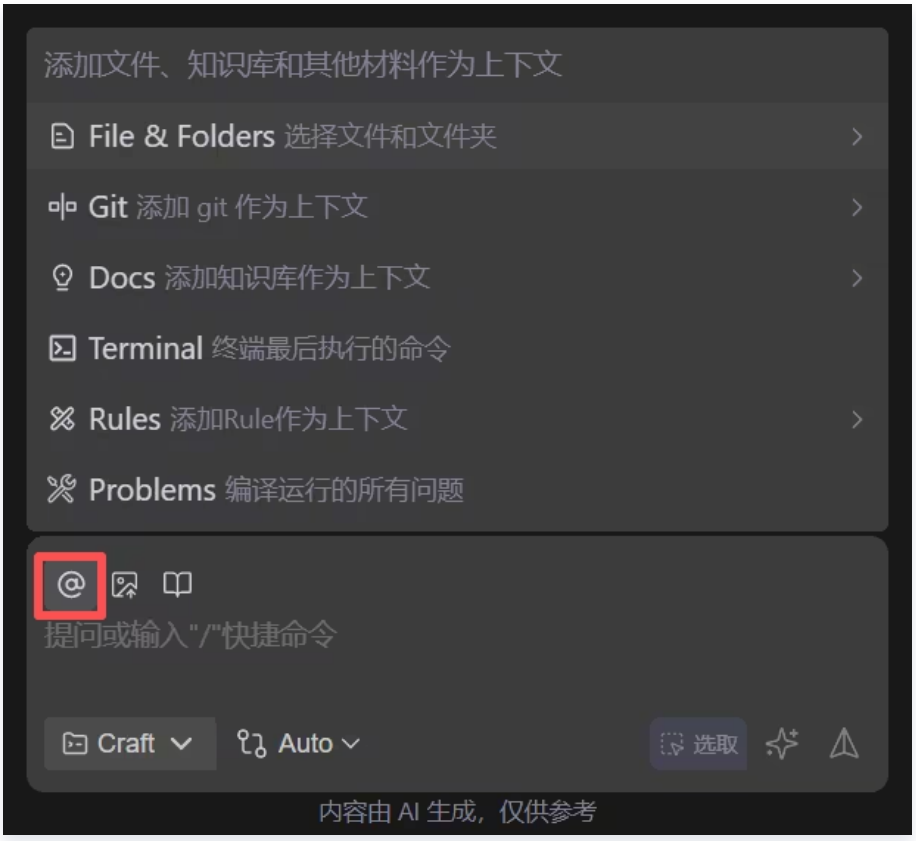
- 在对话输入框中，可输入 `/` 调用预置的快捷指令。



- **在对话输入框中，可输入 @ 选择添加文件、知识库等内容作为对话上下文。**

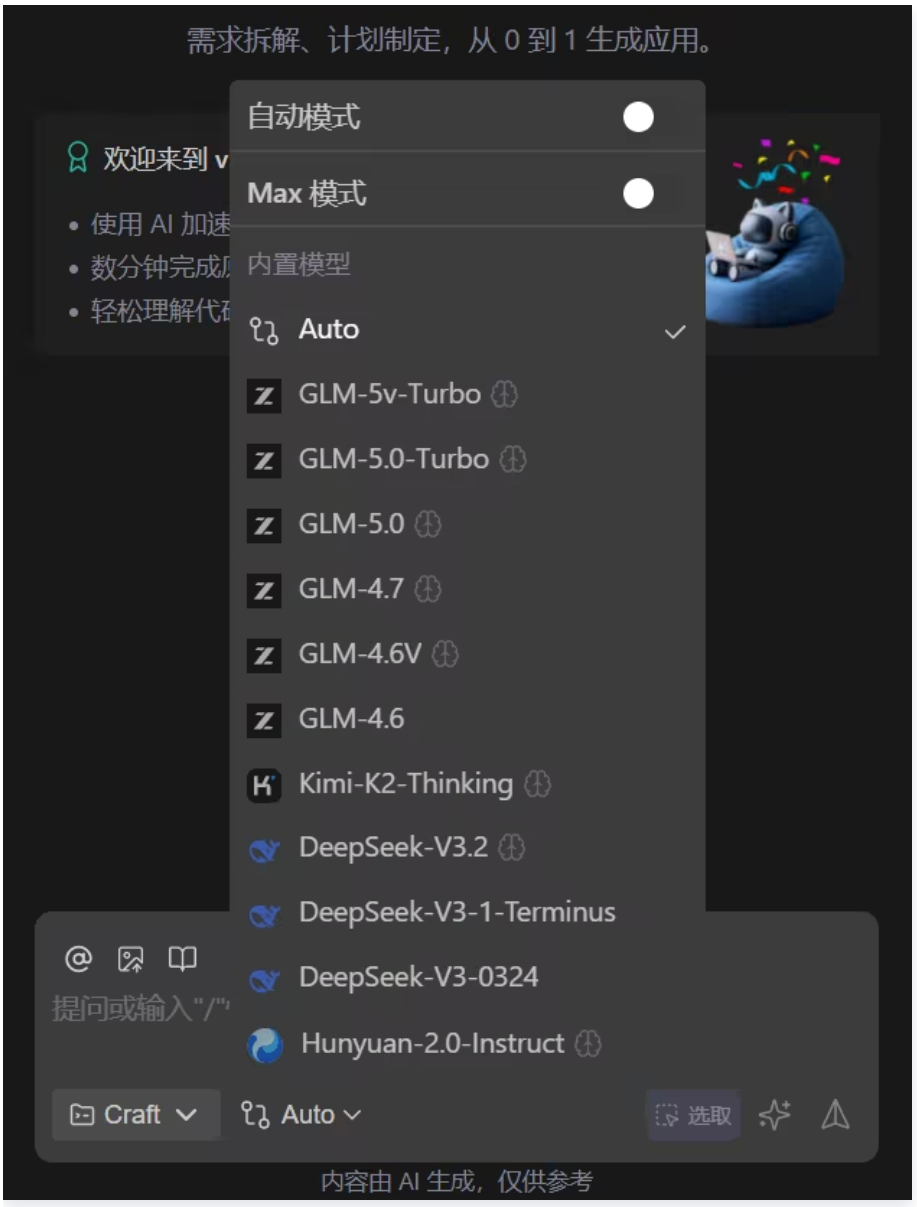


在对话框中，单击 `@` 可以选择添加文件、知识库等内容作为上下文，除了支持搜索之外，其它的与在对话输入框中输入 @ 添加上下文一致。



模型切换

支持内置模型和自定义模型，您可以根据实际需要自由切换。关于如何接入模型请参见 模型配置 。

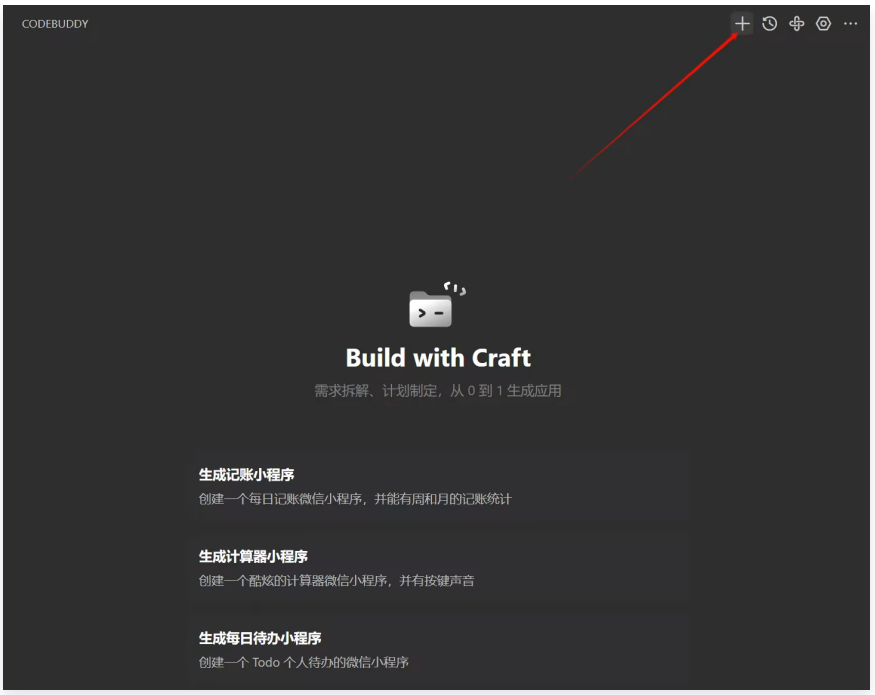


对话管理

对话管理包括创建新对话和历史对话记录管理。

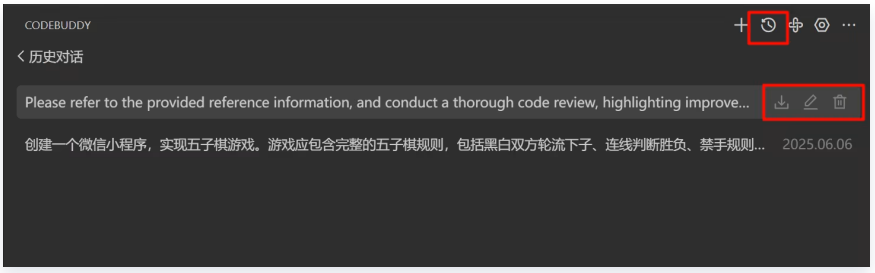
- 创建新对话

  如果您想重新开启对话或者开启一个新的技术话题，您可以单击 `Start new chat` 。



- **历史对话记录管理**

  单击进入 `Show chats` 后，在这里，您可以查看历史对话记录，并支持对历史对话进行编辑、导出和删除操作。



单元测试

最近更新时间：2026-07-20 19:49:31

腾讯云代码助手单元测试功能的定位是辅助用户全面、精细化地生成每个单元测试代码，并确保每个单元测试代码的正确性和高覆盖率，促使用户关注业务代码的正确性，以此提升整体代码库的质量。

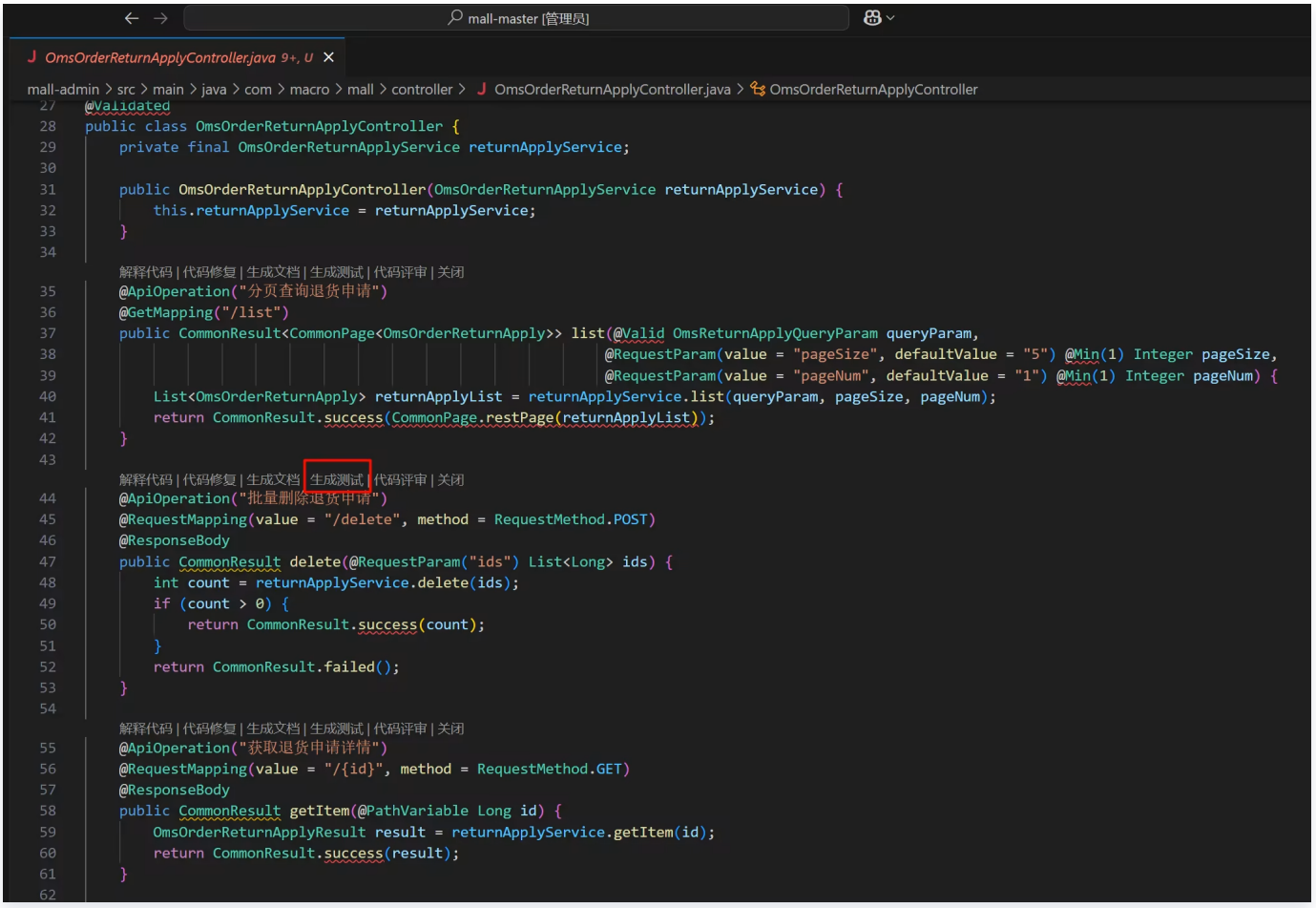
CodeBuddy 插件

VS Code

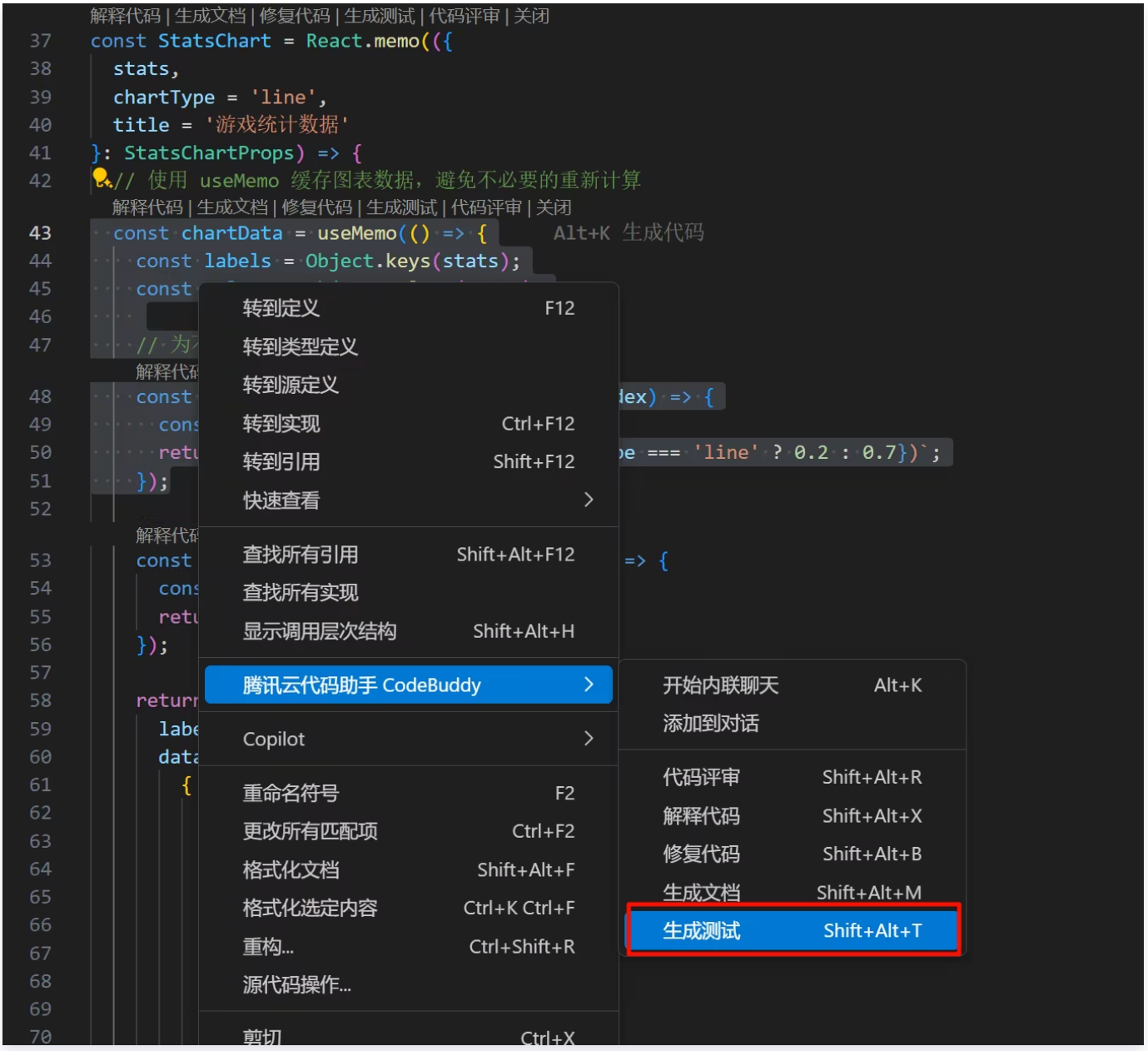
触发单元测试生成

方法级触发单元测试生成

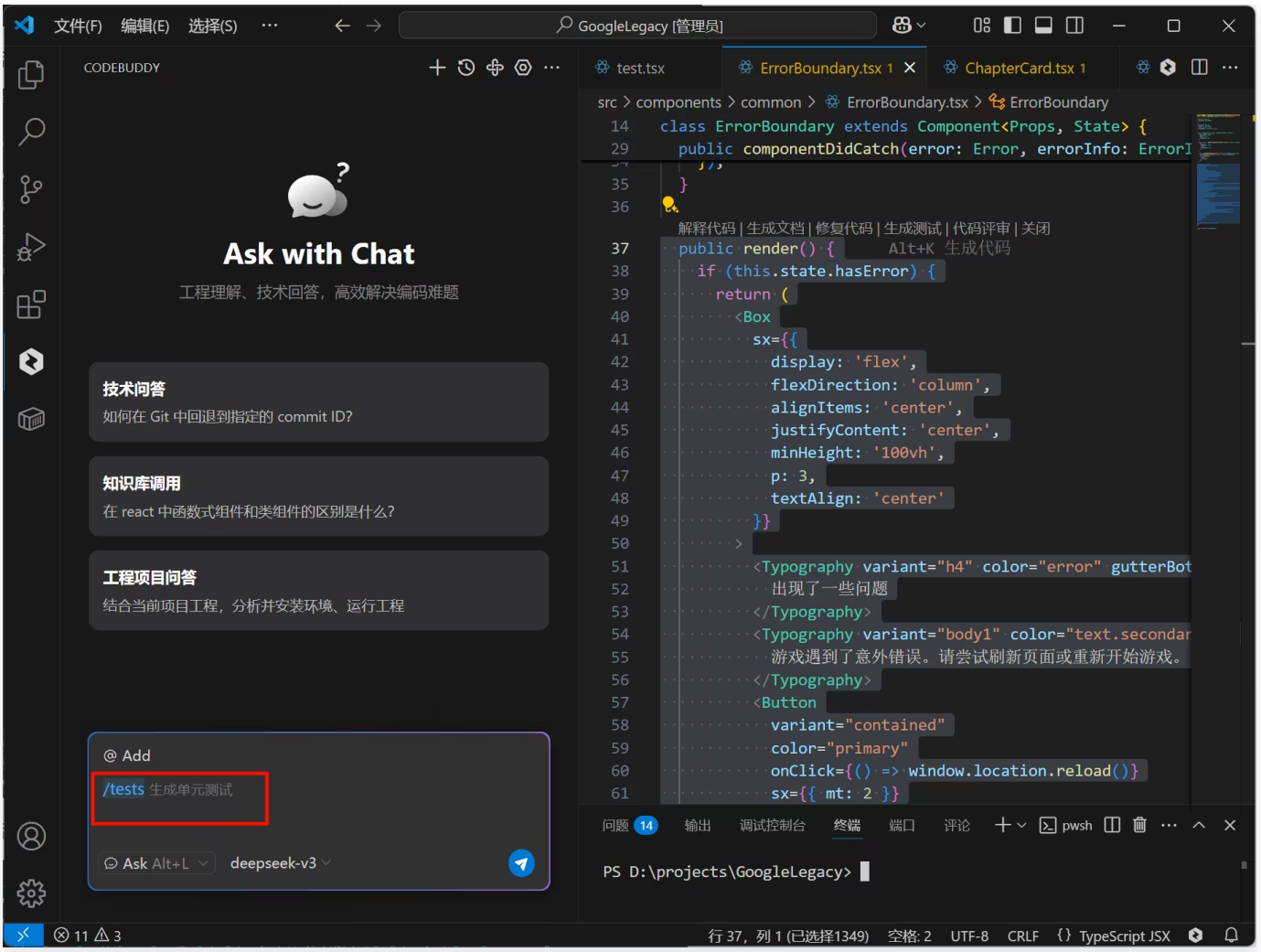
- 在打开的文件编辑区中，单击函数上方快捷功能按钮生成单元测试。



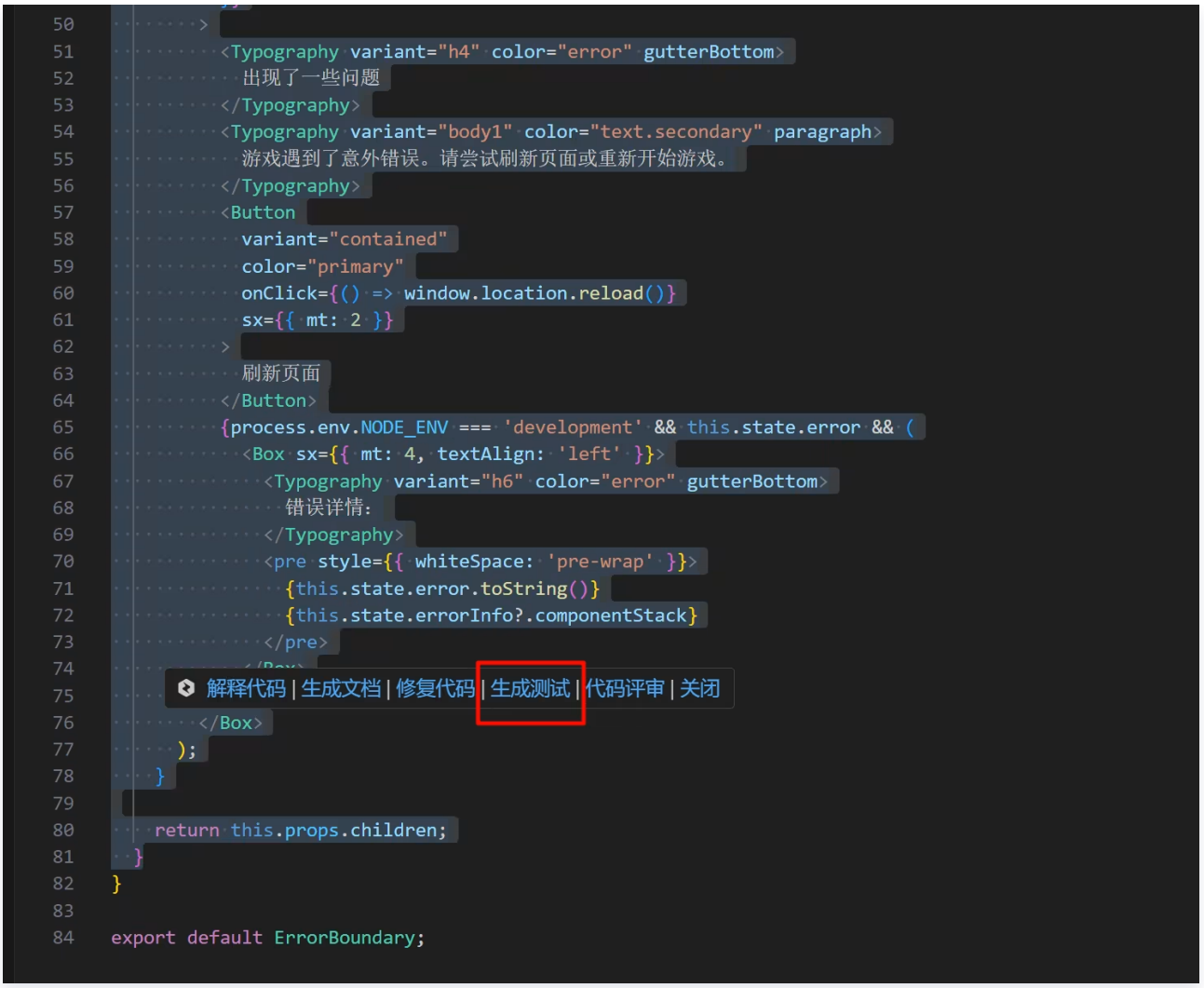
- 圈选代码后，右键选择**腾讯云代码助手 CodeBuddy > 生成测试**。



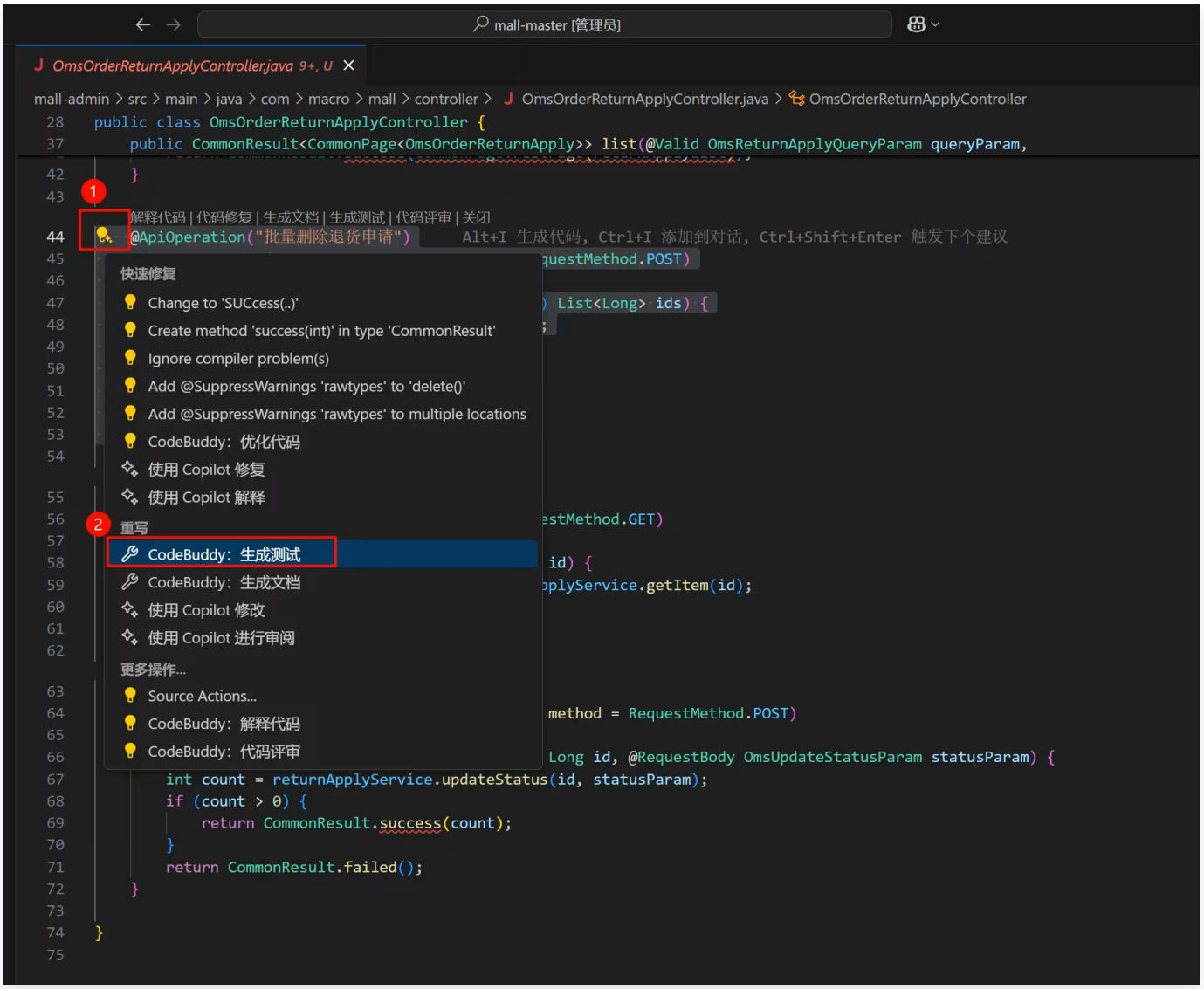
- **在左侧对话面板中，圈选代码区域后通过 `/tests` 指令生成单元测试。**



- **选中代码后，将鼠标悬浮圈选代码后出现功能面板，选择生成测试。**

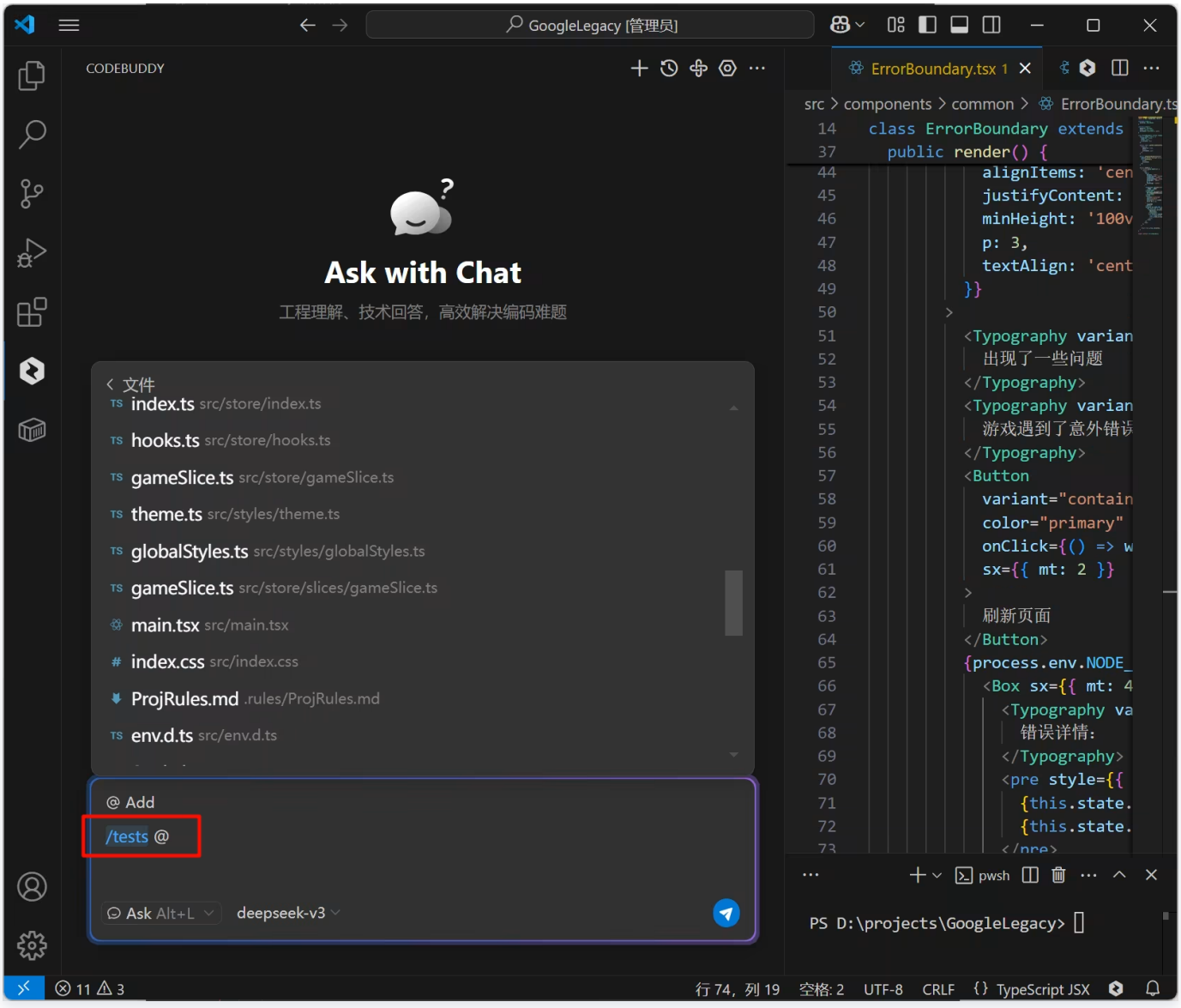


- 圈选代码后，单击显示操作的小灯泡，选择 **CodeBuddy：生成测试**。

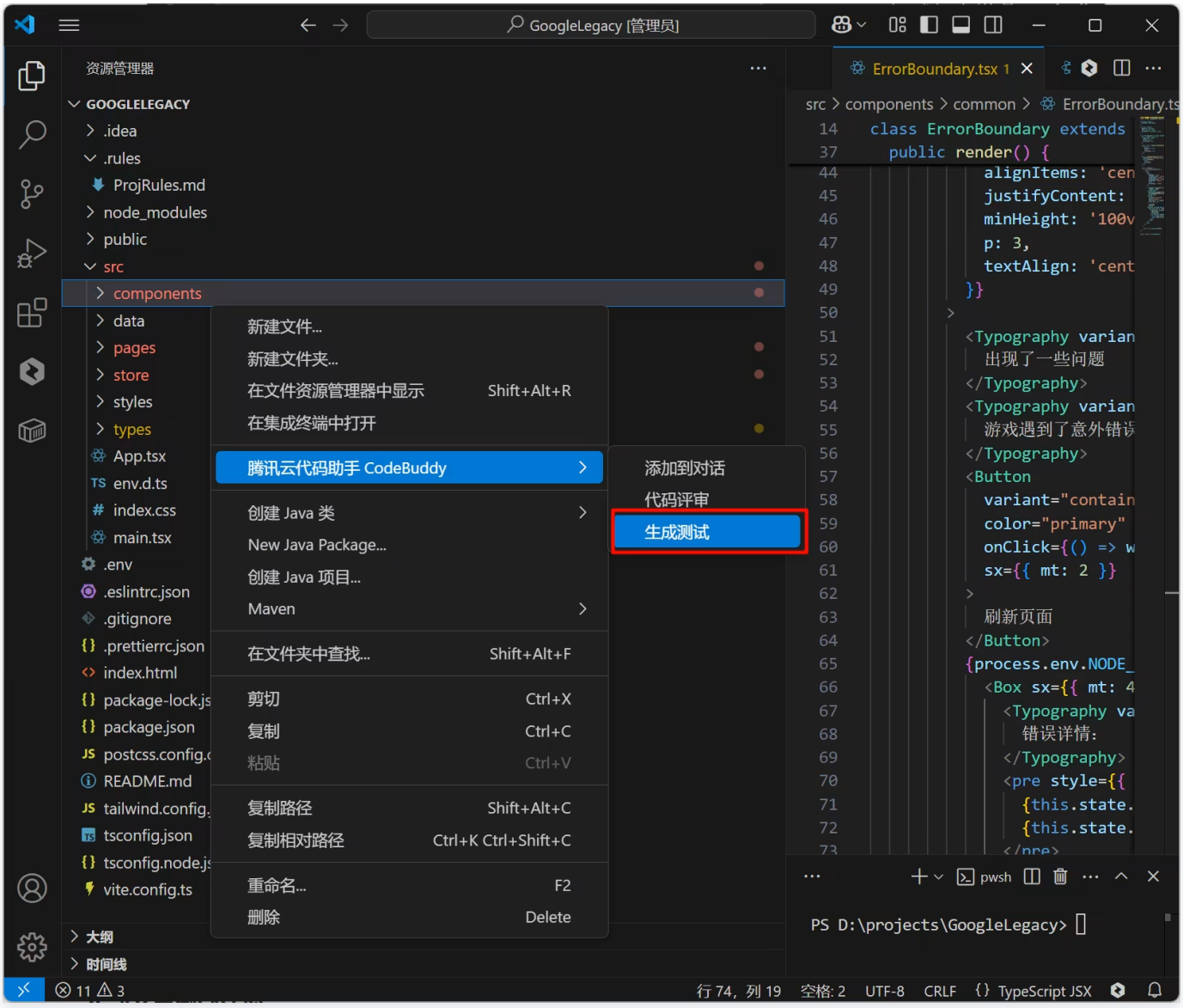


文件/目录级触发单元测试生成

- 在对话输入框中，输入 `/tests` 指令，然后通过 `@ Add` 添加单个或多个文件或目录。



- **在资源管理器的文件树中，可以右键选择 CodeBuddy > 生成测试。**



生成单元测试代码

1. 触发单元测试生成后，代码助手会先分析项目结构、查找现有的测试文件、确认测试目录等一系列操作。可以点击**继续**，让代码助手继续生成单元测试代码。



2. 基于对所需测试代码内容的分析后，生成了完整的单元测试代码。此外，还提供了测试说明，以及注意事项等内容，辅助进行测试。



3. 应用单元测试代码。

   生成单元测试代码后，您可以根据实际情况来选择不同的应用代码方式，支持**插入到 IDE**、**复制**、**新建文件**以及**保存到本地**的方式。应用代码后，可以根据测试说明来运行测试验证。

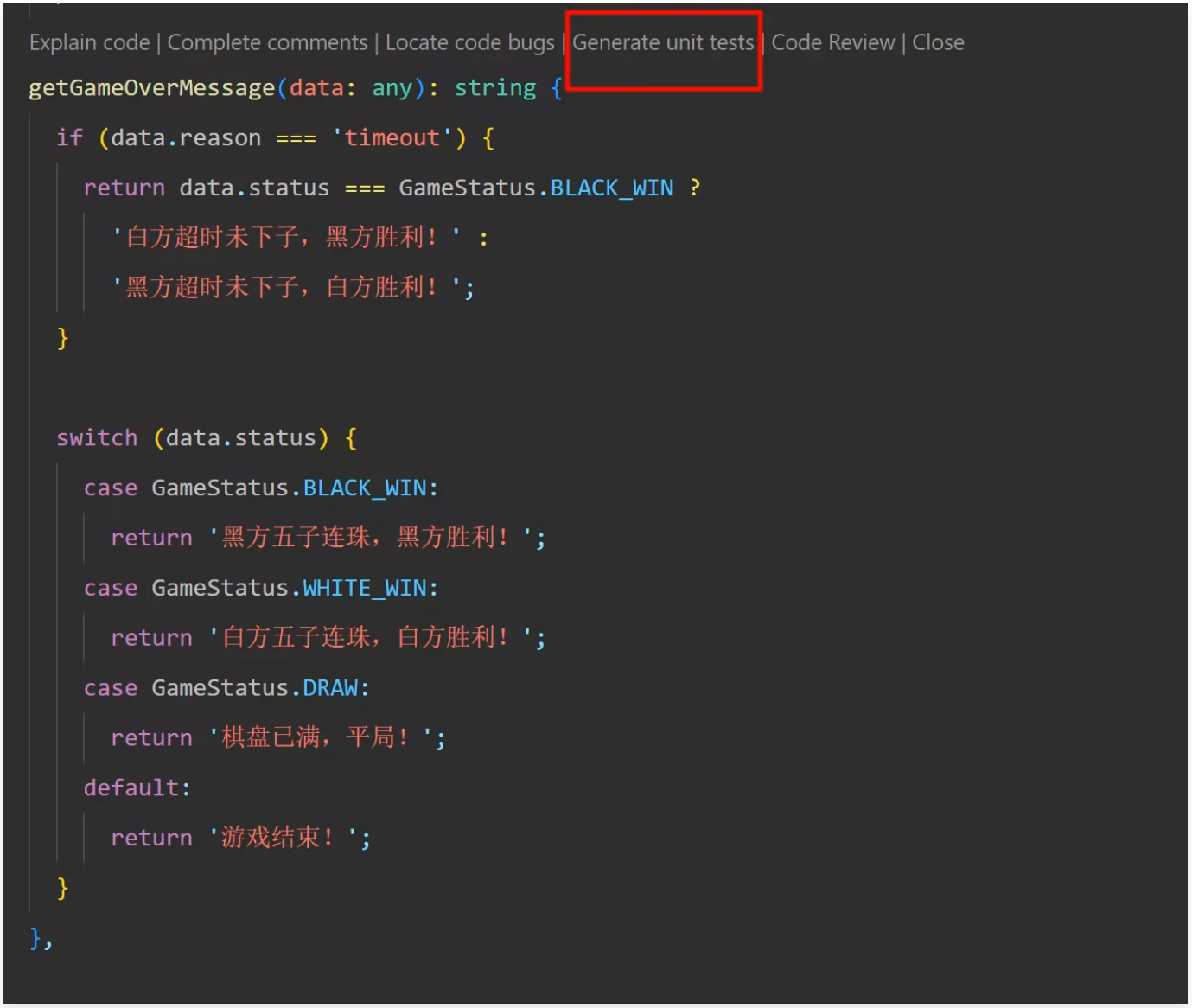
微信开发者工具 IDE

您可以根据方法级或文件/目录级选择多种触发方式触发单元测试生成。

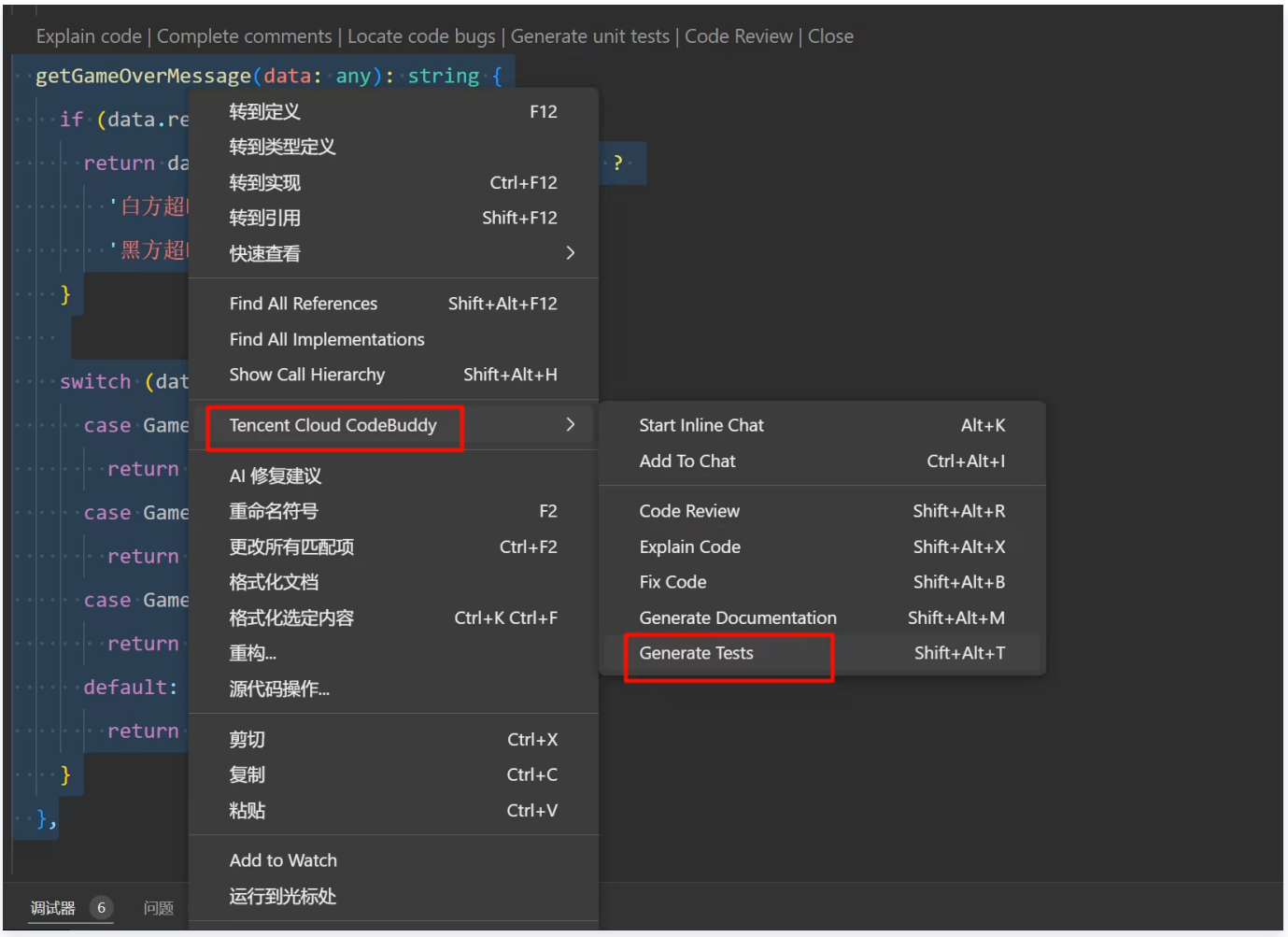
触发单元测试生成

方法级触发单元测试

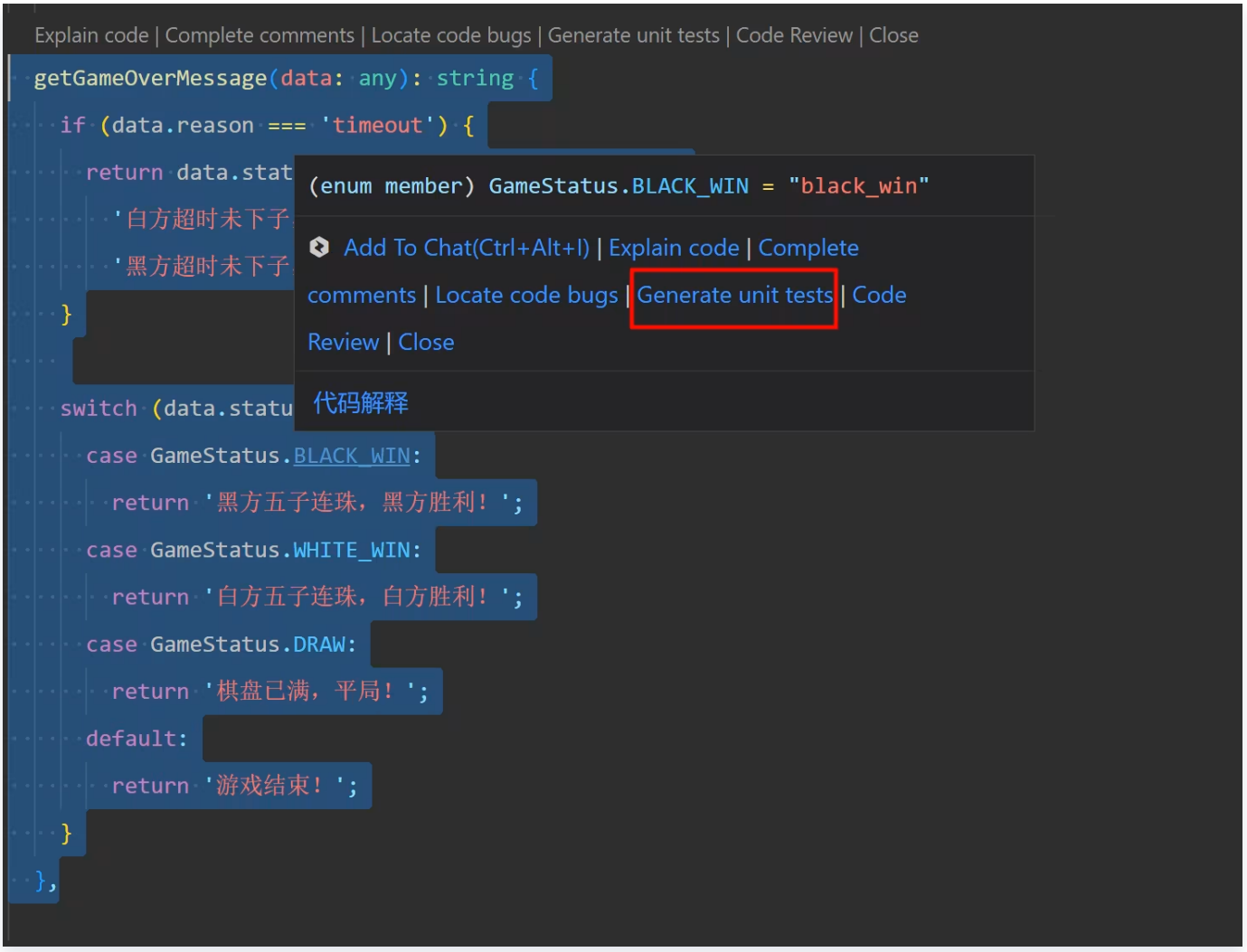
- 在打开的文件编辑区中，可以单击函数或方法前的快捷方式触发单元测试生成。



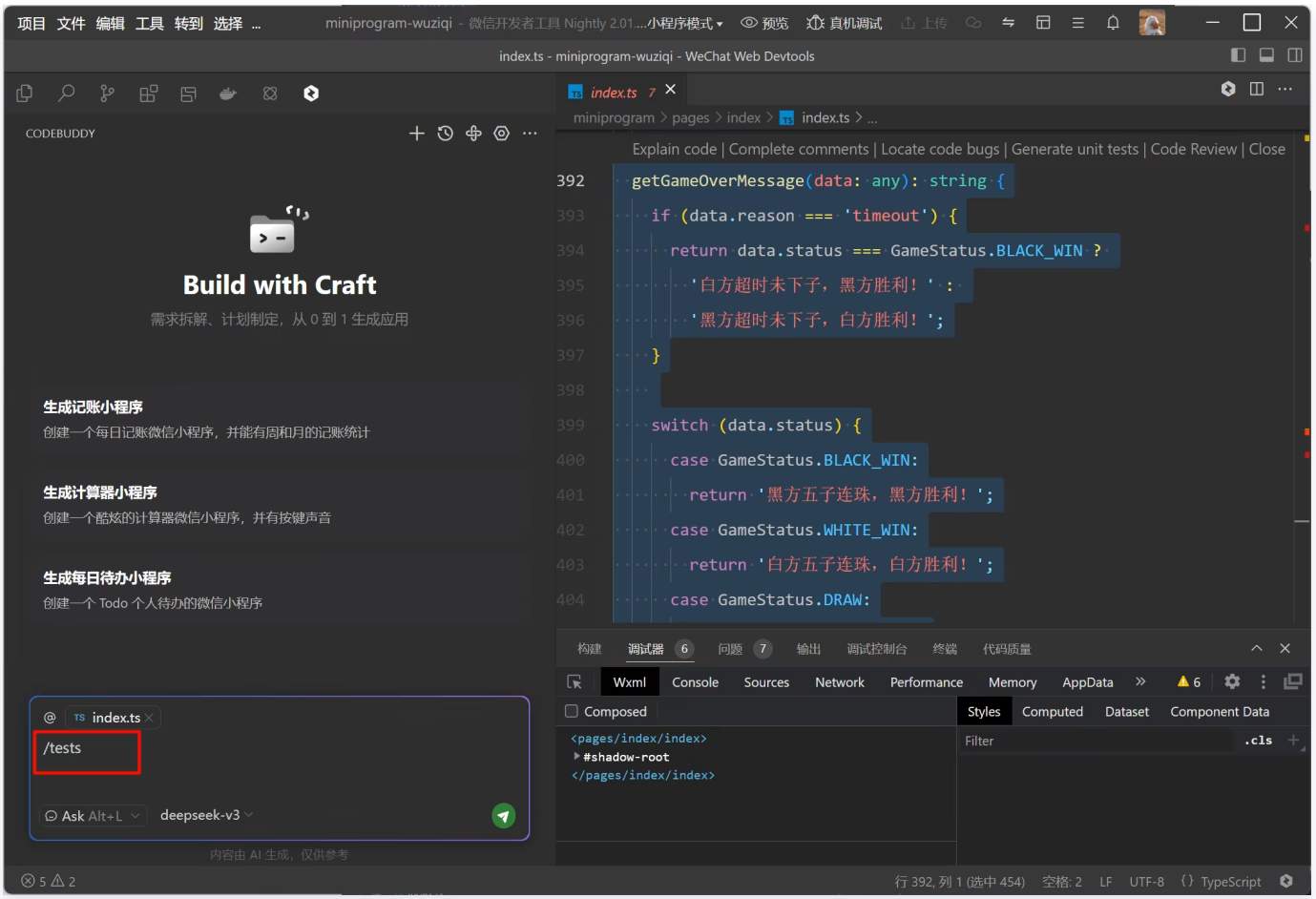
- 圈选代码后，可以右键选择 **Tencent Cloud CodeBuddy > Generate tests** 触发单元测试生成 。



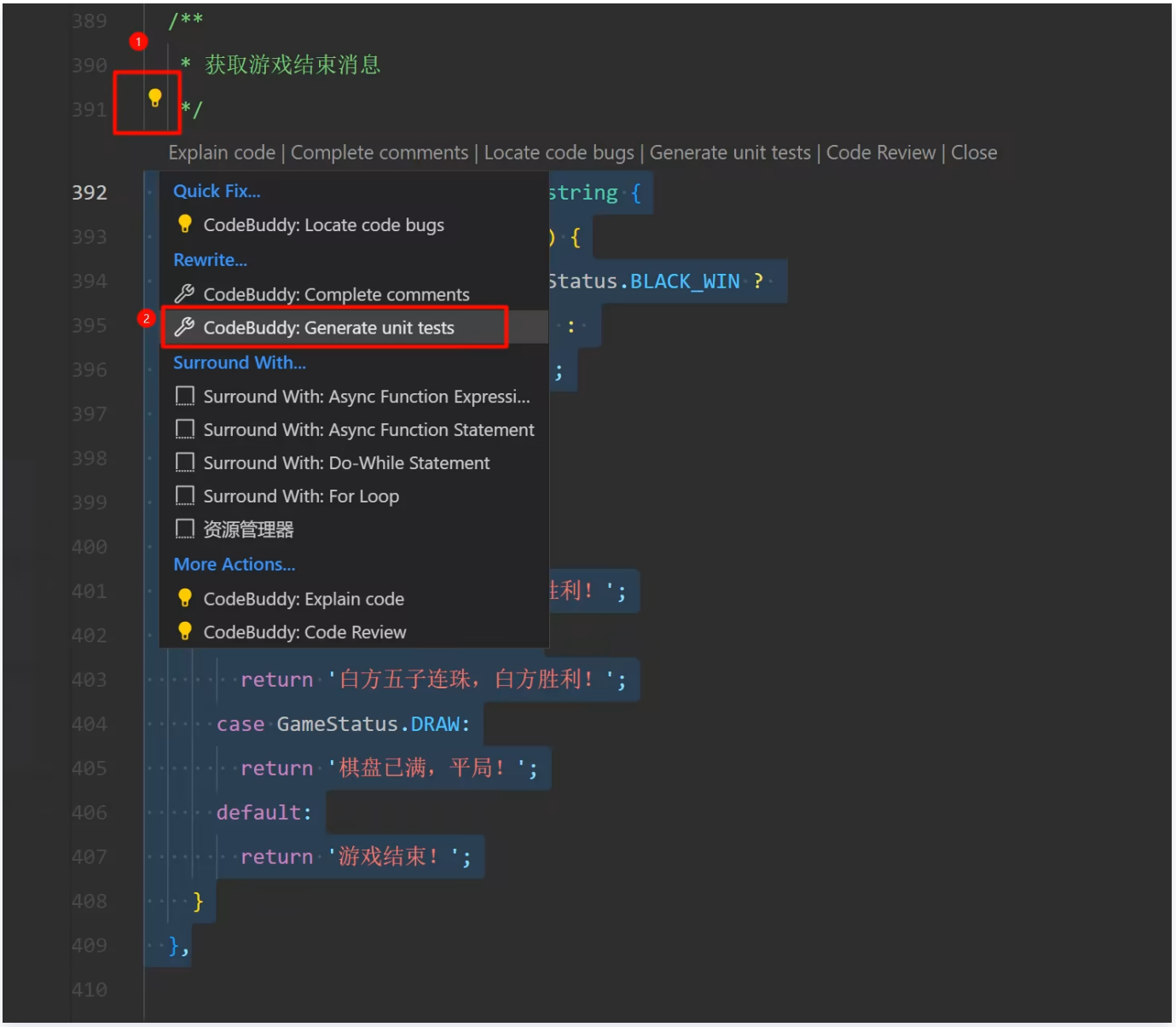
- 圈选代码后，将鼠标悬浮在代码上，在弹出的悬浮菜单中可以选择 **Generate unit tests** 触发单元测试生成。



- 圈选代码后，在左侧对话框中，可以通过 `/tests` 指令触发单元测试生成。

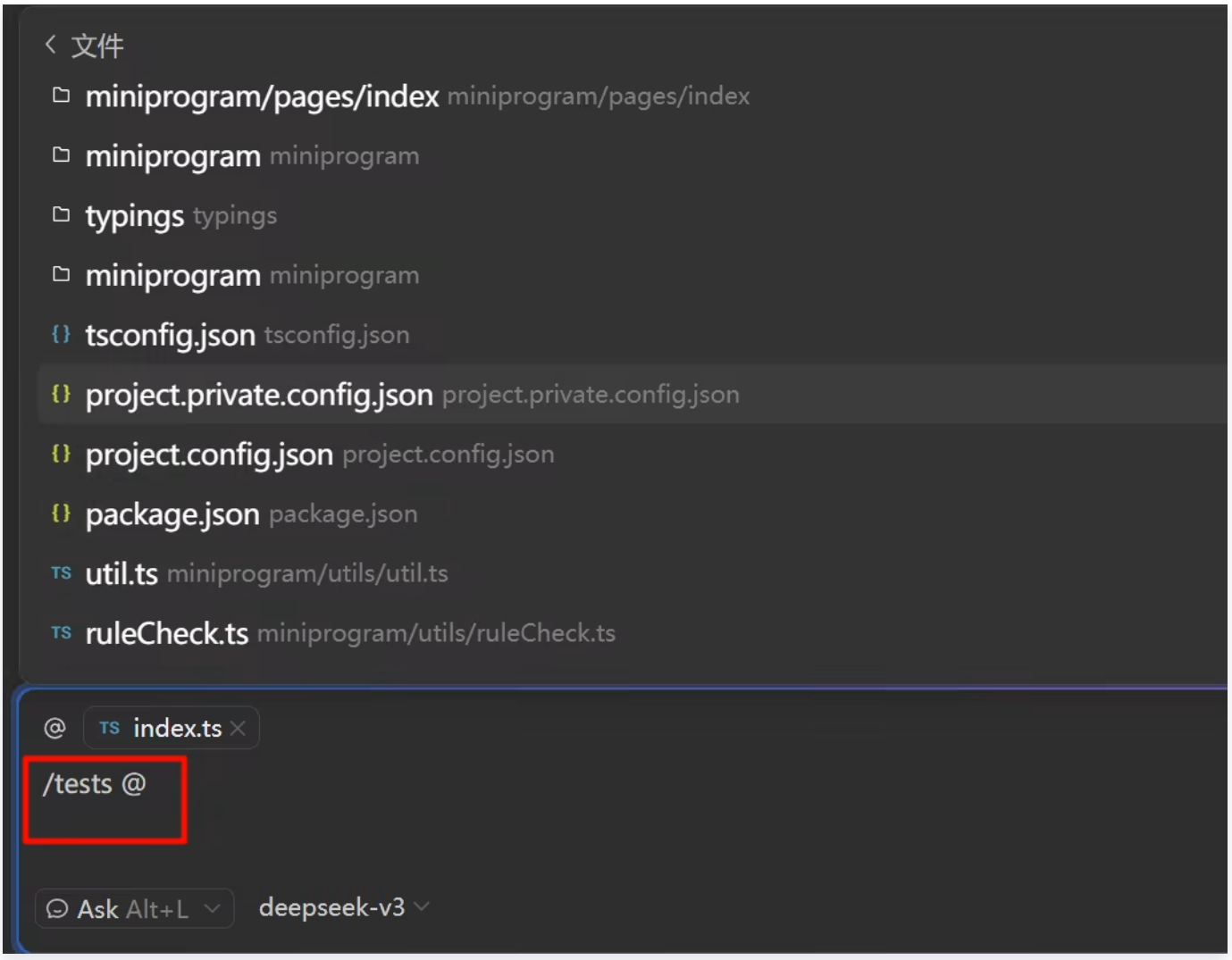


- 圈选代码后，会弹出一个小灯泡，单击后在弹出菜单中可以选择 **CodeBuddy:Generate unit tests** 触发单元测试生成。

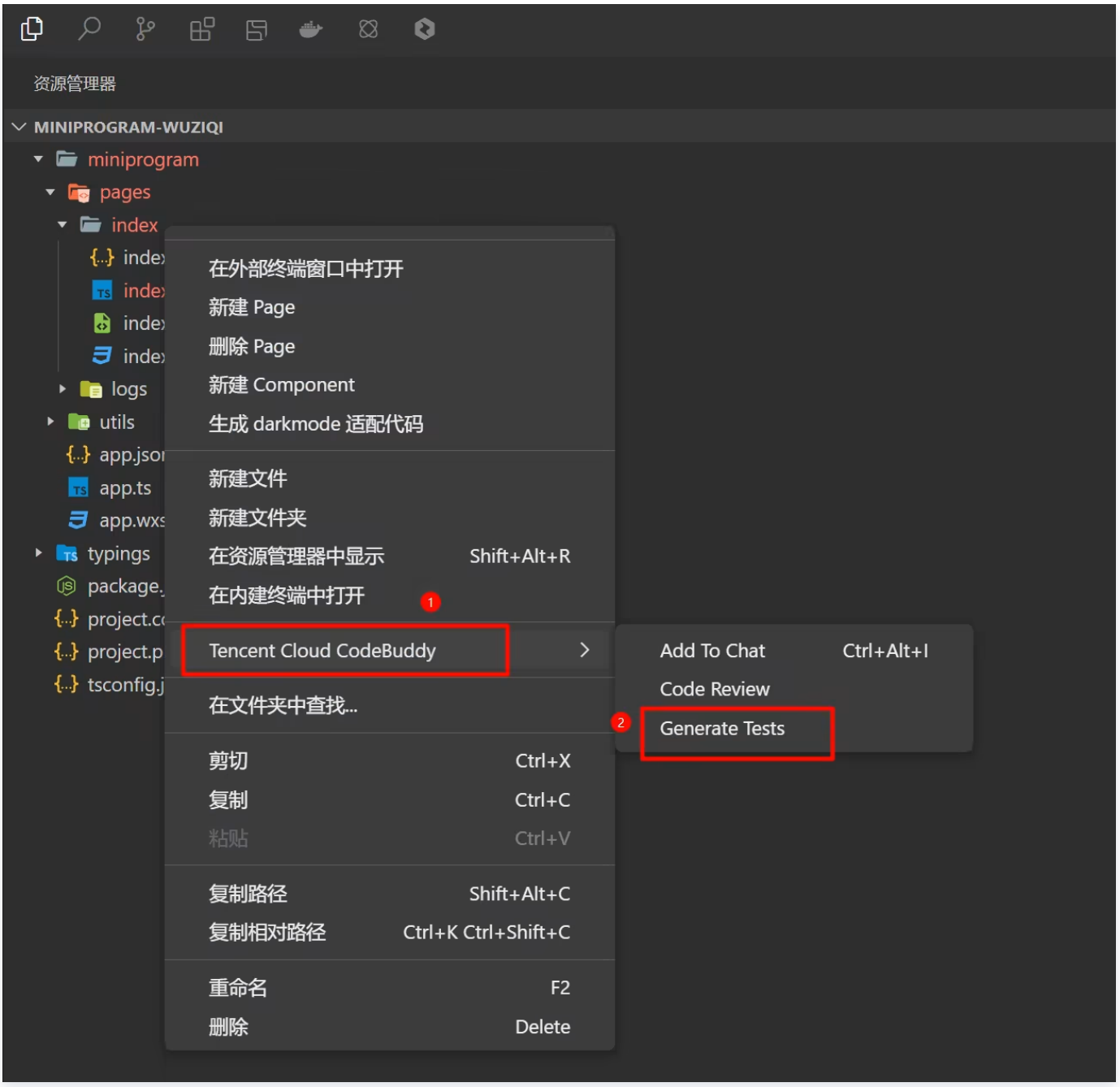


文件/目录级触发单元测试生成

- 在对话输入框中，您可以直接输入 `/tests` 指令，然后通过 `@ Add` 添加单个或多个文件 / 目录。

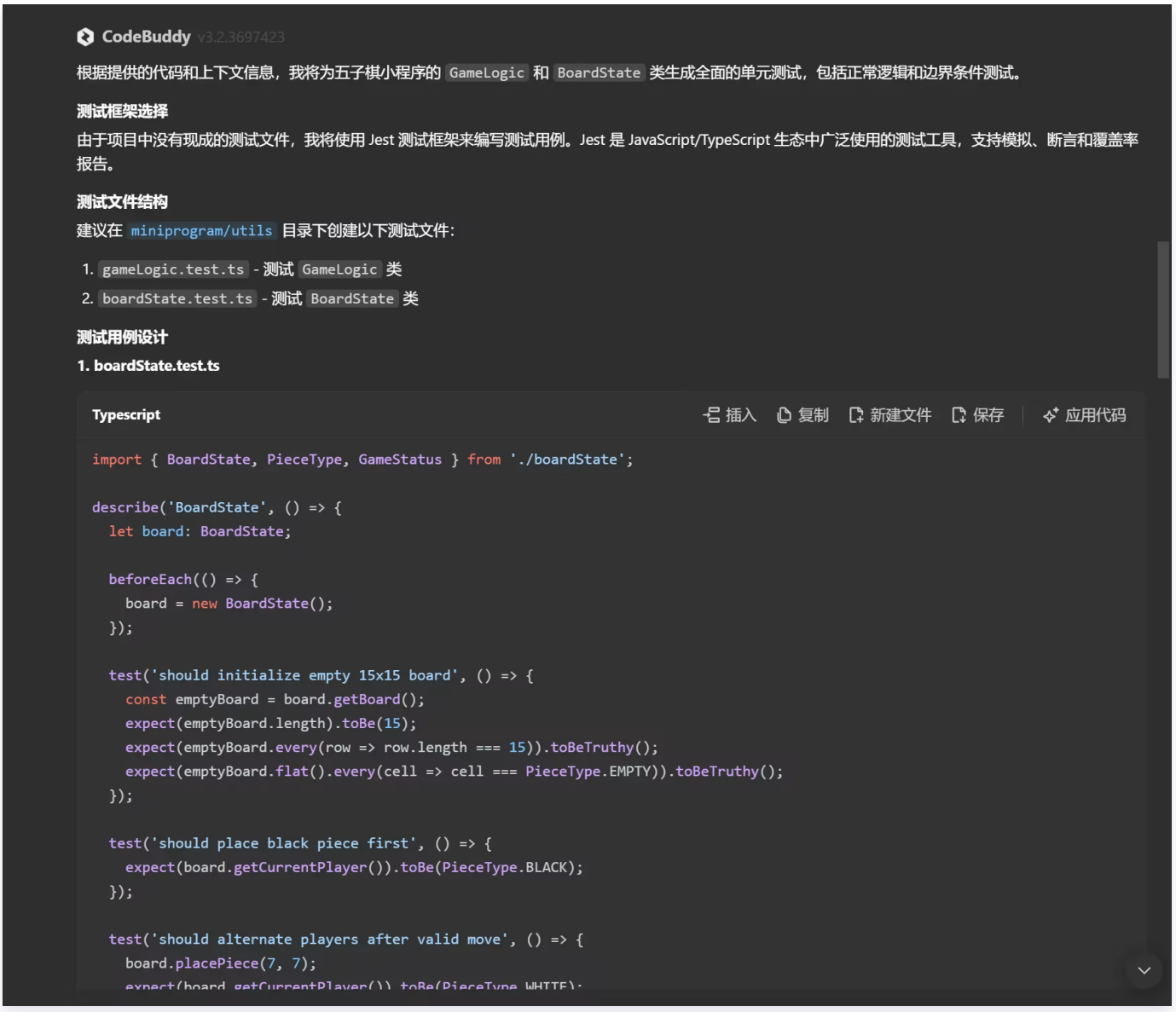


- 在文件树中，可以右键单击文件 / 目录，然后选择 **Tencent Cloud CodeBuddy > Generate Tests** 触发单元测试生成。

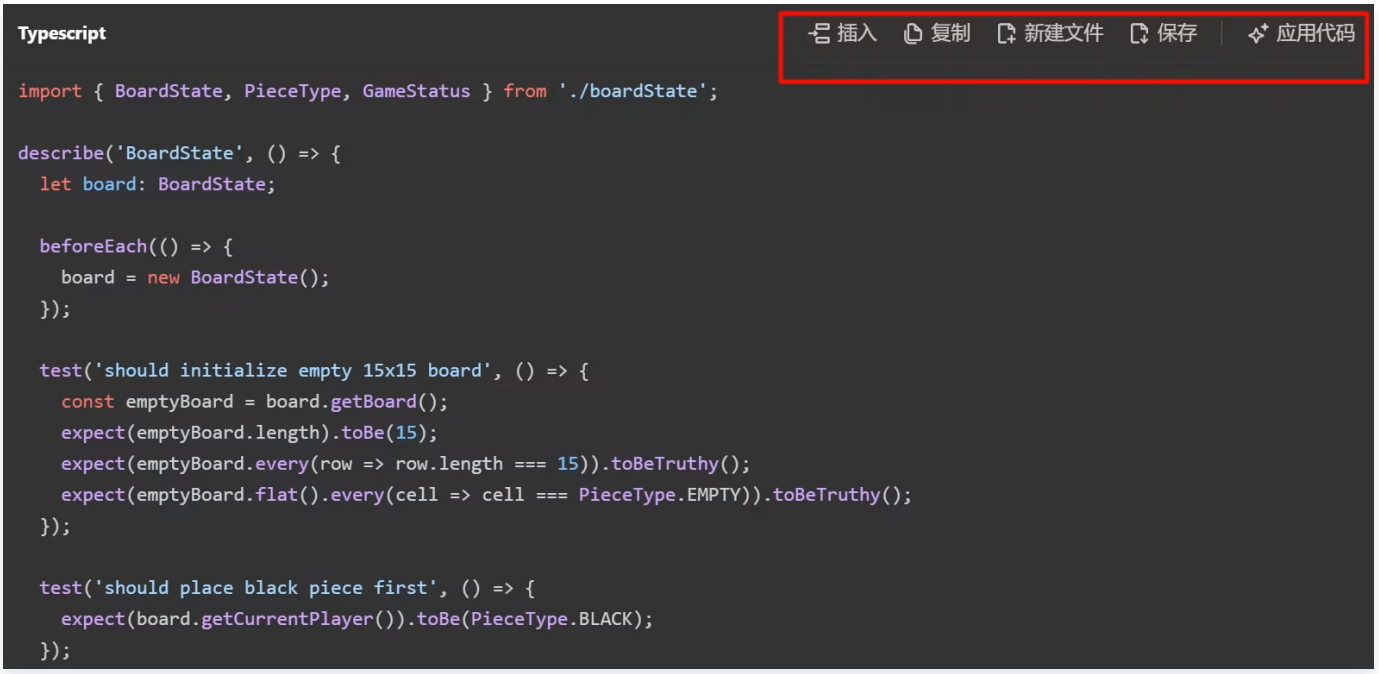


生成单元测试代码

1. 触发单元测试生成后，代码助手会先对测试代码内容和项目结构进行分析，并选择合适的测试框架生成单元测试代码。同时，也生成了测试覆盖范围、执行建议等内容，辅助进行测试。



2. 生成单元测试代码后，您可以根据实际情况来选择不同的应用代码方式，支持**插入**到 IDE、**复制**、**新建文件**以及**保存**到本地、**应用代码**的方式。



CodeBuddy IDE

触发单元测试生成

方法级触发单元测试生成

圈选方法或函数的代码块后，在弹出菜单中单击选择 **Add to Chat** 按钮，或者直接使用菜单右侧提示的快捷键，将代码添加到对话中触发单元测试生成。

```
// 更新物品
update(deltaTime, cameraX, screenWidth, groundLevel) {
    // 更新金币
    this.coins.forEach(coin => coin.update(deltaTime));

    // 更新障碍物
    this.obstacles.forEach(obstacle => obstacle.update(deltaTime));

    // 移除屏幕外的物品
    this.coins = this.coins.filter(coin =>
        !coin.collected && coin.x > cameraX - 100
    );
    this.obstacles = this.obstacles.filter(obstacle =>
        obstacle.x > cameraX - 100
    );

    // 生成新物品
    while (this.lastItemX < cameraX + screenWidth * 1.5) {
        this.generateNextItems(groundLevel);
    }
}
```

Add to Chat Ctrl + L Edit Ctrl + I

文件/目录级触发单元测试生成

- 在对话框中，通过 @ 符号将文件/目录添加到对话上下文，触发单元测试生成。



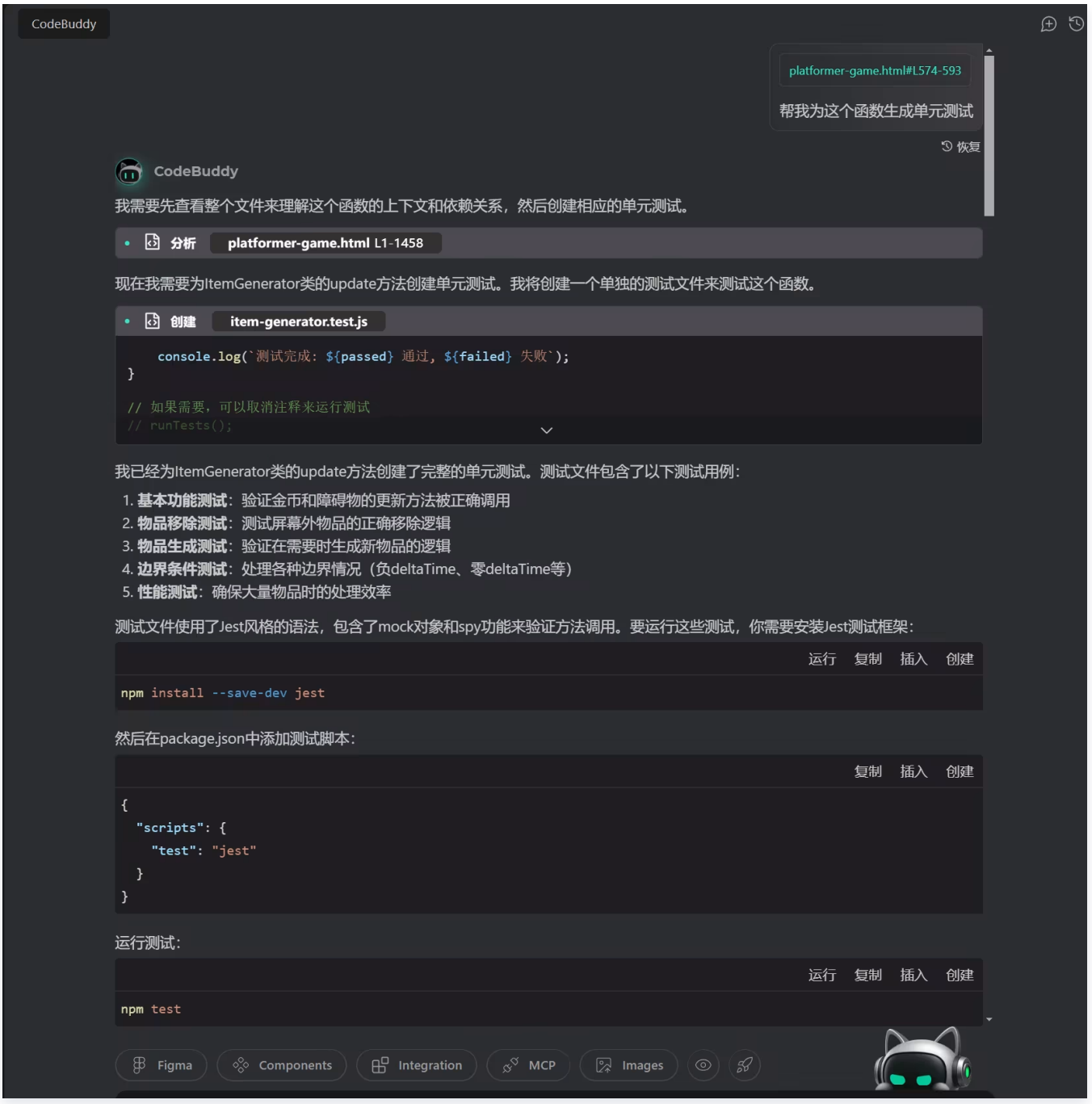
- 在文件树中，右键单击文件/目录，在菜单中选择 **添加到 CodeBuddy 对话**，将文件/目录添加到对话中，触发单元测试生成。



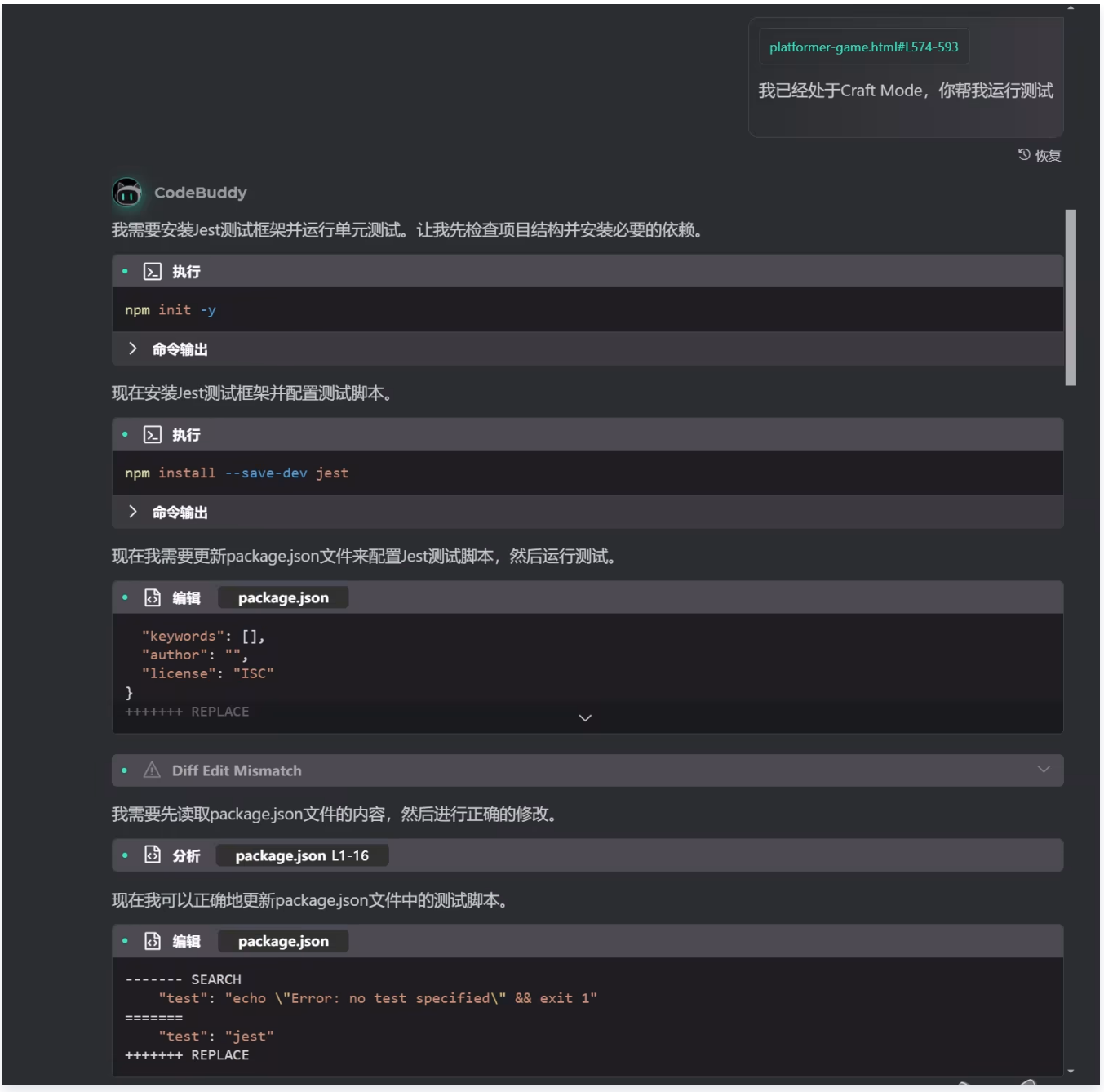
生成单元测试代码

如果您希望 Agent 生成单元测试代码并自动改写文件、运行测试，可以选择使用 Craft 模式；如果希望 Agent 只输出单元测试代码及方案，自己手动应用单元测试代码并运行测试，可以选择使用 Chat 模式。下面以 Craft 模式进行介绍。

1. 触发单元测试生成后，Agent 会先读取整个文件的内容来理解所需单元测试的上下文和依赖关系，然后创建完整的单元测试文件，单元测试文件覆盖了多个测试用例。并且，给出了运行测试的方法。



2. 创建单元测试后，如果希望 Agent 自动运行测试，可以提醒 Agent，Agent 将会自动安装测试框架、配置测试脚本、运行测试。



3. 运行测试会覆盖所有测试用例，如果某个测试用例失败，Agent 会自动修复，直到所有测试用例都验证成功。


现在再次运行测试来验证所有测试用例是否通过。
执行
npm test
命令输出
Continue Response
Continue
完美！所有测试用例都已通过。我已经成功为ItemGenerator类的update方法创建了完整的单元测试，包括：
1. 基本功能测试：验证金币和障碍物的更新方法被正确调用
2. 物品移除测试：测试屏幕外物品的正确移除逻辑
3. 物品生成测试：验证在需要时生成新物品的逻辑
4. 边界条件测试：处理各种边界情况（负deltaTime、零deltaTime等）
5. 性能测试：确保大量物品时的处理效率
所有11个测试用例都已通过，覆盖了update方法的所有主要功能。测试使用了Jest框架，包含了mock对象和spy功能来验证方法调用。
测试文件已经配置好，可以通过 npm test 命令运行所有测试。

代码评审

最近更新时间：2026-07-20 19:49:31

腾讯云代码助手推出**智能评审辅助**的新功能。这项功能旨在为 VS Code 和 JetBrains IDEs 上的开发者们提供全面支持，包括 **AI 辅助自评审**和 **AI 提交信息生成**两大核心功能。通过这些功能，开发者能够在项目开发过程中及时发现并解决本地代码变更可能引入的问题，从而提高代码质量，加速开发流程。

AI 辅助自评审

AI 辅助自评审功能，旨在提高 IDE 下的代码自我审查的效率和质量。在 Chat 模式中使用，可以通过 `/cr` 指令直接触发本地自动化代码审查（AICR）。

触发评审

腾讯云代码助手支持三种评审范围：**评审文件或文件夹（指定单个或多个文件或文件夹）**、**评审变更（当前暂存区代码变更）**、**评审编辑区（指定编辑区内容）**。

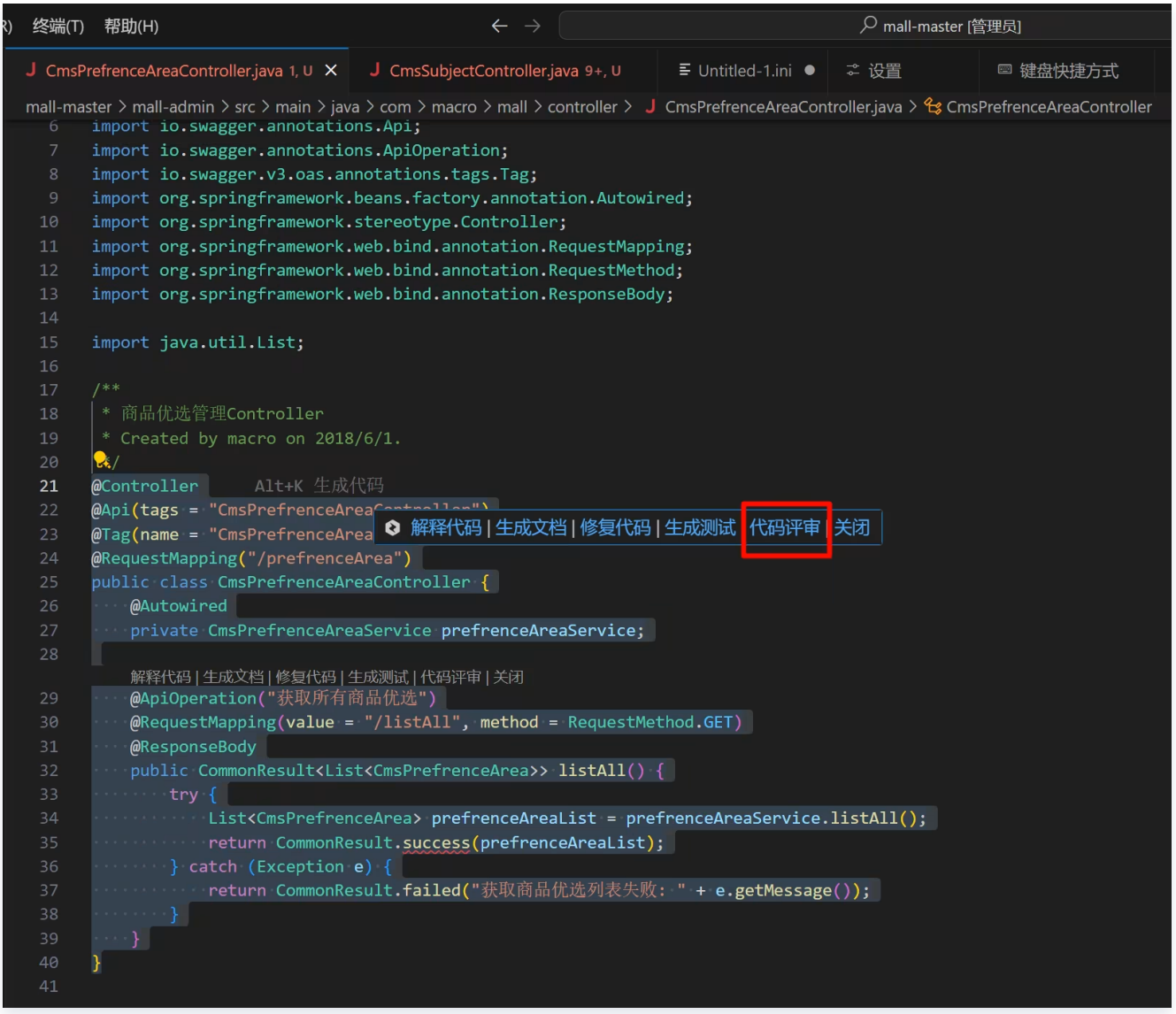
评审编辑区

在打开的代码文件编辑区中，可以通过圈选代码区域触发 **IDE 编辑区范围内**的代码作为评审内容。

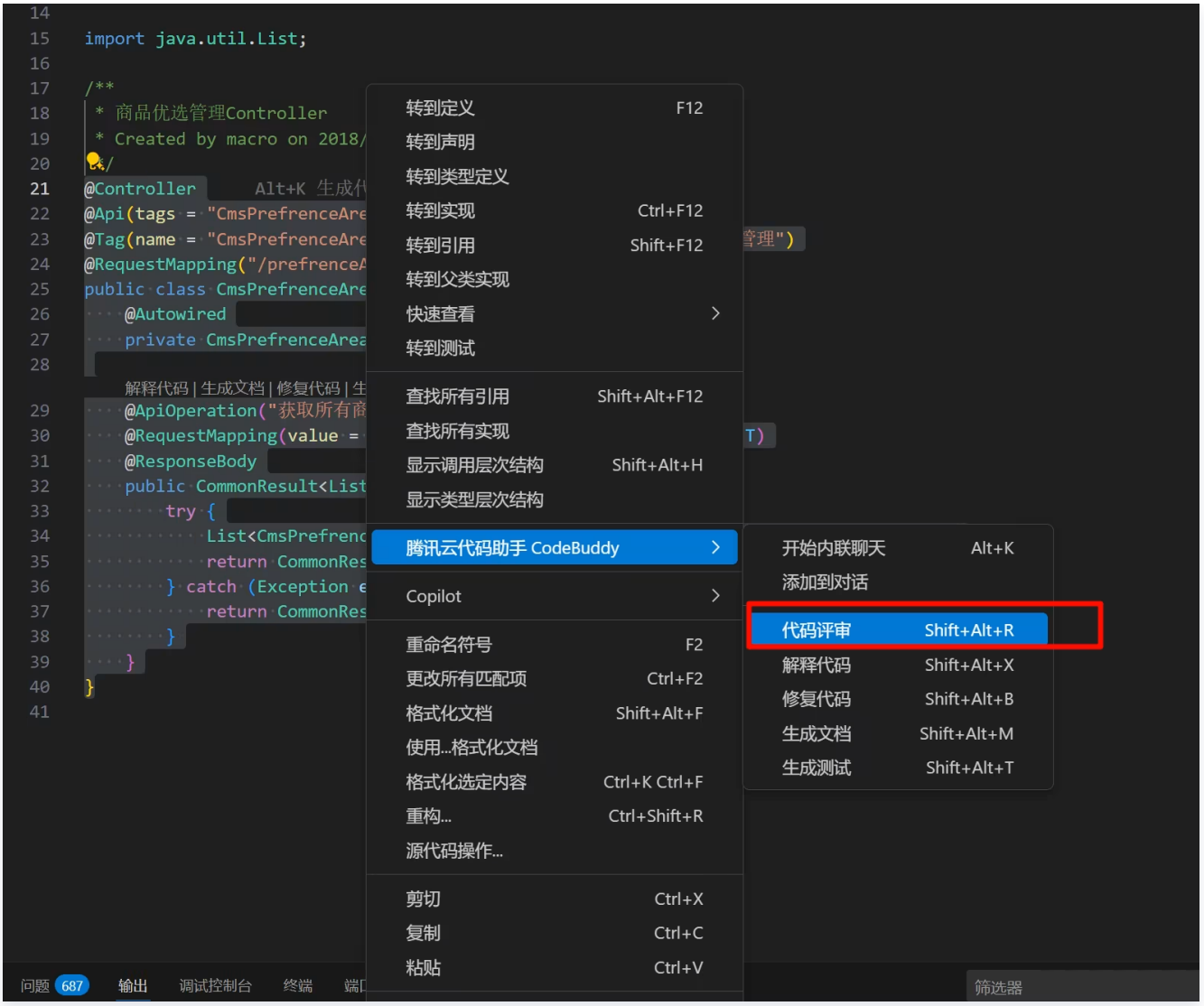
> **说明：**
> 如果在编辑区进行了划词圈选操作，触发评审功能后会将划词圈选部分的代码作为评审范围。如果未进行划词圈选，触发评审功能后会将整个工程文件的代码作为评审范围。

- 圈选代码区域后，将鼠标悬浮在所选代码区域上会出现功能面板，可以选择**代码评审**触发 CR 自动评审功能。

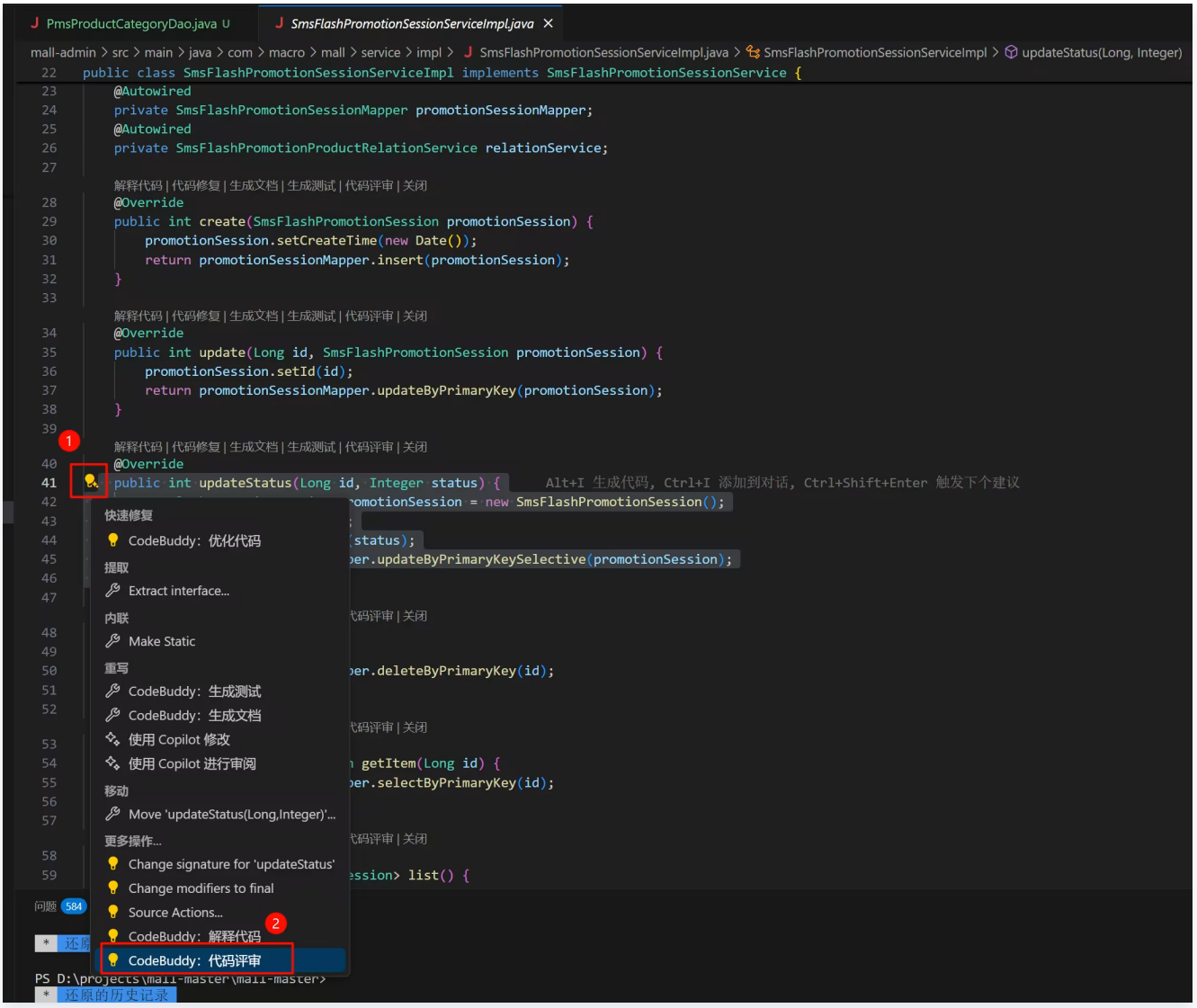
版权所有：腾讯云计算（北京）有限责任公司



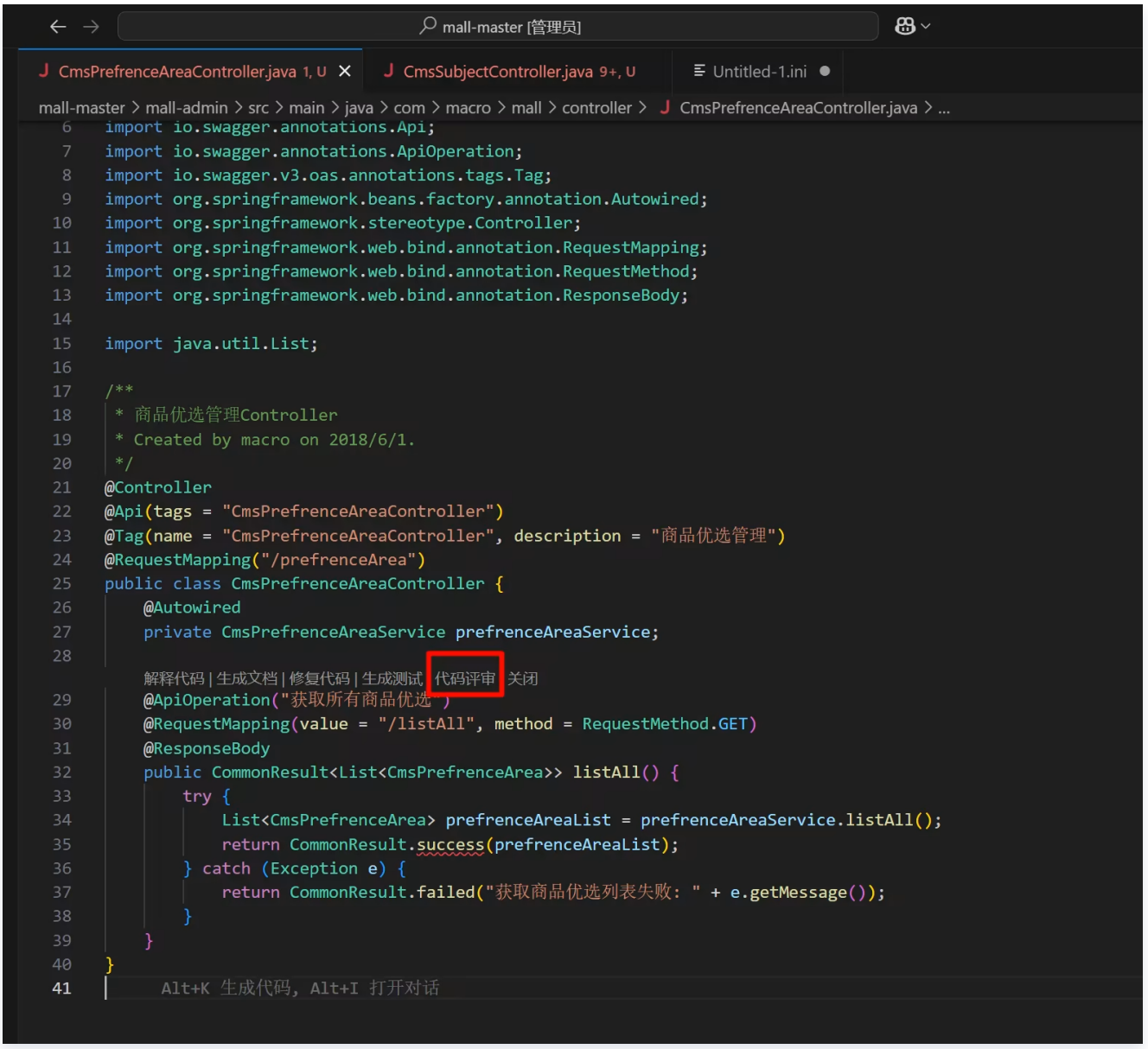
- 圈选代码区域后，也可以右键选择 **腾讯云代码助手 CodeBuddy > 代码评审**来触发评审功能。



- **圈选代码区域后，也可以单击代码操作的小灯泡来触发评审功能。**



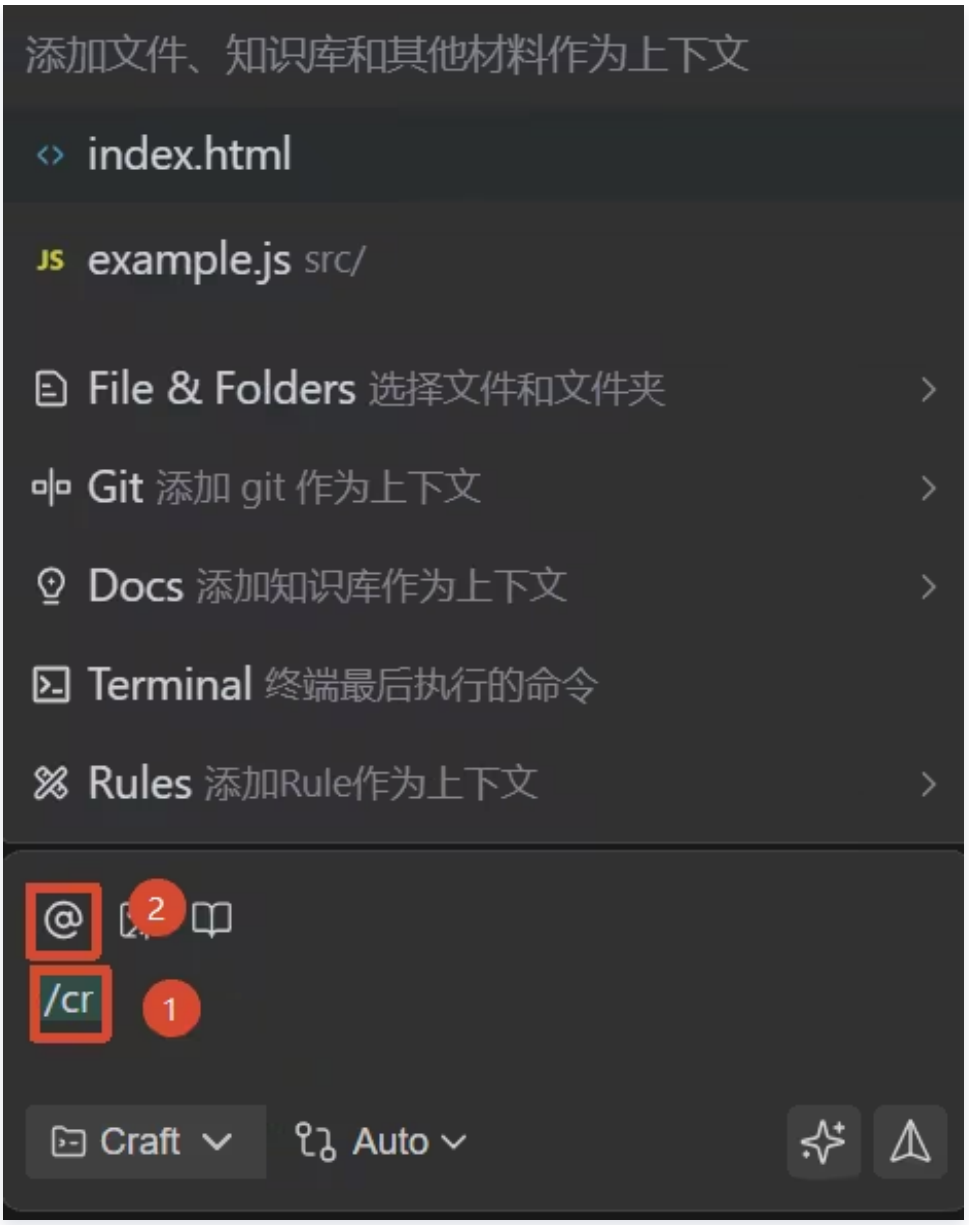
- **对于方法级评审，可以直接单击方法上方的代码评审来触发评审功能。**



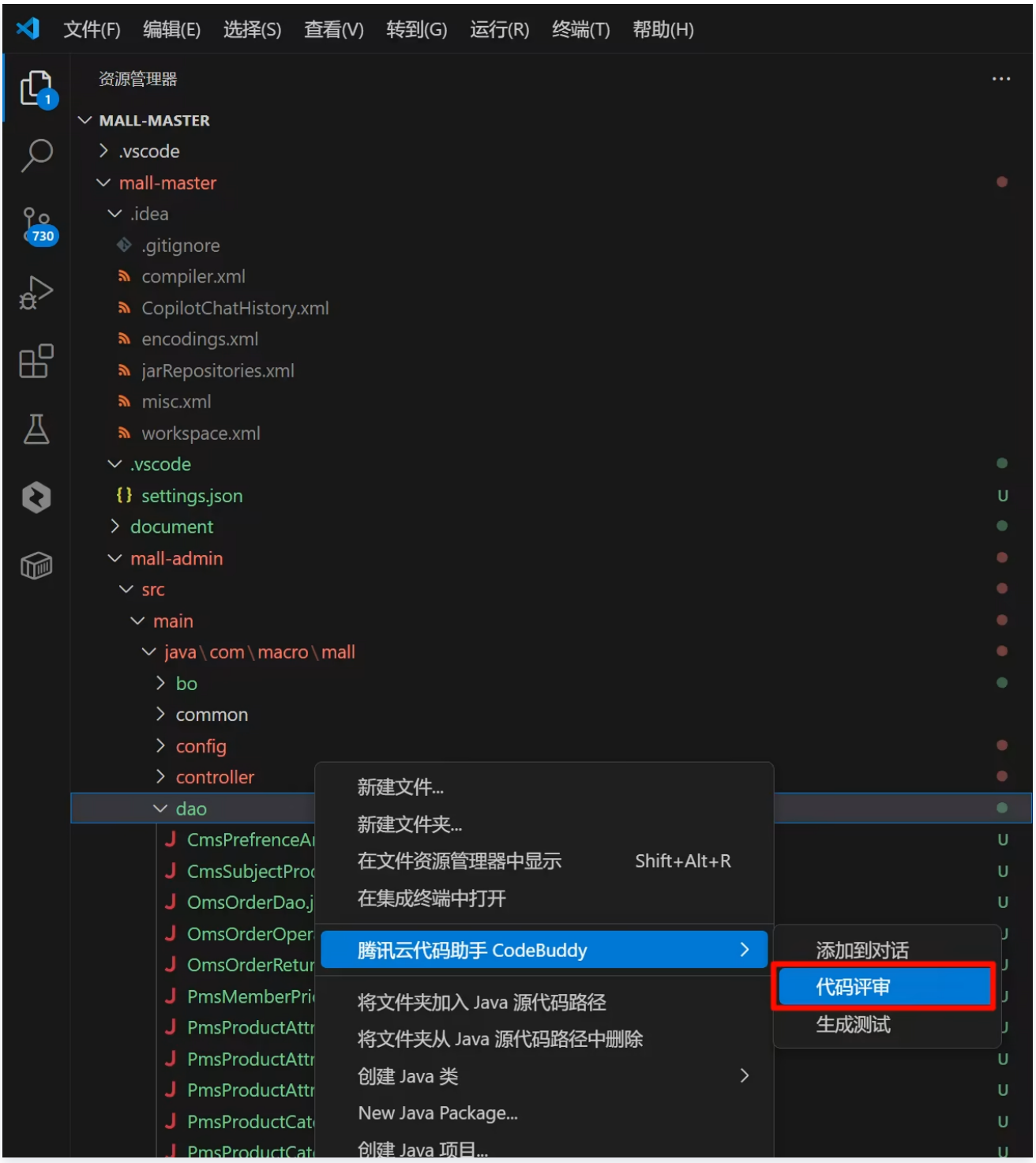
评审文件或文件夹

代码助手支持将文件或文件夹作为评审范围，支持在功能面板的对话输入框或文件树中进行评审触发。

- 在对话框中输入 `/cr`，然后点击 @ 引用需要评审的文件或文件夹。



- 在文件树中，鼠标单击文件或文件夹单击右键选择**腾讯云代码助手 CodeBuddy > 代码评审**。



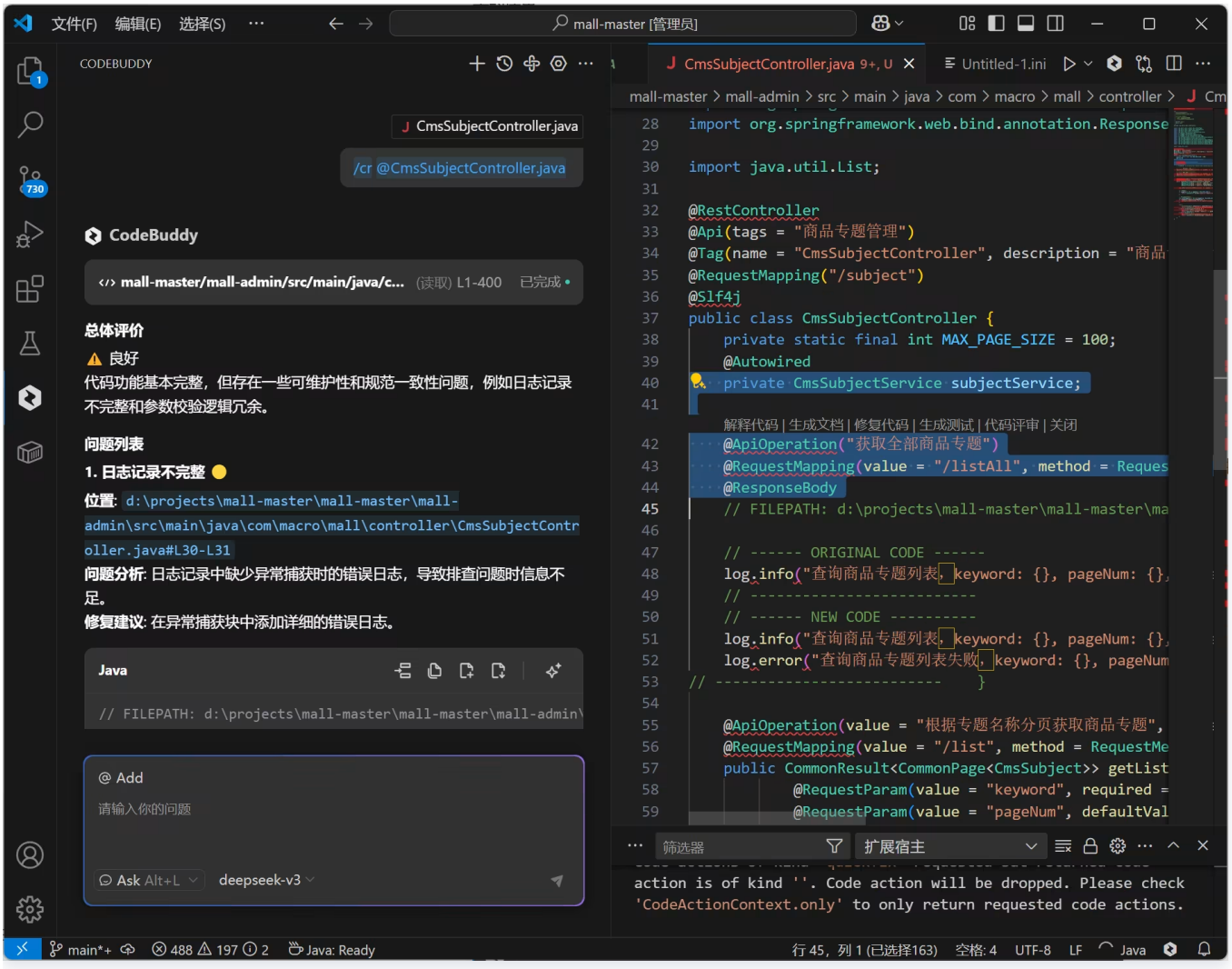
评审执行

您可以根据自己的需求，选择评审范围。例如：选择评审文件或文件夹进行评审时，通过 `/cr` 指令选择评审文件或文件夹后回车，将直接触发对所选文件或文件夹的评审。



查看评审结果

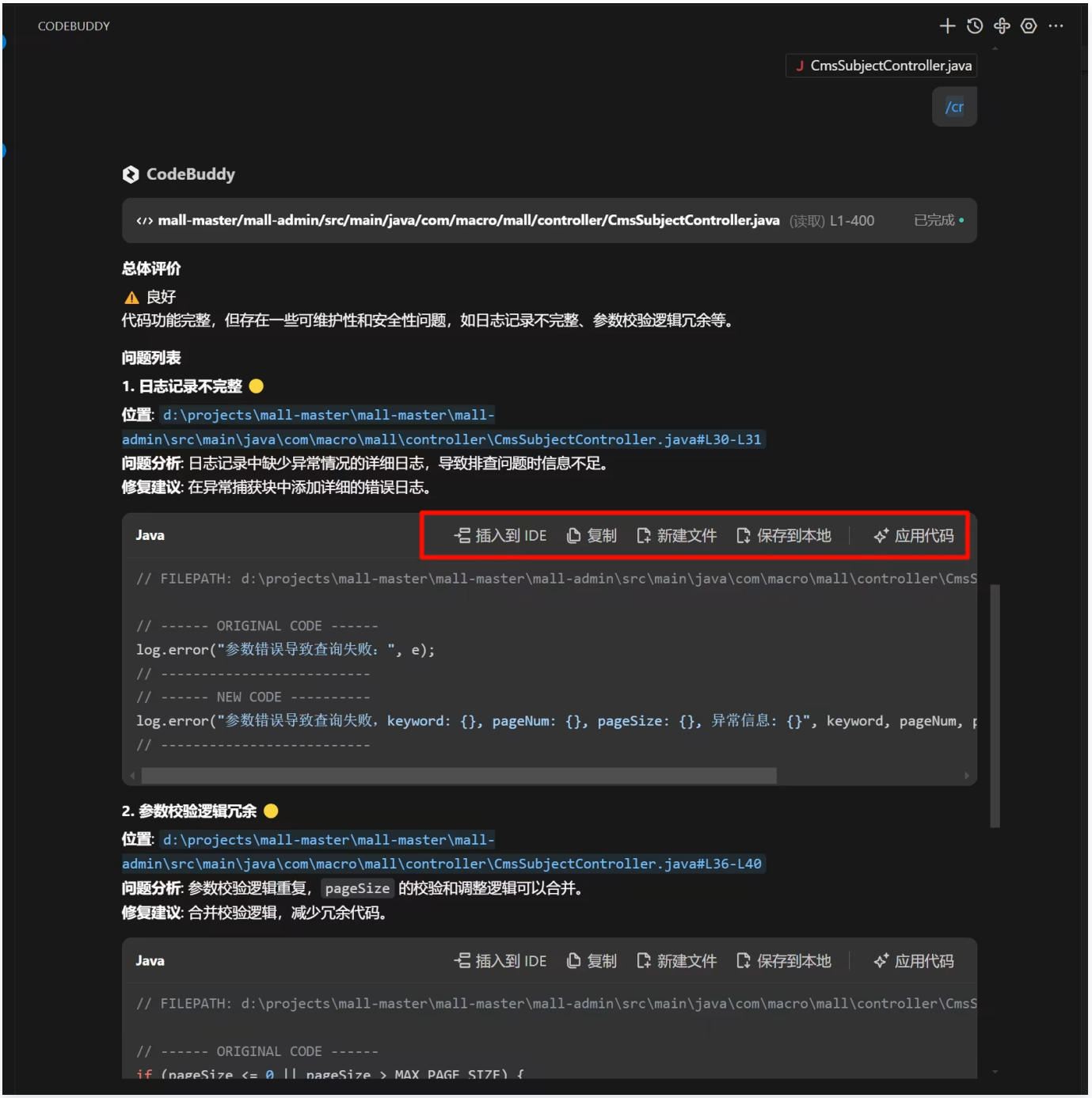
代码评审结束后，在对话面板内将直接返回评审结果，评审结果中将包含总体评价和问题列表，在问题列表中支持显示并单击蓝色位置区域跳转至问题所在文件代码片段进行查看。



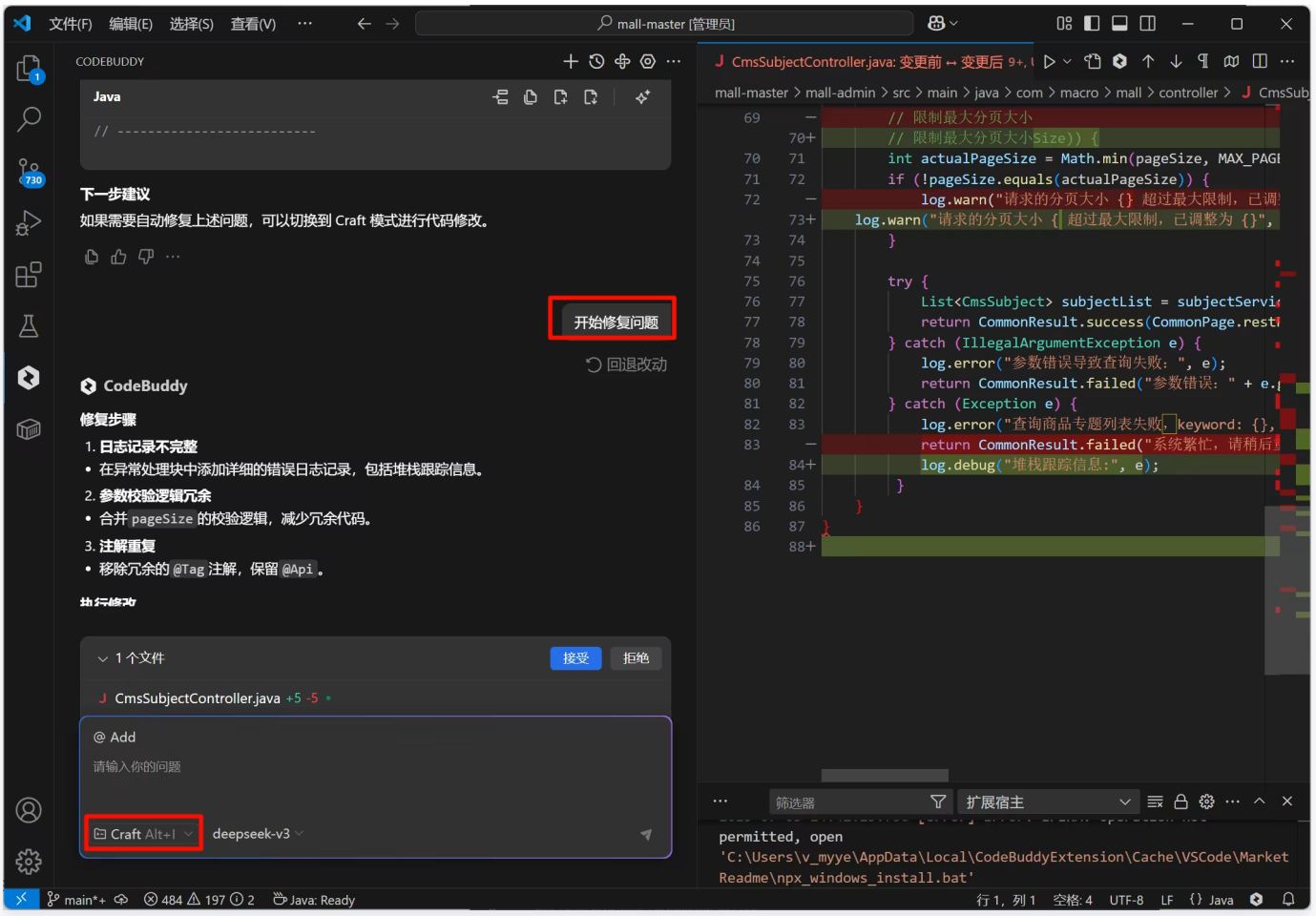
代码修复

评审结束后，您可以针对代码助手给出的评审意见进行修复。

- 单个问题修复 。您可以在问题列表中针对单个问题进行修复，例如选择**应用代码**时，将自动切换至 Craft 模式自动修复问题。

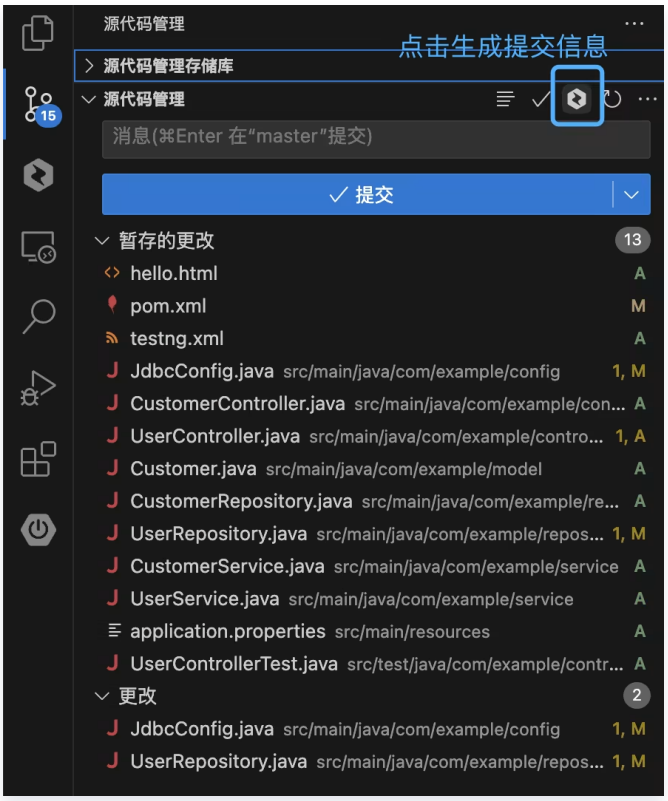


- 多个问题修复。您可以基于问题列表中的所有问题一次性进行修复。手动切换至 Craft 模式（可以单击对话输入框左下方模式切换按钮进行切换，也可以通过快捷键 **Alt + I** 进行切换），然后输入**开始修复问题**即可，Craft 将自动帮您将问题列表中的问题进行修复。在修复的过程中，将以 Diff 的形式进行展示，最后修复完成后，检查如果没问题选择**接受**即可。



AI 提交信息生成

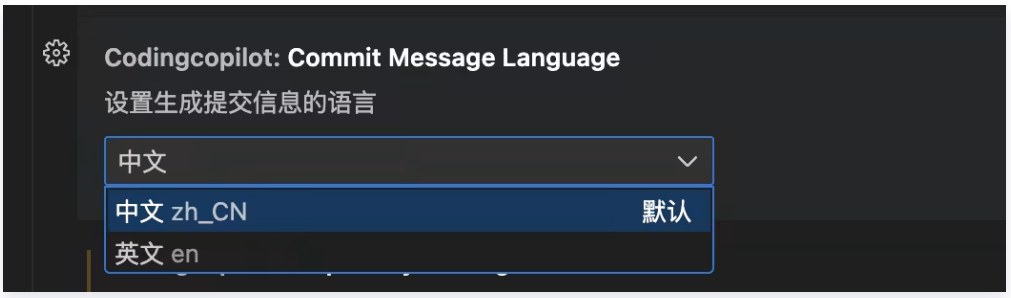
在本地 IDE 中提交代码变更时，支持一键唤起 AI，自动基于提交变更生成代码提交信息。该信息默认遵守 Conventional 提交信息规范，支持设置生成的语言、风格。借助 AI 快速生成规范化的提交信息，帮助开发者提升效率。



设置风格、语言

设置 AI 提交信息语言

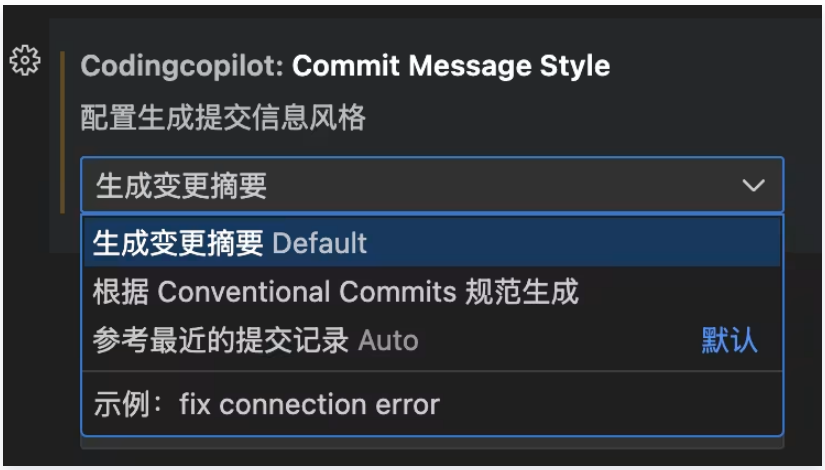
进入到插件设置里，在 **Commit Message Language** 中，配置 AI 提交信息语言，目前支持中文（默认）、英文。



设置 AI 提交信息风格

进入到插件设置里，在 **Commit Message Style** 中，选择配置 AI 提交信息生成的风格，如下图。

- 风格1：生成变更摘要（默认）。
- 风格2：根据 Conventional Commits 规范生成。
- 风格3：参考最近的提交记录。


Codingcopilot: Commit Message Style
配置生成提交信息风格
生成变更摘要
生成变更摘要 Default
根据 Conventional Commits 规范生成
参考最近的提交记录 Auto
默认
示例：fix connection error

代码解释

最近更新时间：2026-07-20 19:49:31

代码解释依托大模型，支持对文件、目录、函数、代码片段生成通俗易懂的结构化解读，适用于接手新项目、阅读无注释遗留代码、梳理复杂算法 / 业务逻辑等场景。支持三大环境：VSCode、微信开发者工具 IDE 插件、CodeBuddy IDE。

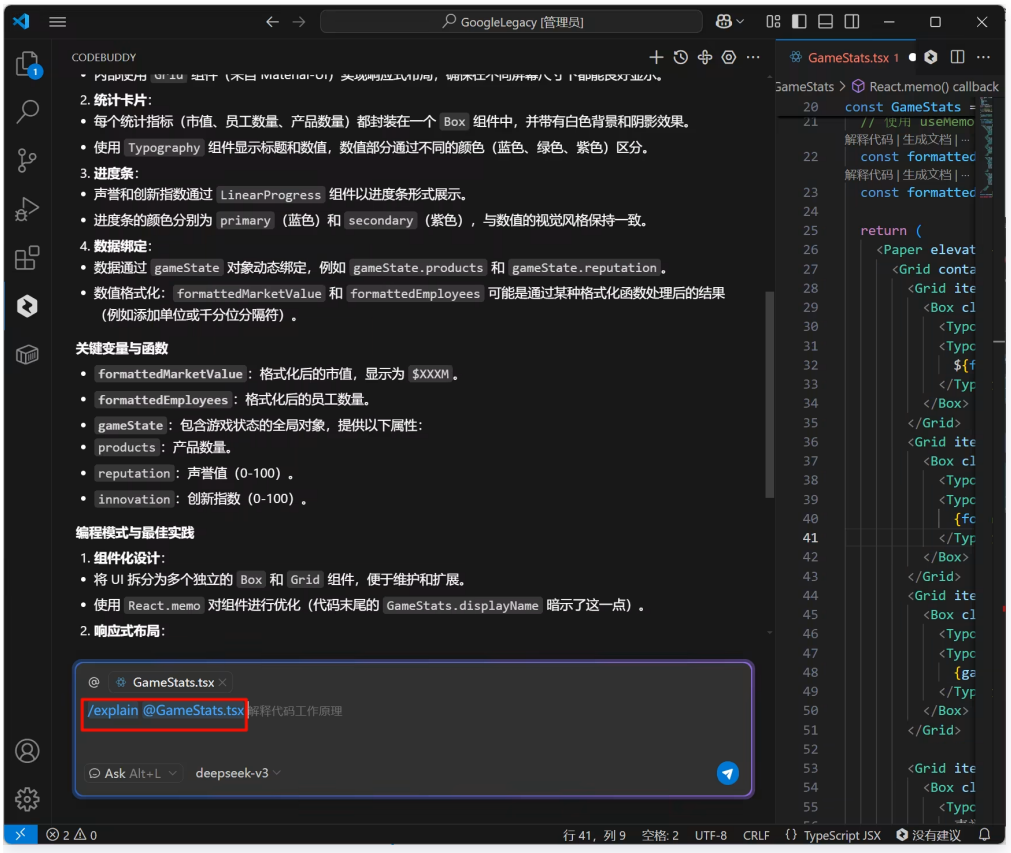
CodeBuddy 插件

VSCode

代码助手提供多种方式触发代码解释功能，您既可以在代码编辑区域中触发代码解释，也可以在代码助手对话输入框中使用 `/explain` 指令进行触发，方式如下：

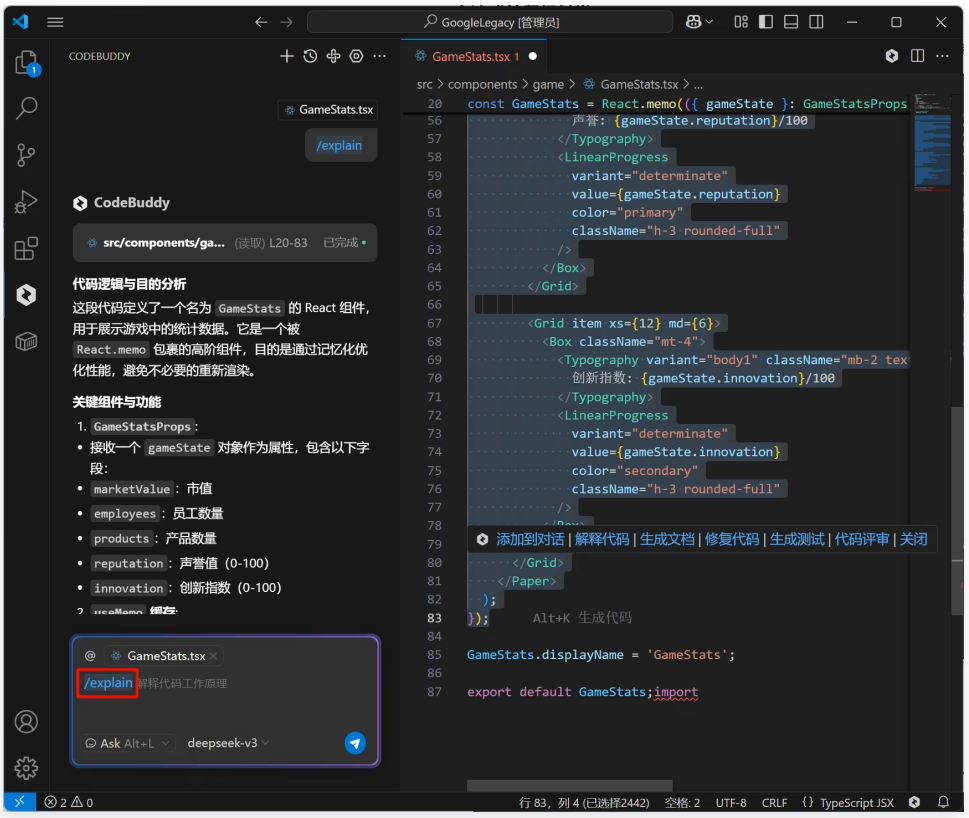
文件或目录级触发

在代码助手对话输入框中，输入 `/explain` ，然后选择**文件**或**目录**进行触发。触发后，代码助手会根据所选内容生成详细的解释，例如代码逻辑和目的、关键组件与设计、关键变量与函数、编程模式与最佳实践等等，帮助您快速理解。



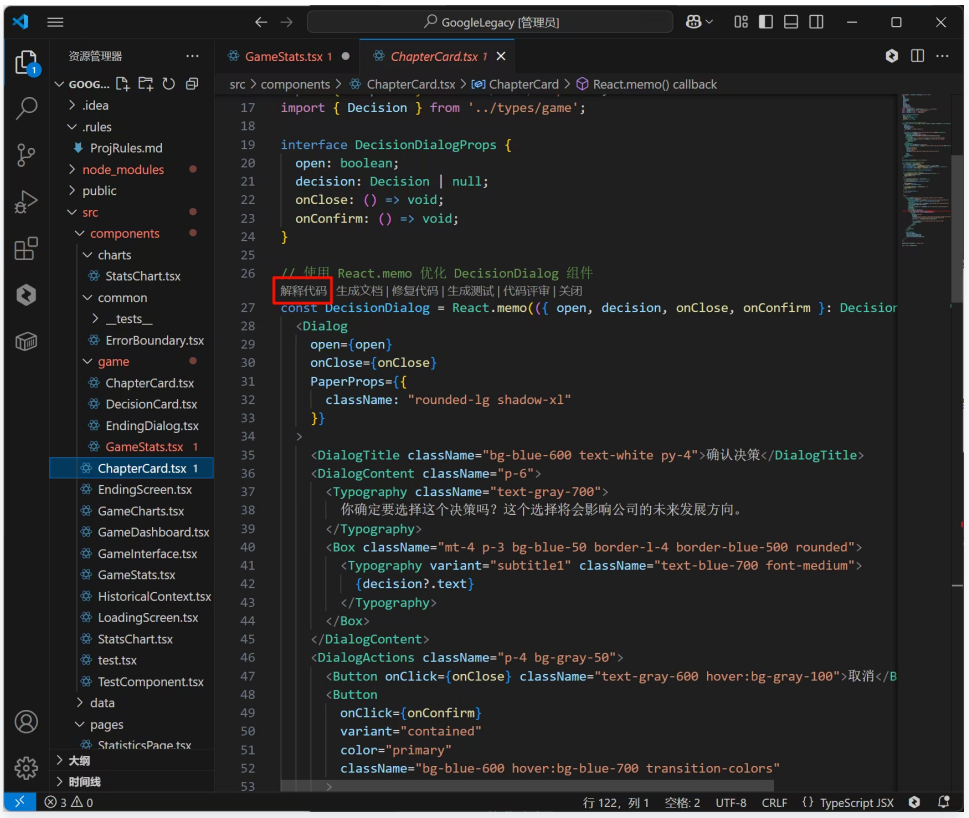
方法或片段级触发

- **方式一：**圈选代码片段后，在左侧对话输入框输入 `/explain` 指令触发代码解释。但这里有个点需要注意，输入 `/explain` 指令后会自动调用 @ 来添加上下文，这里已经选择了代码片段，不用添加其它上下文，可以直接按退格键（Backspace）删除 @ 即可。

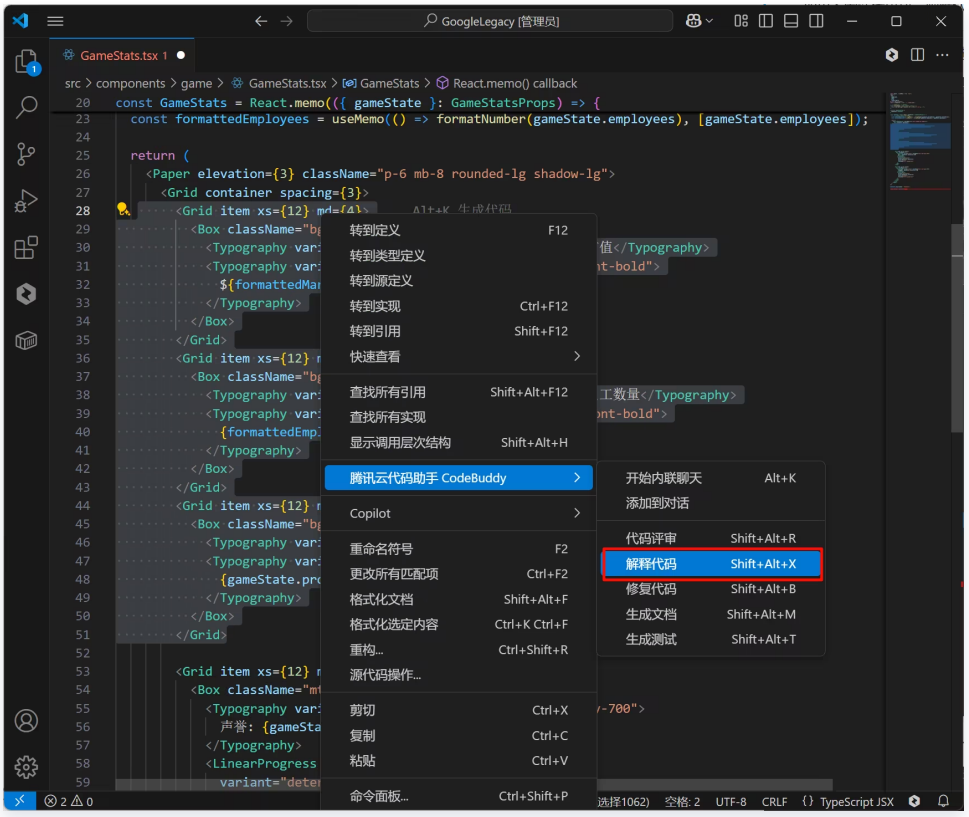


> **说明：**
> 如果不圈选代码块，则默认对整个代码编辑区中的代码进行解释。

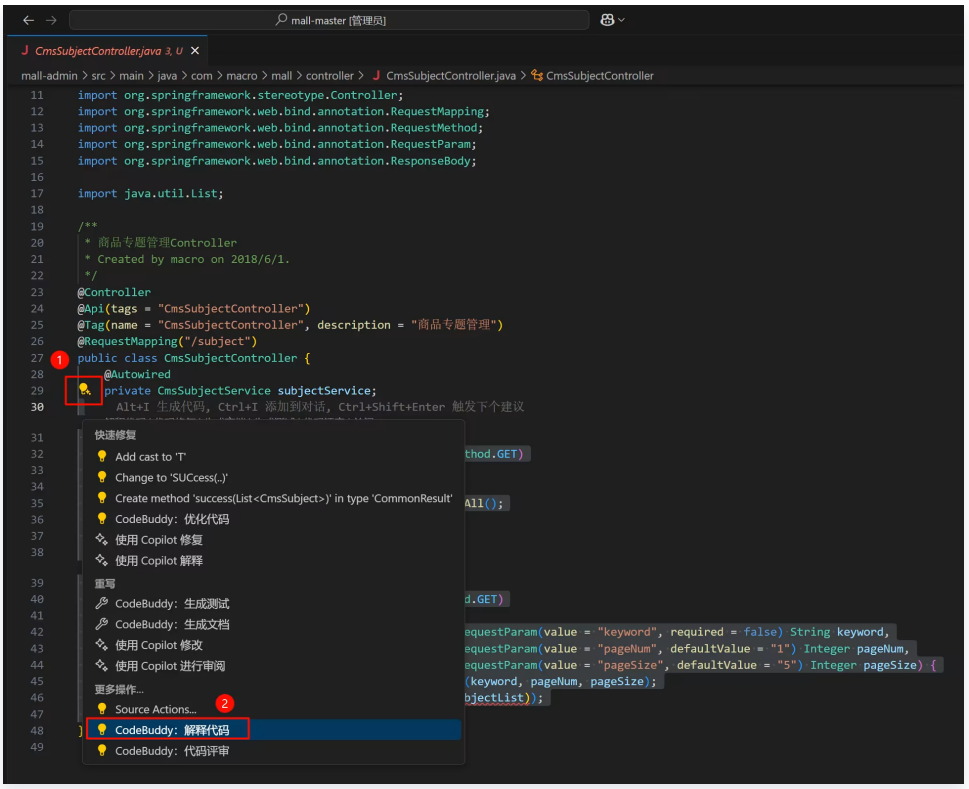
- **方式二：**在方法或函数上方的操作条中，选择**解释代码**即可。



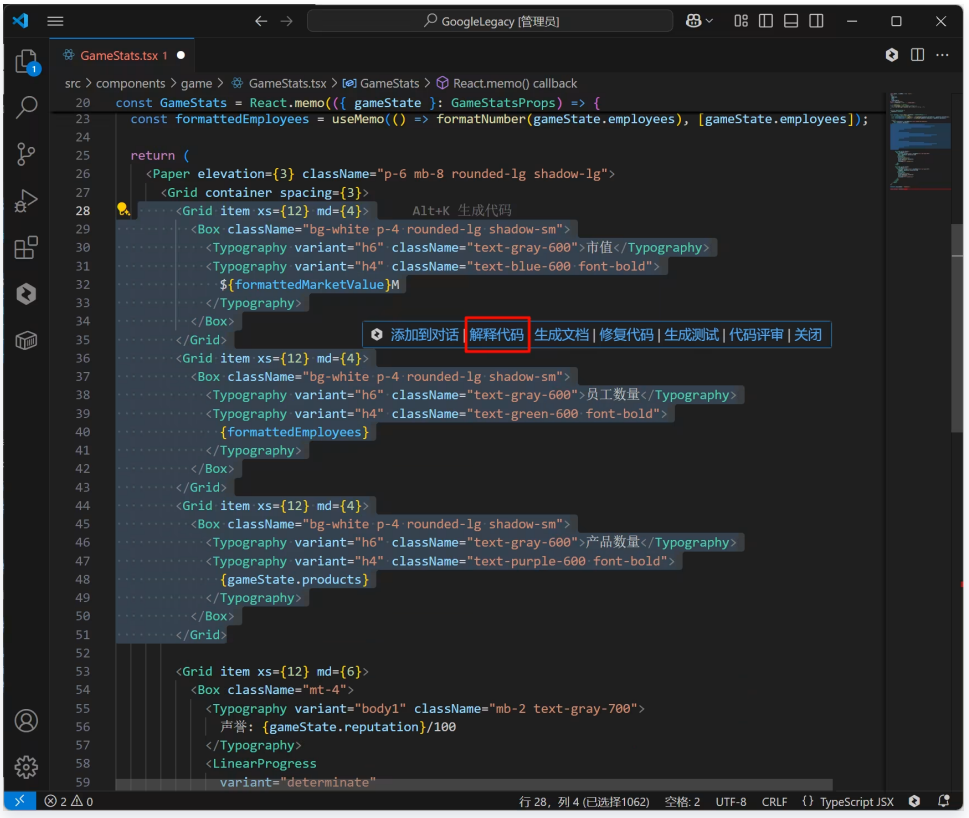
- **方式三：**圈选代码后，在编辑区域中直接右键选择 **腾讯云代码助手 CodeBuddy > 解释代码**，即可获得代码解释。或者根据右侧提示的快捷键进行快速触发。



- **方式四：**在圈选代码块后会出现灯泡图标，单击后出现一个功能对话框，选择 **CodeBuddy：解释代码**。



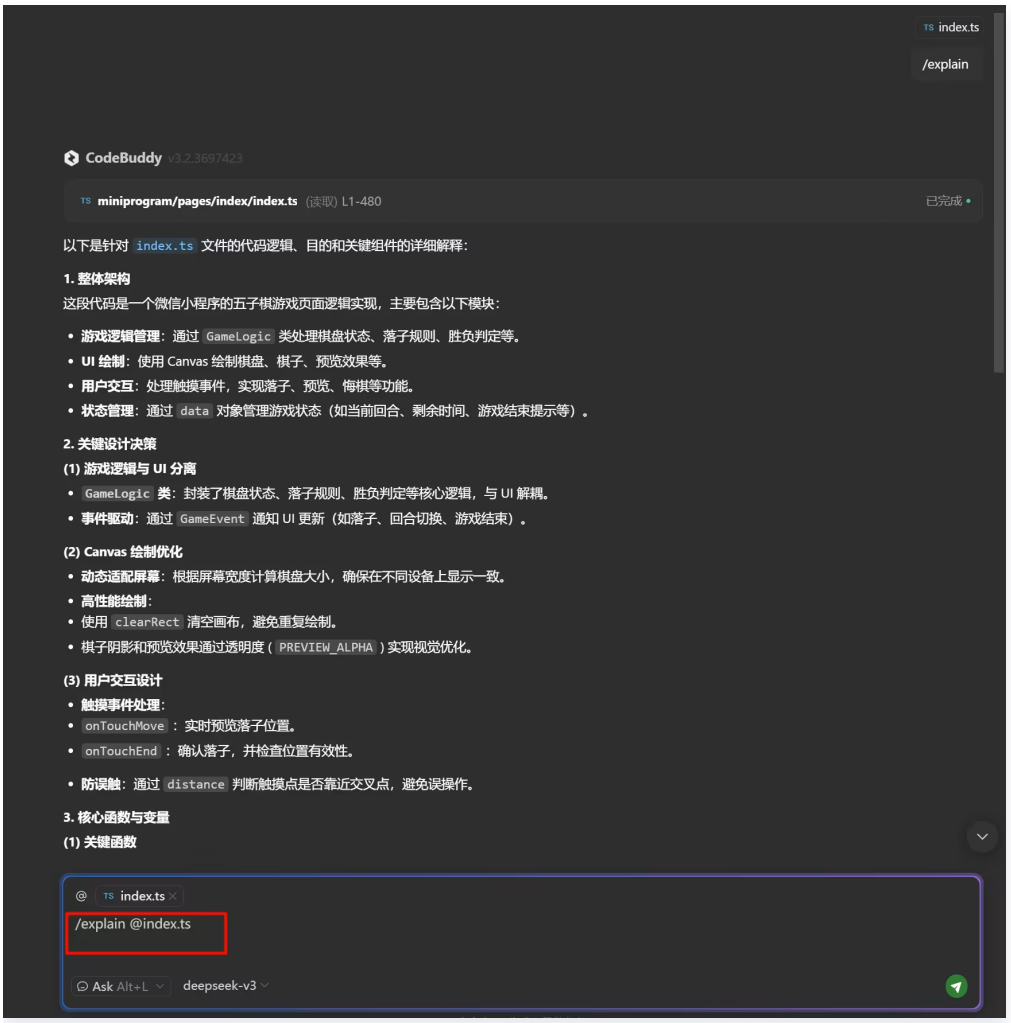
- **方式五：**在圈选代码块后，将鼠标光标悬浮在所选代码，会弹出一个工具条，选择**解释代码**即可。



微信开发者工具 IDE

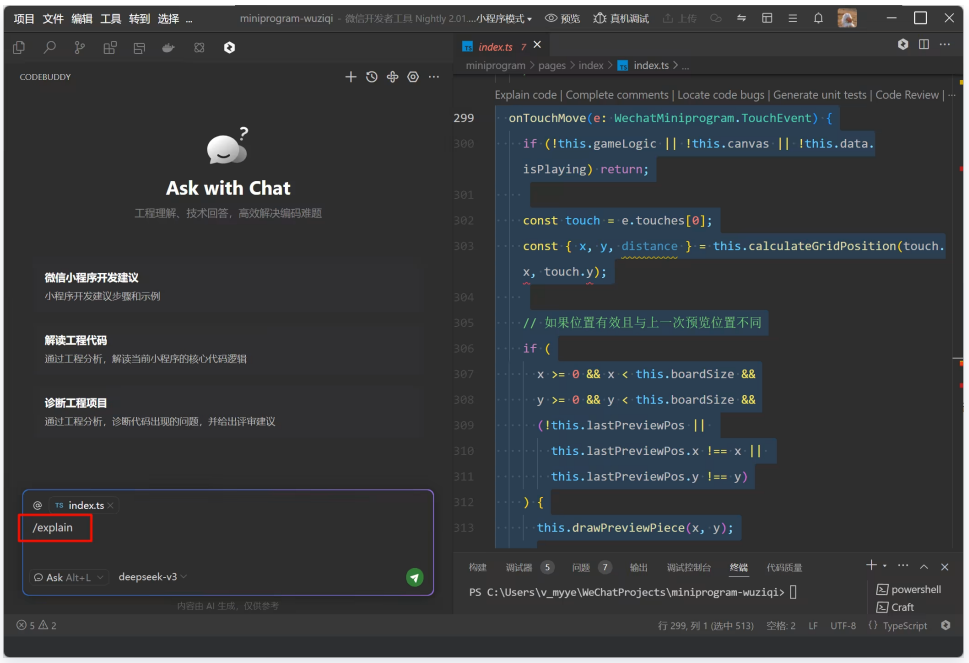
文件或目录级触发

在代码助手对话输入框中，输入 `/explain` ，然后选择**文件**或**目录**进行触发。触发后，代码助手会根据所选内容生成详细的解释，例如整体架构、关键设计、关键变量与函数、编程模式与最佳实践、优化建议等等，帮助您快速理解。



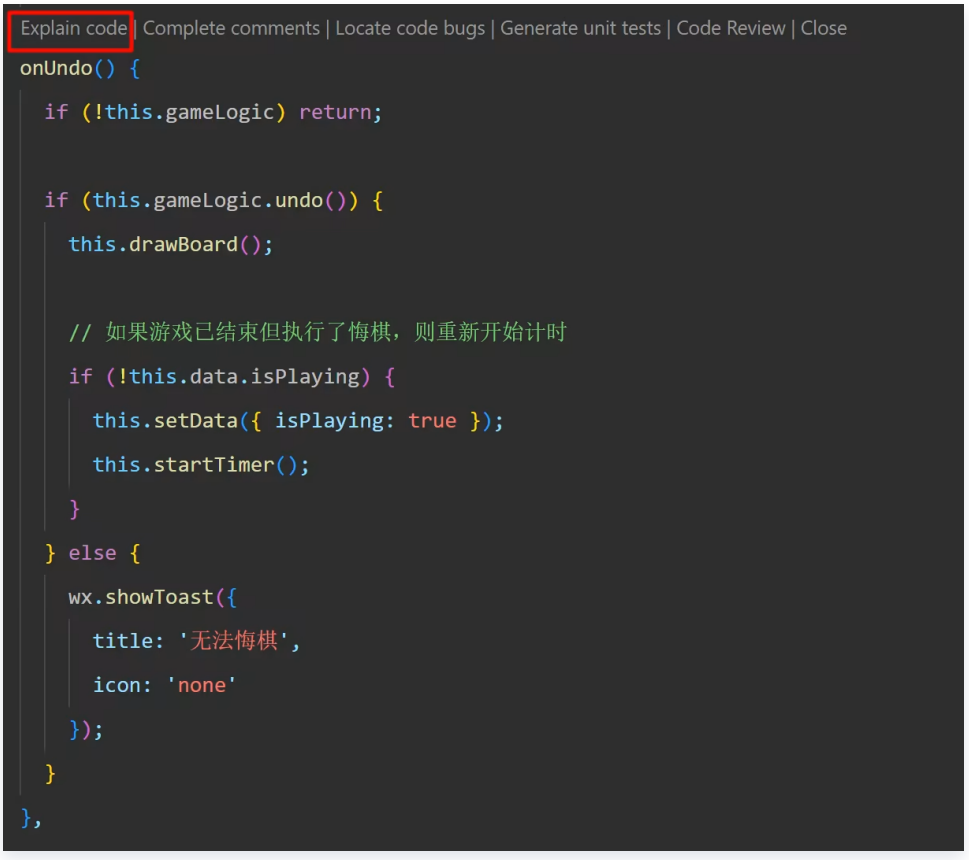
方法或片段级触发

- **方式一：**圈选代码片段后，在左侧对话输入框输入 `/explain` 指令触发代码解释。但这里有个点需要注意，输入 `/explain` 指令后会自动调用 @ 来添加上下文，这里已经选择了代码片段，不用添加其它上下文，可以直接按退格键（Backspace）删除 @ 即可。

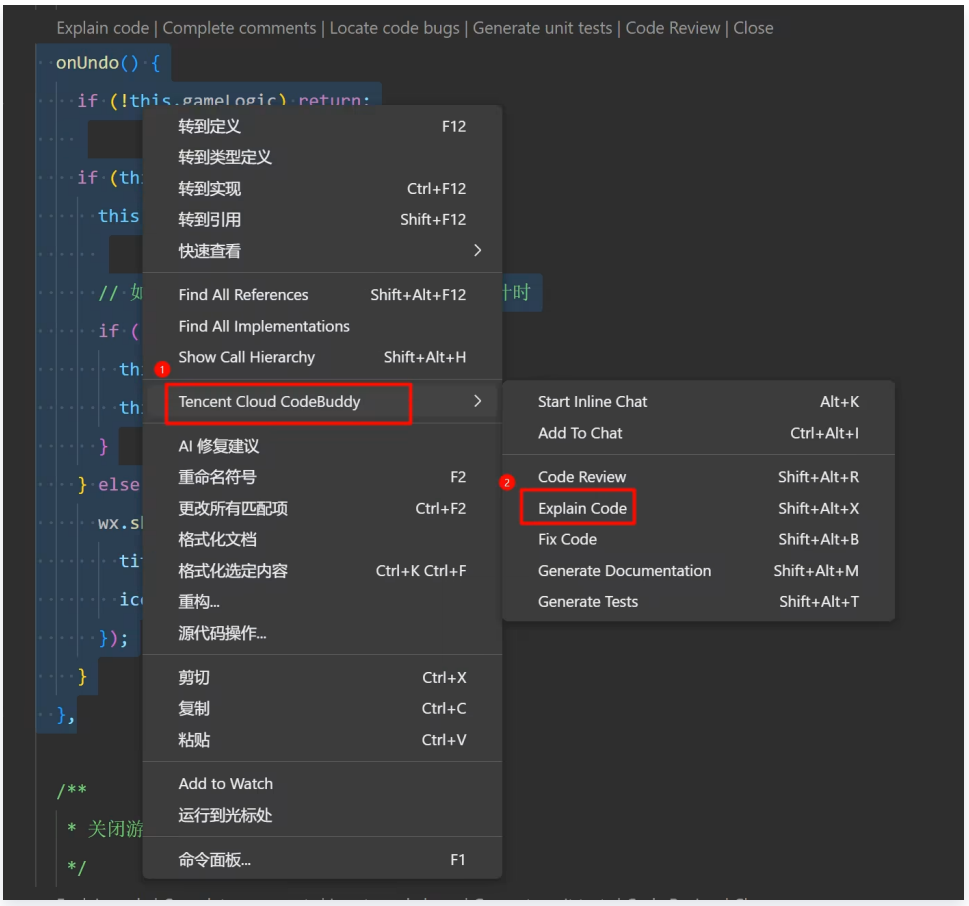


> **说明：**
> 如果不圈选代码块，则默认对整个代码编辑区中的代码进行解释。

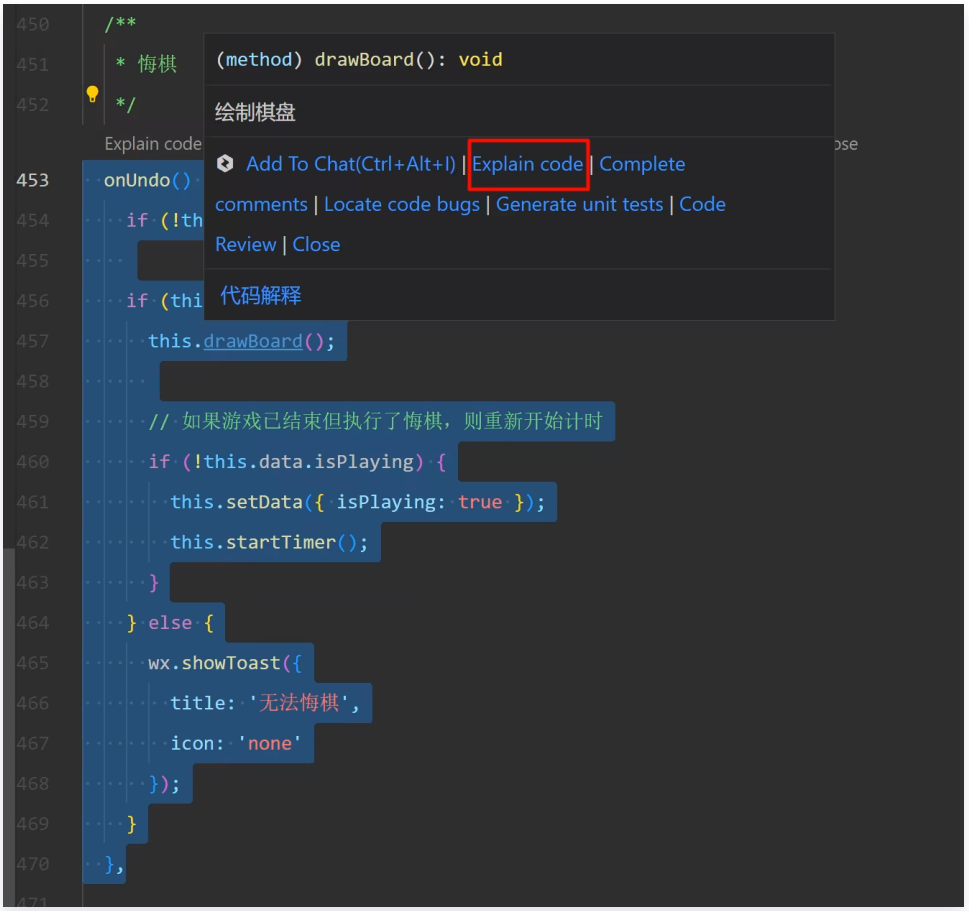
- **方式二：在方法或函数前的快捷方式中，选择 Explain code（解释代码）即可。**



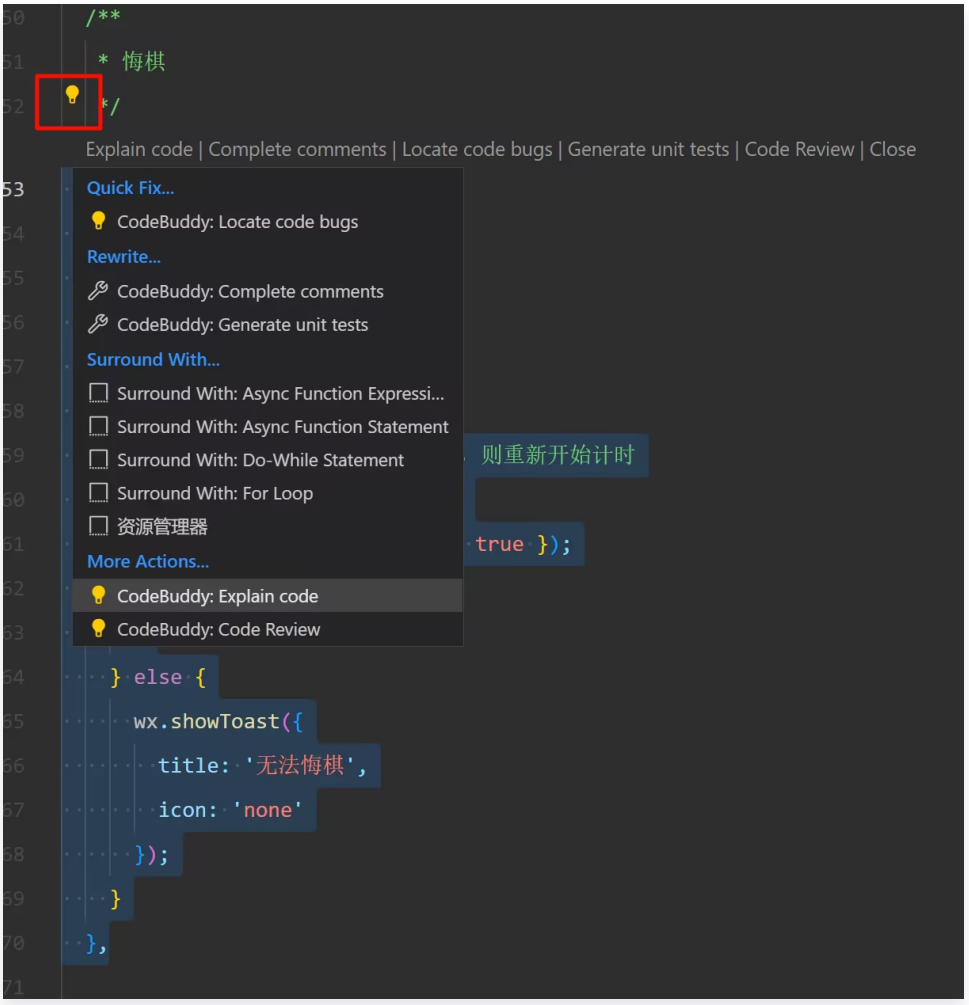
- **方式三：圈选代码后，在编辑区域中直接右键选择 Tencent Cloud CodeBuddy > Explain Code，即可获得代码解释。或者根据右侧提示的快捷键进行快速触发。**



- **方式四：在圈选代码块后，将鼠标悬浮在所选代码，在弹出菜单中选择 Explain code 即可。**



- **方式五：在圈选代码块后会出现一个类似小灯泡的小图标，单击后在弹出菜单中选择 CodeBuddy：Explain code 进行触发。**



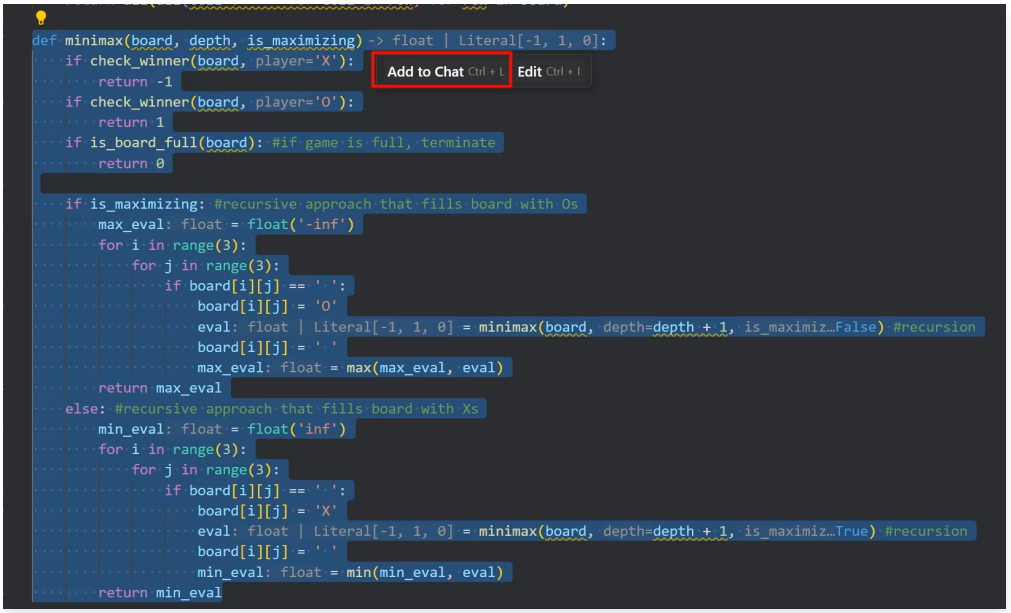
CodeBuddy IDE

在 CodeBuddy IDE 中，你可以触发方法级的代码解释，也可以触发文件/目录级的代码解释。触发后，选择 Chat 模式生成代码解释。

触发代码解释

方法级触发

圈选函数级代码块后，在弹出菜单中选择 **Add to Chat** 将代码添加到对话中，也可以使用右侧提示的快捷键快速将代码添加到对话中，或者右键在菜单选项中选择 **Add to Chat** ，触发代码解释。

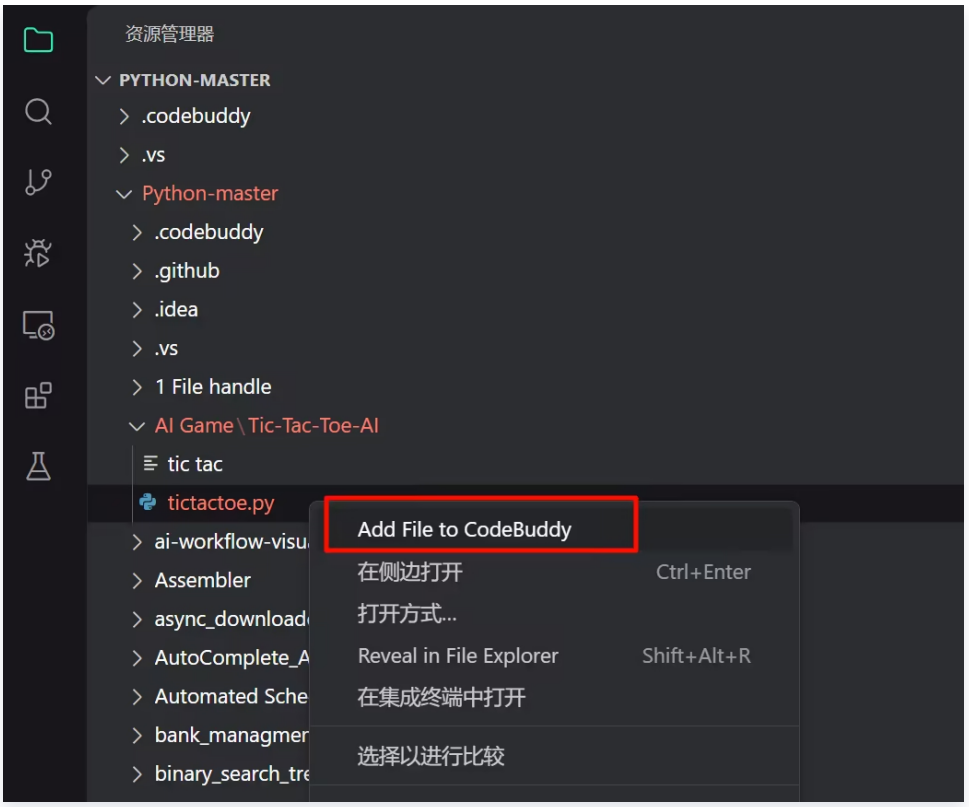


文件/目录级触发

- 在对话框中，可以通过 @ 符号将文件/目录添加到对话上下文中触发代码解释。



- 在文件树中，右键单击文件/目录，在菜单选项中选择 **Add File to CodeBuddy**，将文件/目录添加到对话中，触发代码解释。



代码解释生成（适用于无代码注释或接手新项目的场景）

触发代码解释后，Agent 将针对所选代码进行详细的代码解释，包括代码结构、使用的算法等等。



版权所有：腾讯云计算（北京）有限责任公司

内联对话

最近更新时间：2026-07-20 19:49:31

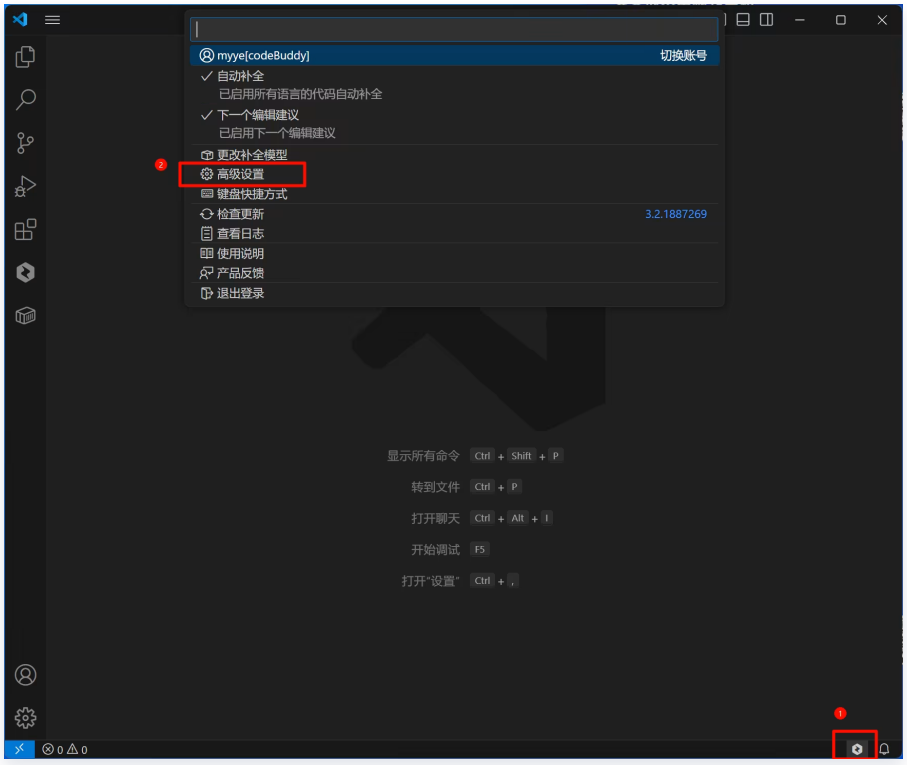
内联对话（Inline Chat）是在代码编辑区中嵌入输入框，并在输入框里直接输入需求或调用 / 指令后，直接生成代码并插入到当前光标位置。该效果极大地加强并拓展了编码过程中代码生成的能力。

CodeBuddy 插件

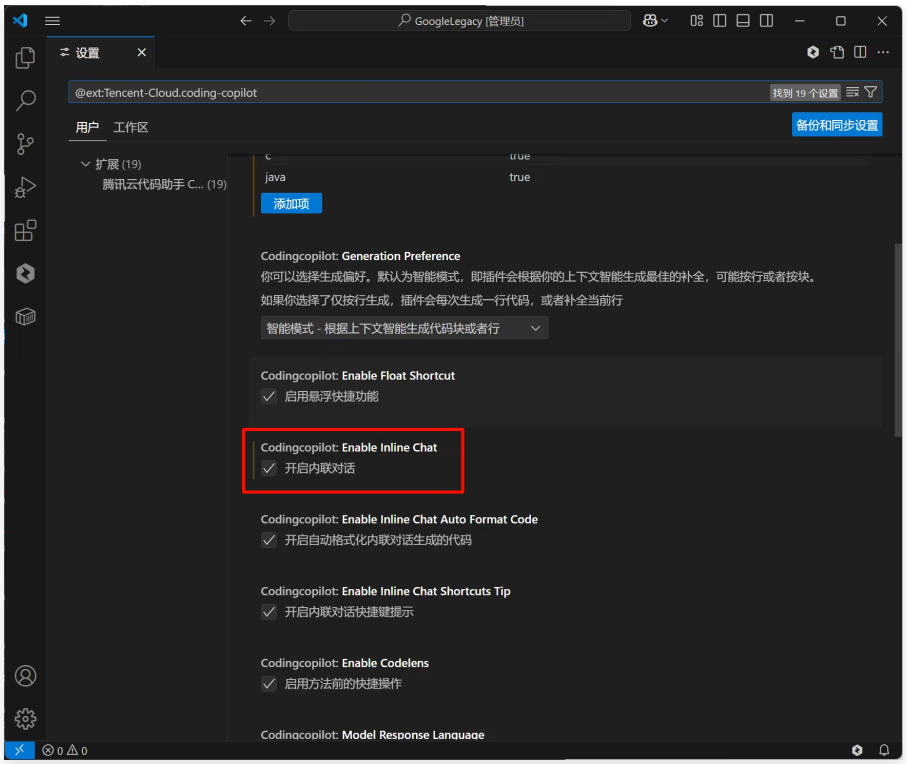
开启内联对话

VS Code

1. 单击底部腾讯云代码助手的 **icon 按钮**，弹出顶部菜单条，选中**高级设置**。

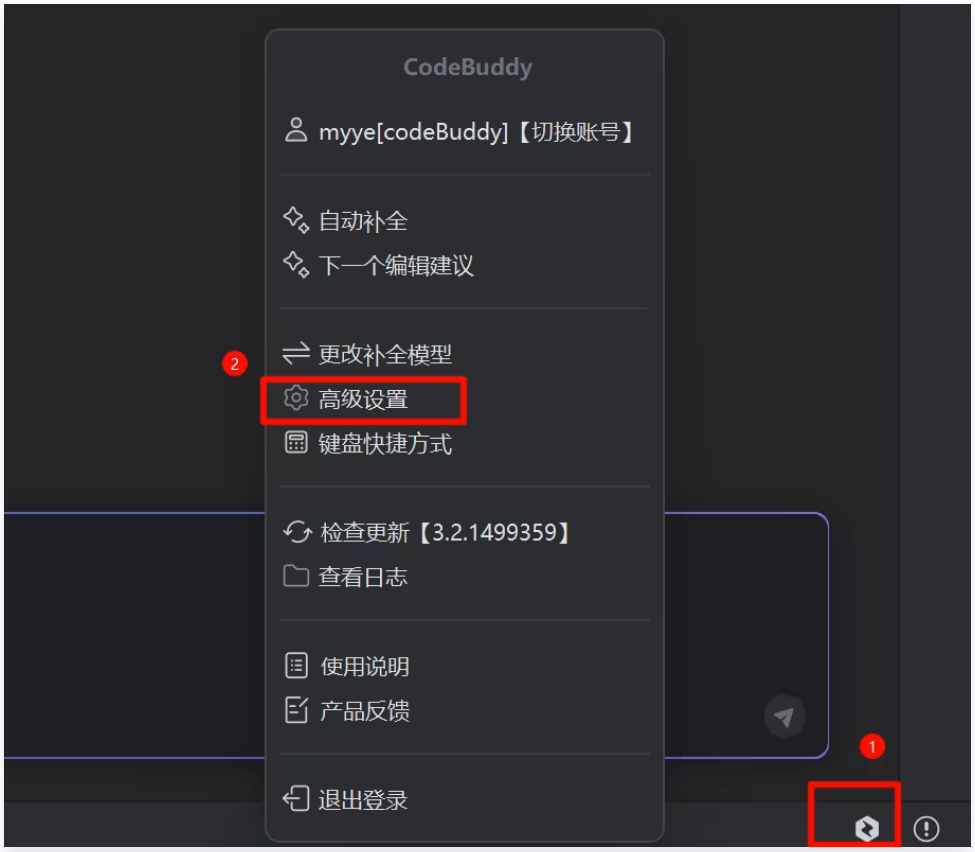


2. 确认 **Enable Inline Chat** 已经开启。

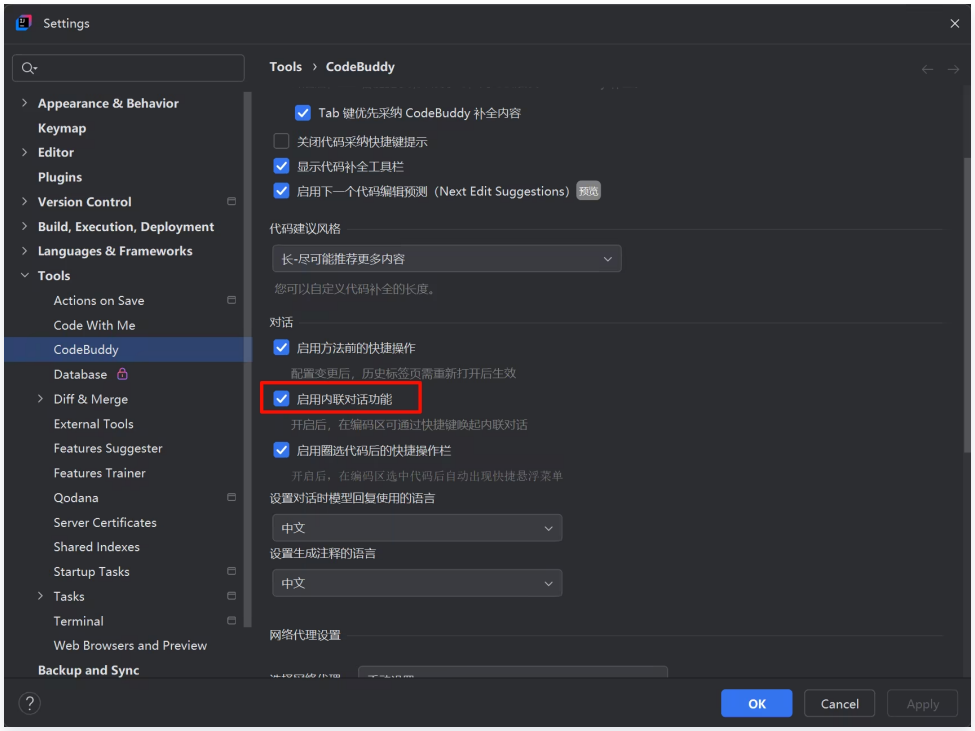


JetBrains IDE

1. **单击底部腾讯云代码助手的 icon 按钮，弹出菜单条，选择高级设置。**

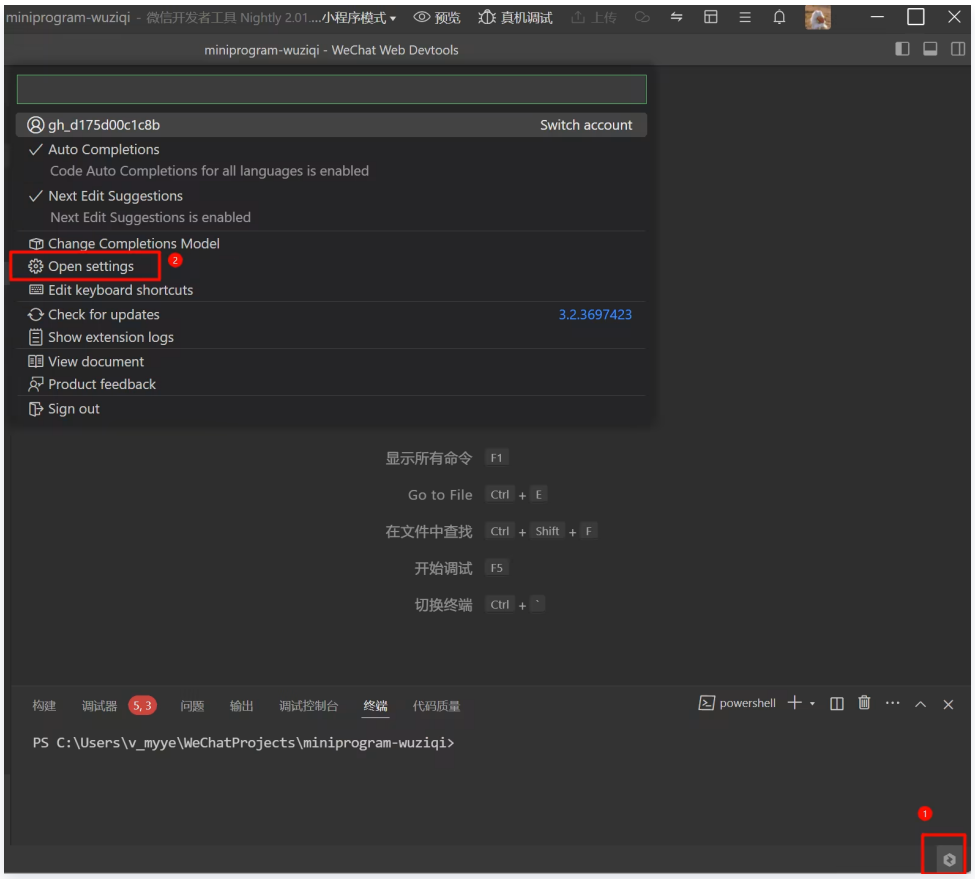


2. 确认**启用内联对话功能**已开启。

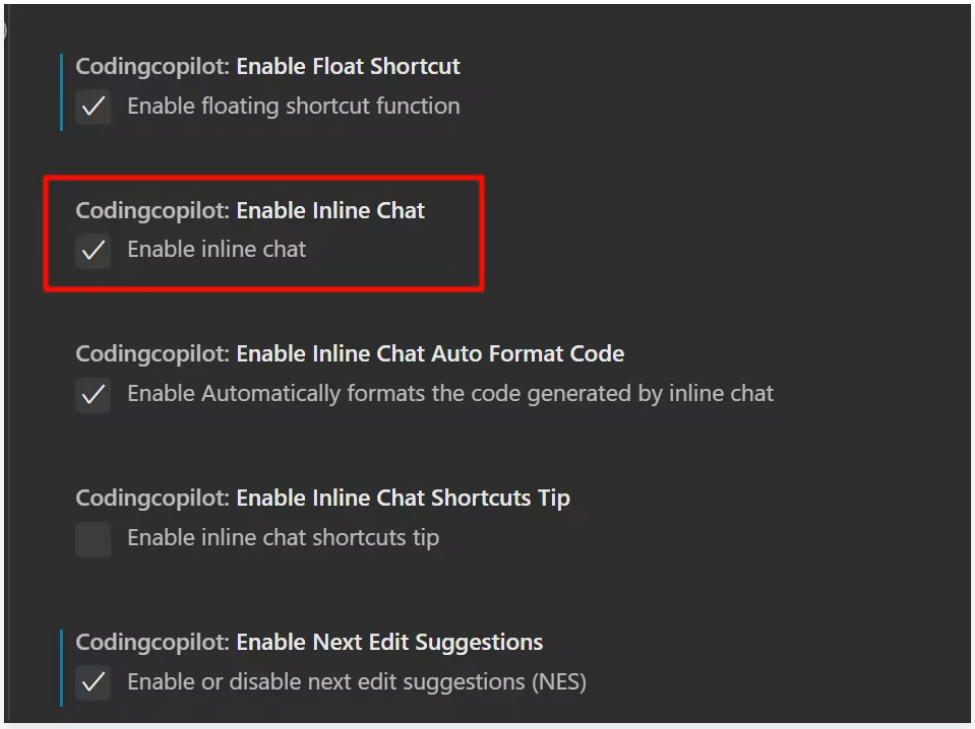


微信开发者工具 IDE

1. 单击底部腾讯云代码助手的 **icon 按钮**，在弹出菜单中选择 **Open settings**。



2. 确认**启用内联对话功能**已开启。

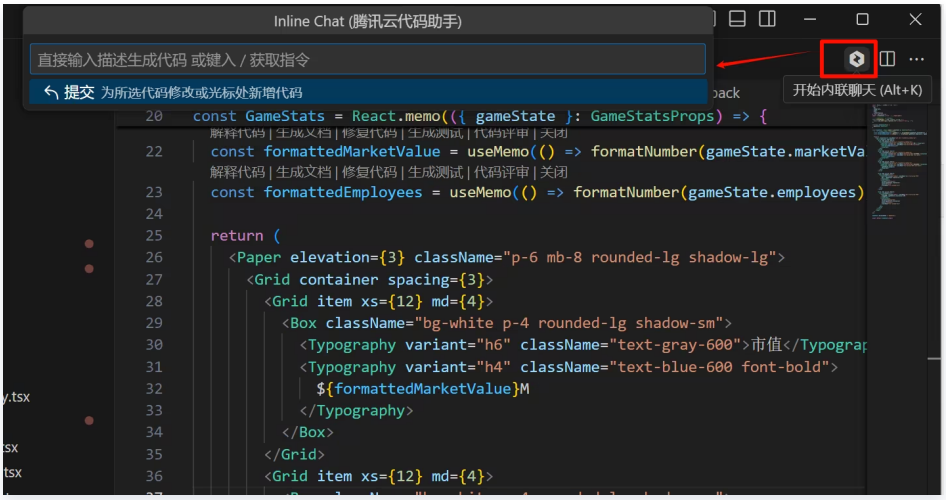


使用内联对话

可通过如下方式唤起内联对话功能。

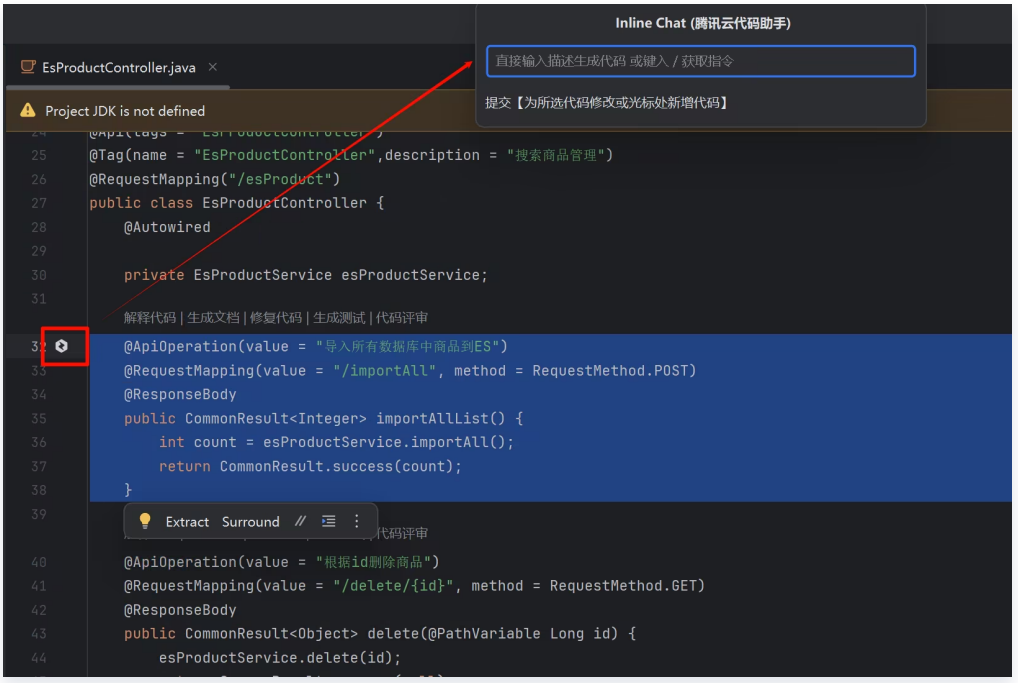
VS Code

在编辑区的右上方，单击腾讯云代码助手的 icon 图标，可唤起内联对话功能。



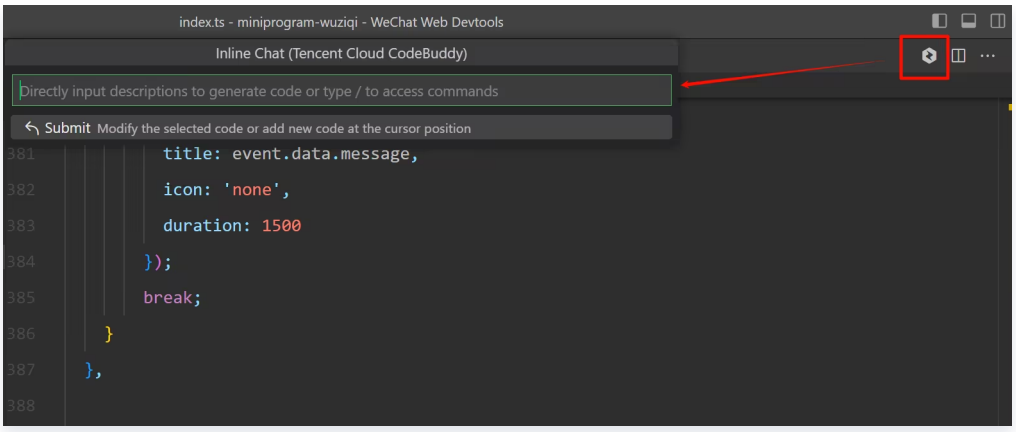
JetBrains IDEs

圈选代码区域后，在侧边弹出腾讯云代码助手的 icon 图标，单击可唤起内联对话功能。



微信开发者工具 IDE

在编辑区的右上方，单击腾讯云代码助手的 icon 图标，可唤起内联对话功能。



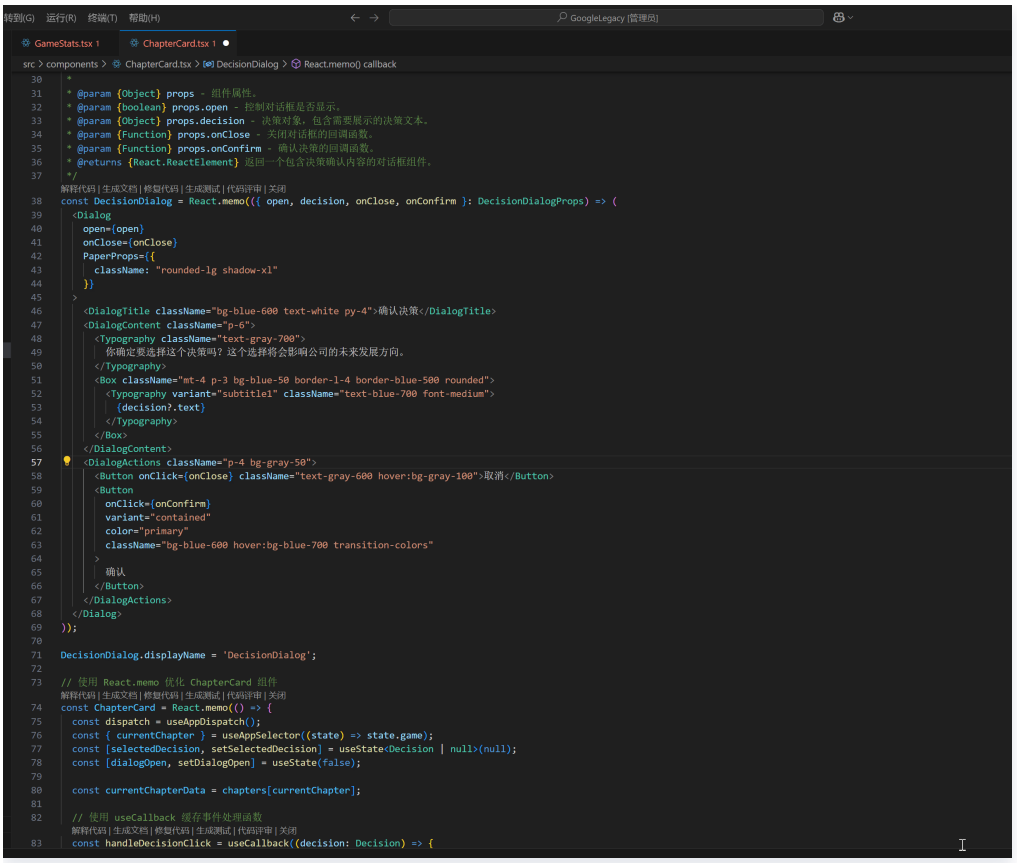
此外，也可以通过快捷键的方式快速唤起内联对话功能。首先确保插件快捷键没有冲突。我们唤起内联对话的快捷键如下：

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

唤起内联对话之后，您可以输入自然语言描述的需求，或者通过指令 `/` 做快捷场景。
典型的应用场景有：

- 快速生成或插入常用代码模板，例如函数、类、配置等。
- 对现有代码进行重构、重命名、逻辑优化。
- 实时代码诊断及修复提示，例如类型校验、安全检测等。
- 在开发新功能时，通过自然语言快速实现思路落地。

版权所有：腾讯云计算（北京）有限责任公司

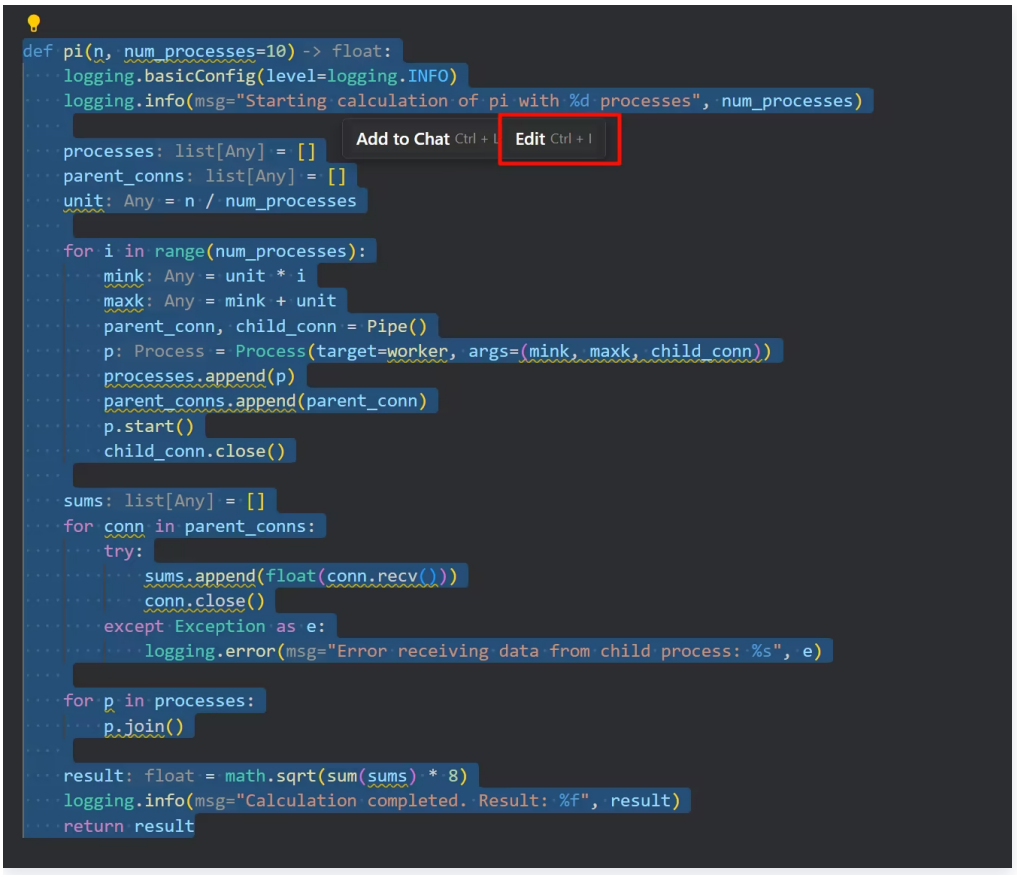


如上面视频所示，使用 `/doc` 生成注释后，开发者可以选择接受或者取消。具体的快捷键如下表所示：

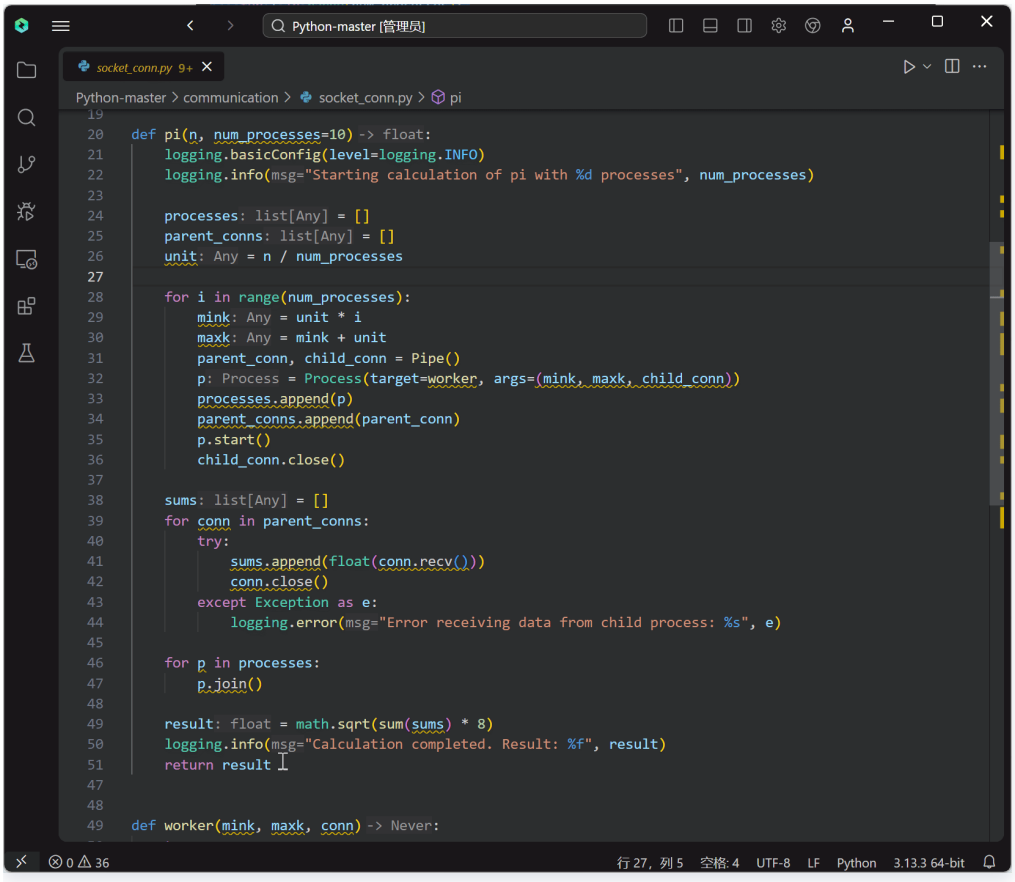
系统	操作	描述
Windows	Alt + A	接受（Accept）内联对话生成的代码
	Alt + Z	取消内联对话生成
	Alt + X	拒绝（Reject）内联对话生成的代码
Mac	Opt + A	接受（Accept）内联对话生成的代码
	Opt + Z	取消内联对话生成
	Opt + X	接受（Accept）内联对话生成的代码

CodeBuddy IDE

在 CodeBuddy IDE 中，内联对话功能已经默认开启，你可以圈选代码块后在弹出菜单中单击 **Edit** 按钮，也可以直接通过右侧提示的快捷键（Windows：Ctrl + I，Mac：⌘ Command + I）快速唤起内联对话输入框。唤起对话输入框后，可以通过快捷键 **Escape** 退出。



唤起输入框后，直接输入需求描述即可，AI 将根据您编辑所选代码或生成新代码。编辑/生成代码后，提供单个/批量的接受或拒绝处理。



版权所有：腾讯云计算（北京）有限责任公司

工程理解

最近更新时间：2026-07-20 19:49:31

代码库问答功能（@workspace，#Codebase）旨在协助开发者迅速找到与整个代码库相关的问题解答。它能够针对代码结构、函数功能、类之间的关系，以及复杂的代码逻辑和业务流程的疑问，提供精确并且与上下文相关的答案。

workspace

VS Code

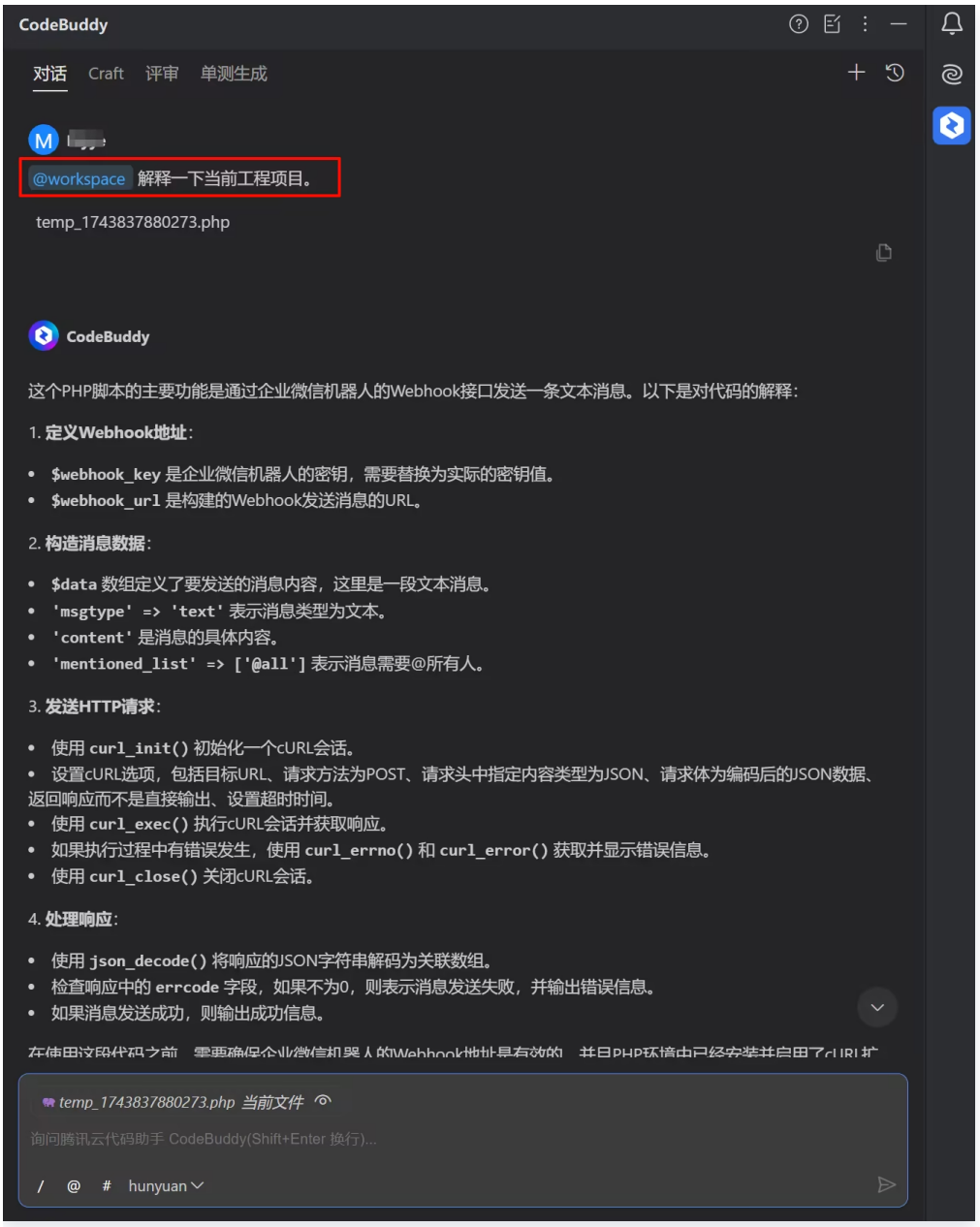
在 VS Code 中，单击侧栏对话，打开对话框，输入@workspace及问题描述后，按 Enter 建发送。
例如，在对话框中输入提示词“@workspace 解释一下当前工程项目。”后按 Enter 建发送，触发 @workspace 能力，生成效果如下图。您可以清楚地看到当前工程项目的主要组成部分以及各个组件等内容。

版权所有：腾讯云计算（北京）有限责任公司



JetBrains IDE

在 JetBrains IDE，例如 IDEA 中，打开侧栏对话，单击对话框， @workspace，输入问题描述，按 Enter 建发送。

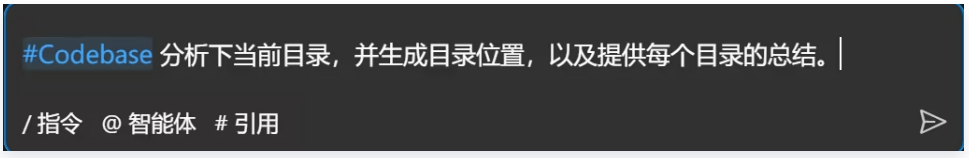


Codebase

#Codebase 作为腾讯云代码助手的一大功能，旨在提升代码查询和片段召回的精准性，帮助开发者高效编写、理解和优化代码，为开发者提供更智能的代码建议和解决方案。以下将为您介绍 #Codebase 功能。

- **代码助手能正确分析当前工程目录，协助您快速理解当前工程。**

  输入提示词：“#Codebase 分析下当前目录，并生成目录位置，以及提供每个目录的总结”，触发 #Codebase 能力。



  从下图的输出结果中，输出结果有召回内容，查询到的目录位置正确，没有出现不存在的目录，而且也生成了每个目录的总结描述，可以协助您快速理解当前工程。

VS Code

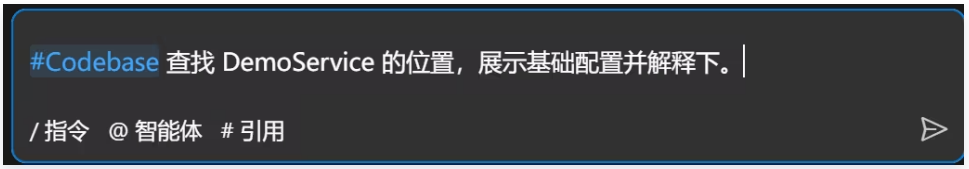


JetBrains IDE



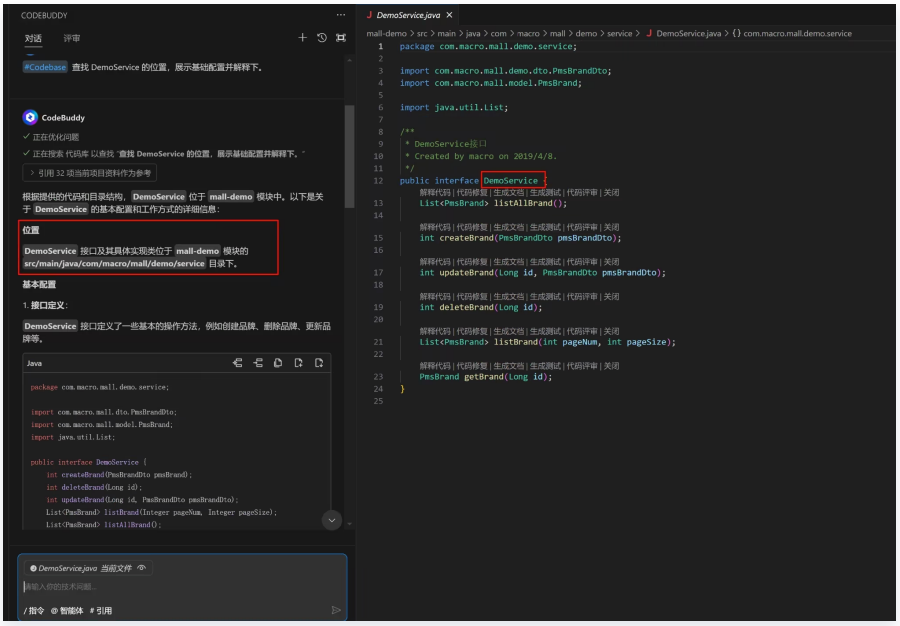
- **代码助手能正确找到某个方法/函数，可以协助您快速理解该方法/函数，以便进行快速开发。**

  输入提示词：“#Codebase 查找 DemoService 的位置，展示基础配置并解释下”，触发 #Codebase 能力。

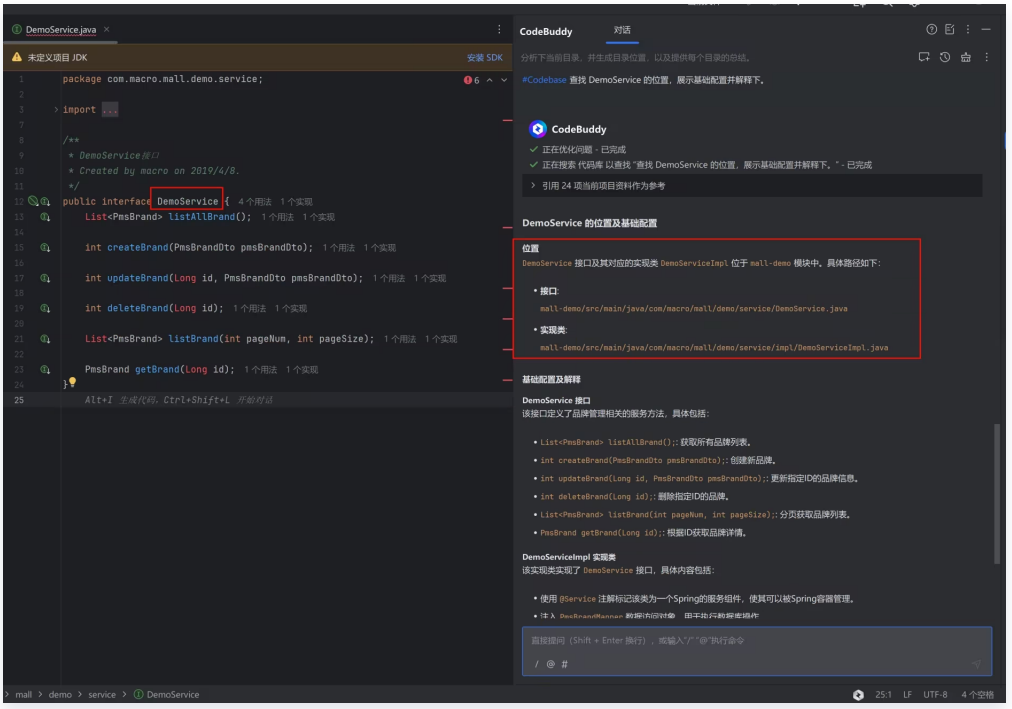


  从下图的输出结果中，您可以看到，输出结果有召回内容，可以查询到 DemoService 方法位于哪个目录文件下，而且在召回列表中单击文件可以实现跳转，同时生成解释说明。

VS Code

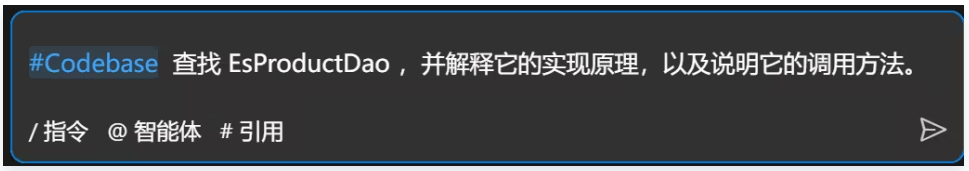


JetBrains IDE



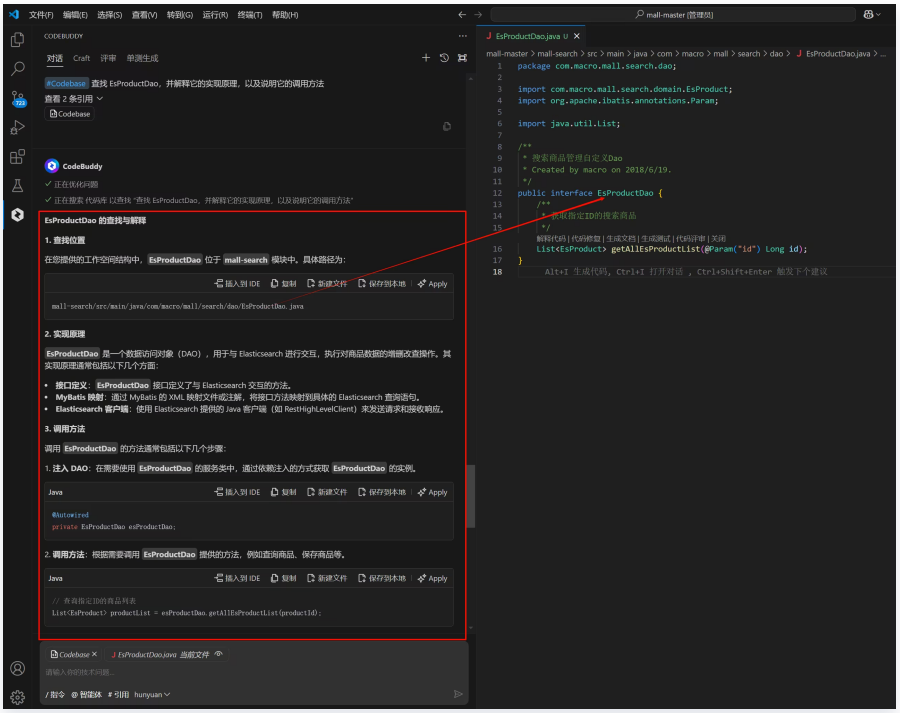
- **代码助手能正确找到某个函数/方法，并解释该函数/方法的实现原理以及调用方法。**

  输入提示词：“#Codebase 查找 EsProductDao，并解释它的实现原理，以及说明它的调用方法”，触发 #Codebase 能力。

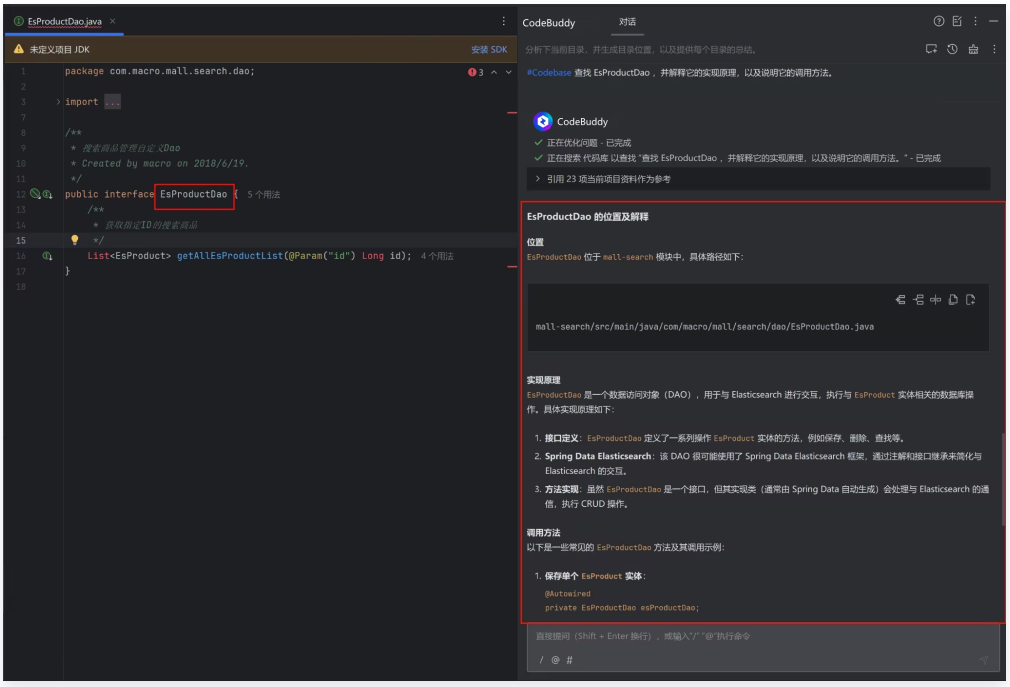


从下图的输出结果中，您可以看到，输出结果有召回内容，EsProductDao 方法能正确被找到，并解释了该方法的实现原理以及调用方法。

VS Code

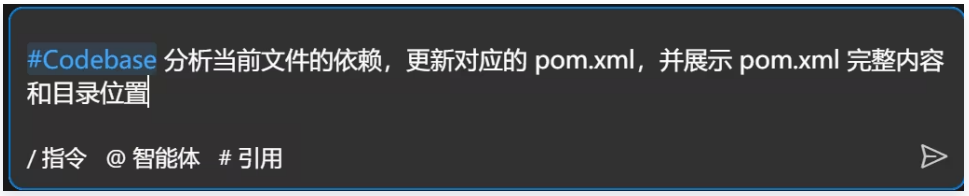


JetBrains IDE



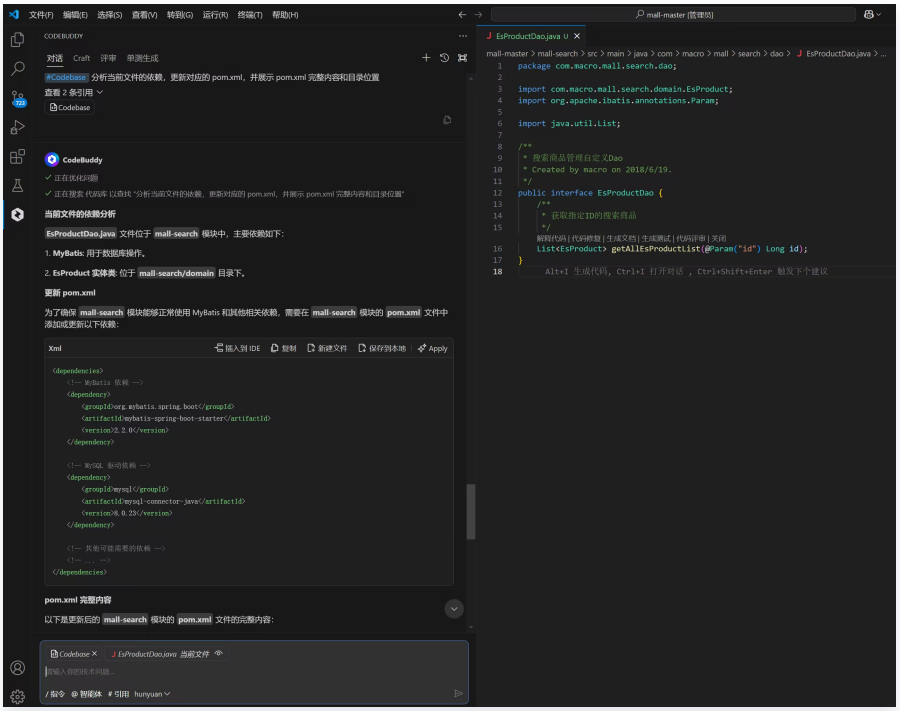
- **代码助手能分析当前文件的依赖，协助您了解当前文件依赖的信息，并更新对应的 pom.xml，确保项目配置的更新。**

  输入提示词：“#Codebase 分析当前文件的依赖，更新对应的 pom.xml，并展示 pom.xml 完整内容和目录位置”，触发 #Codebase 能力。

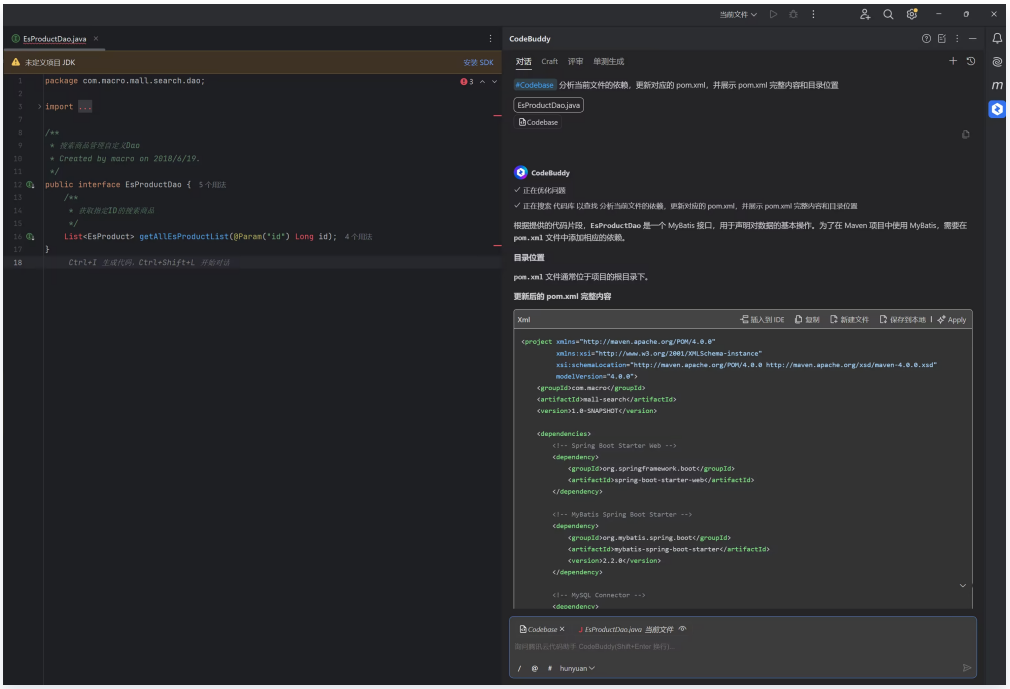


  从下图的输出结果中，您可以看到，输出结果有召回内容，能够分析当前文件依赖的信息，并展示更新后的 pom.xml 文件内容，以及文件所在的目录位置。

VS Code

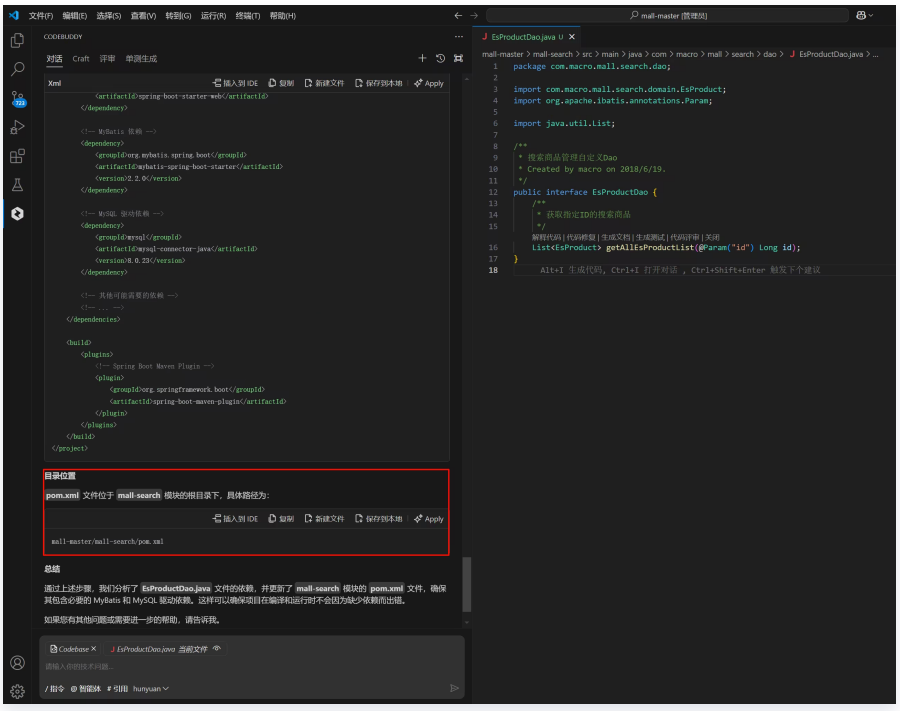


JetBrains IDE

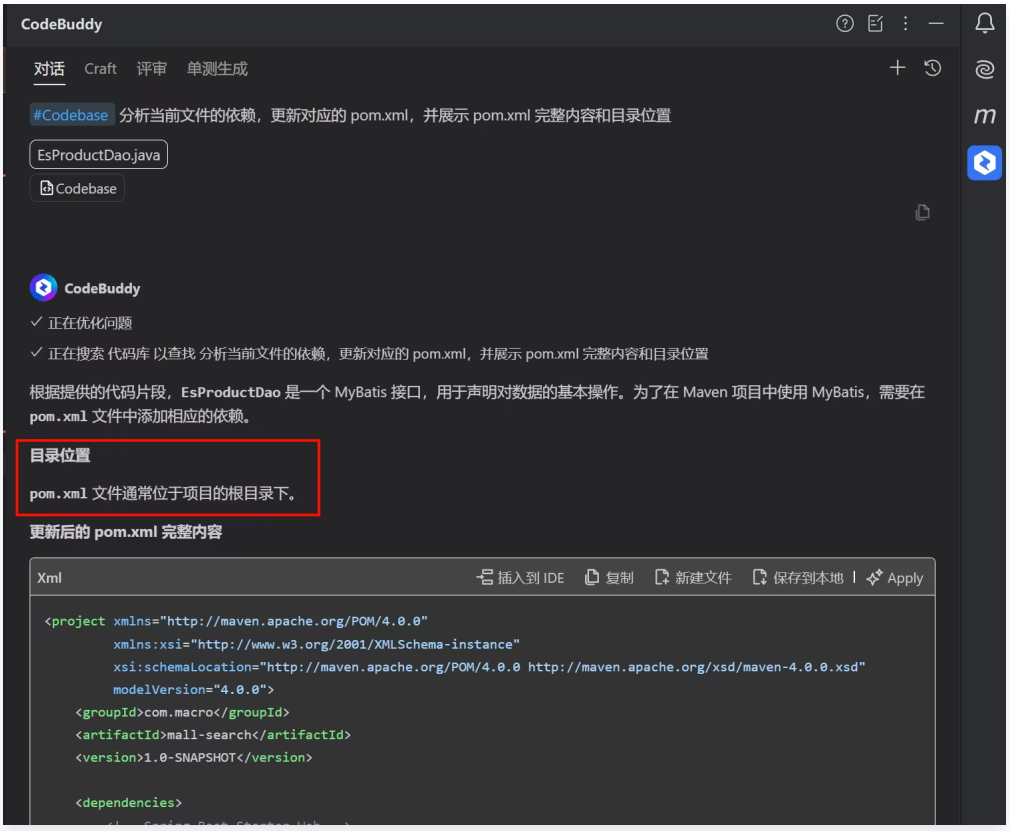


目录位置展示如下：

VS Code

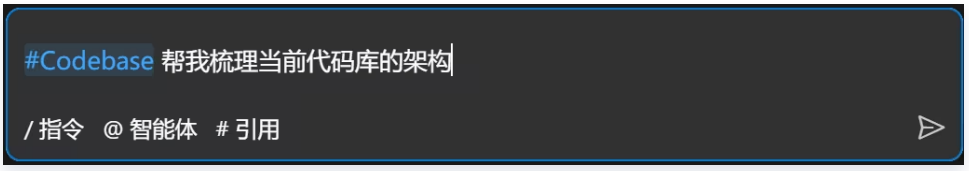


JetBrains IDE



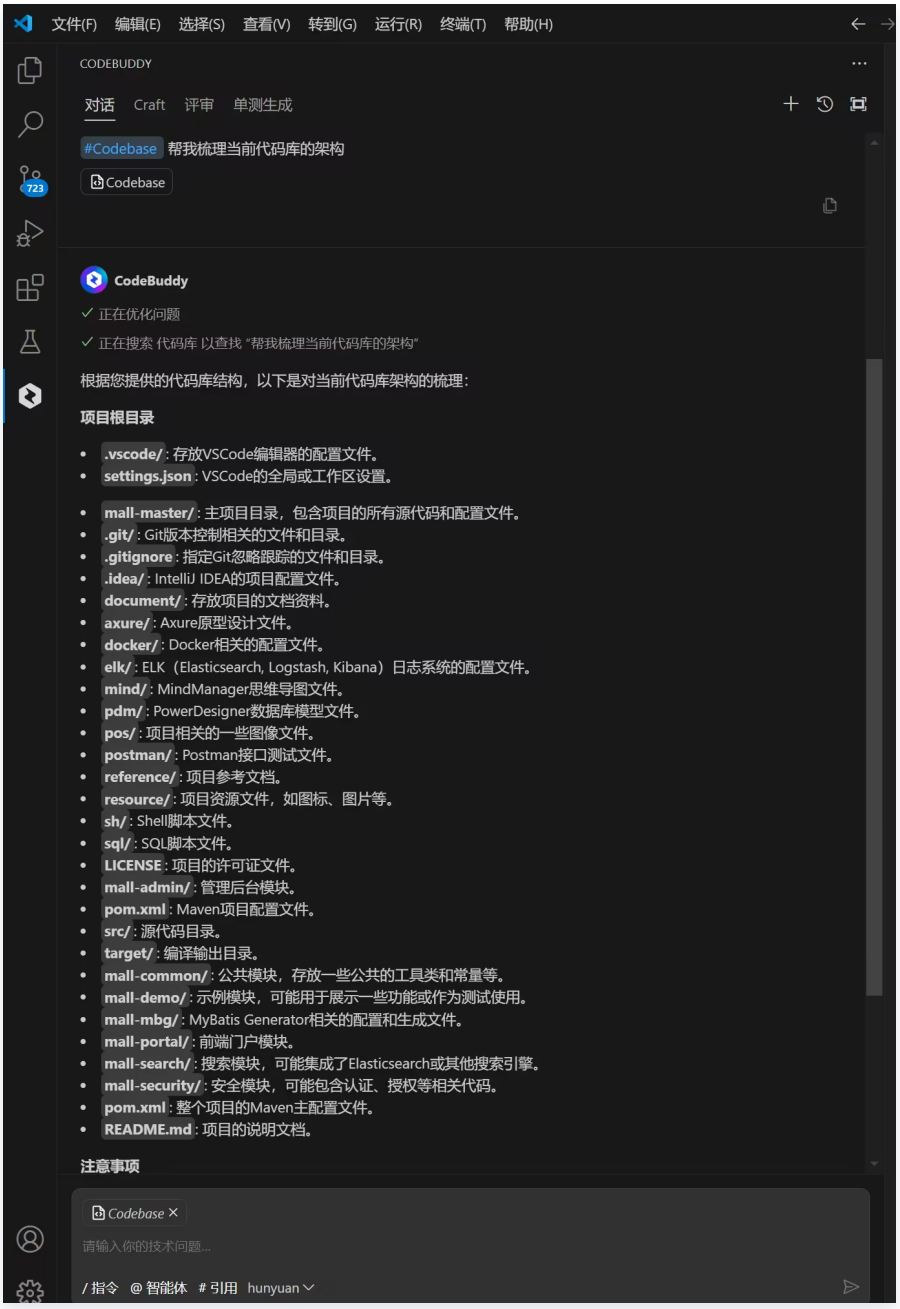
- 代码助手能分析当前项目的结构，方便您后续的开发和维护。

  输入提示词：“#Codebase 帮我梳理当前代码库的架构”，触发 #Codebase 能力。



从下图的输出结果中，您可以看到，输出结果有召回内容，能分析项目结构，并且对项目进行模块划分，分析关键部分，以及技术栈等，协助您了解项目的整体结构，便于后续的开发和维护。

VS Code



JetBrains IDE



#Codebase 帮我梳理当前代码库的架构

CodeBuddy

✓ 正在优化问题 - 已完成

✓ 正在搜索 代码库 以查找 "帮我梳理当前代码库的架构" - 已完成

› 引用 32 项当前项目资料作为参考

根据您提供的代码库结构，以下是对当前代码库架构的梳理：

一、项目整体架构

1. **根目录**：
   - `.vscode/`：包含VS Code的配置文件。
   - `LICENSE`：项目许可证文件。
   - `pom.xml`：项目的总Maven配置文件。
   - 多个子模块目录，如`mall-admin`、`mall-common`、`mall-portal`等。
   - 其他资源目录，如`document/`、`sql/`、`mind/`等。

二、主要子模块及功能

1. **mall-admin**：
   - 负责后台管理系统的开发。
   - 包含用户管理、角色管理、资源管理等核心功能。
   - 相关文件：`UmsAdminServiceImpl.java`（用户管理服务实现）、`UmsResourceServiceImpl.java`（资源管理服务实现）等。
2. **mall-common**：
   - 存放公共组件和服务。
   - 如日志配置、工具类等。
   - 相关文件：`logback-spring.xml`（日志配置文件）。
3. **mall-portal**：
   - 负责前端门户的开发。
   - 包含订单管理、商品管理等前端展示功能。

RAG 知识库

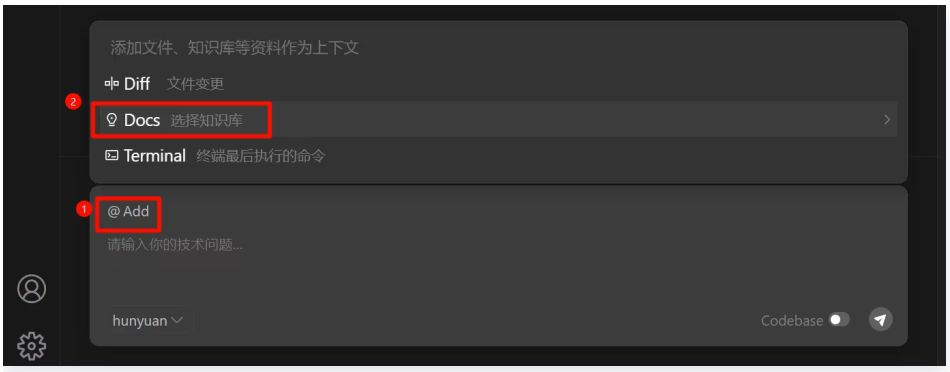
最近更新时间：2026-07-20 19:49:31

RAG 功能简介

RAG（Retrieval-Augmented Generation），这一检索增强生成技术，与腾讯云代码助手的结合，为用户带来了前所未有的智能体验。它不仅基于大语言模型的海量知识，还随时能接入专业打造的“百科全书”级知识库。这使得腾讯云代码助手在提供研发问答和智能编码服务时，更加精准、全面，有效减少部分模型幻觉，助力开发者高效解决问题，提升编码效率。接下来介绍一下使用流程。

基于预置知识库，快速体验 RAG

代码助手中的 @ Docs



代码助手中的 @ Docs 是指更多类型的上下文，Extra Context Info。我们对 @ Docs 的触发位置没有要求，您可以把 @ Docs 知识库放在对话描述的任意位置。
在对话输入框中，单击 @ Add 然后选择 Docs ，或者输入 @ 然后选择 Docs 调用预置的知识库。 知识库涵盖主流的代码仓库集合：

- 腾讯云实时音视频：实时音视频接口文档、辅助迁移文档。
- UnrealEngine 4.5：UnrealEngine 4.5版本知识库。
- TDesign：TDesign 组件 API 知识库 。
- Spring Boot：Spring 开发框架。
- Spring AI：基于 Spring 的 AI 应用框架。
- LangChain：基于 LangChain 的 AI 应用工具包。
- React：前端开发框架。
- Vue：前端开发框架。
- 腾讯云 API：腾讯云 API 开发知识库。
- LeaferJS：前端 Canvas 2D 引擎。
- 微信支付：微信支付开发知识库。
- 微信小程序：微信小程序开发知识库。

版权所有：腾讯云计算（北京）有限责任公司

- 微信小游戏：微信小游戏开发知识库。
- 微信云开发：微信云开发+云托管。
- 云原生构建：CNB 知识库。
- 腾讯地图小程序：腾讯地图小程序知识库。

如果您有更好的仓库或知识库期望预置，可以 联系我们 提出建议。

适用版本

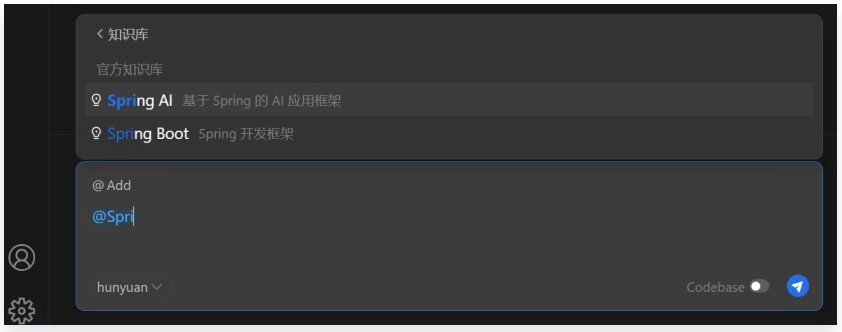
腾讯云代码助手个人版、企业版，点击前往 版本说明 。

VSCode 使用步骤

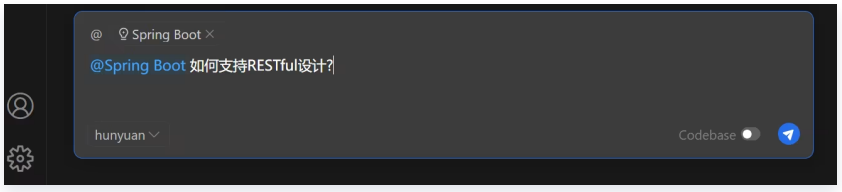
打开侧栏腾讯云代码助手，并在如下的输入框里选中上述内置的知识库。我们提供两种快速输入方式：

通过键盘输入

1. 直接输入 @ 知识库名字。为追求体验，当您输入时我们就启动过滤知识库。可以通过键盘上下键，选中后回车即可。

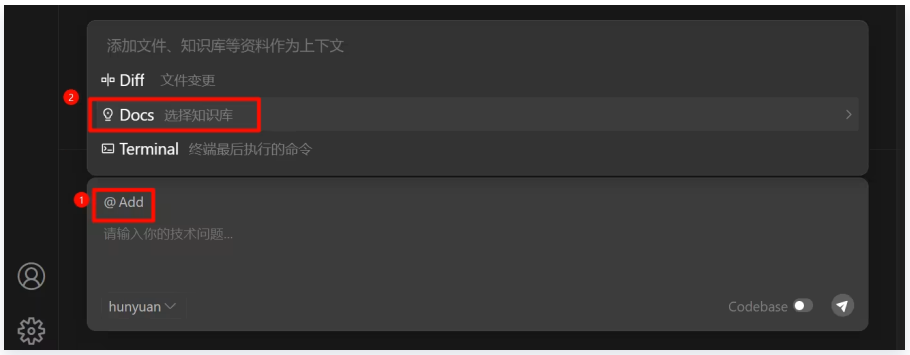


2. 选择知识库后，在输入框上方的操作条中会展示所添加的知识库，接下来输入问题描述即可引用知识库作为上下文，例如下图所示。



通过鼠标单击

1. 鼠标单击 @ Add，选择 **Docs**，然后再选择您需要的知识库。



2. 选择**知识库**的数据类型后，我们会列出所有内置的知识库。这里的知识库，是一系列相关的仓库的合集，包括官方知识库和自定义知识库。

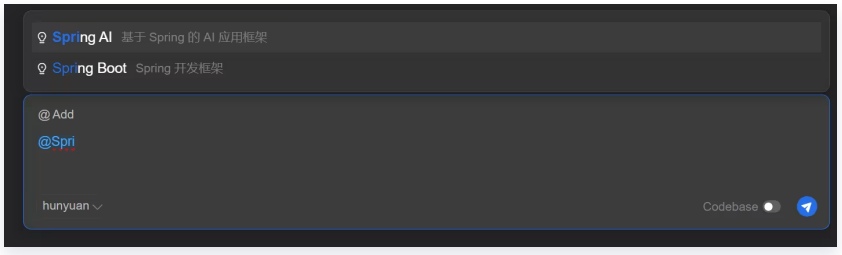


JetBrains 使用步骤

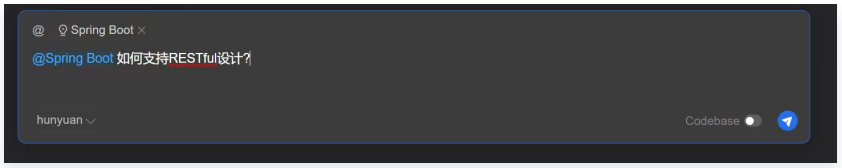
同 VSCode 的操作一样，您可以通过键盘或者鼠标的两种方式选中内置的知识库。

通过键盘输入

1. 输入 @ 知识库名字，然后通过键盘上下键，选中知识库后回车即可，如下图。

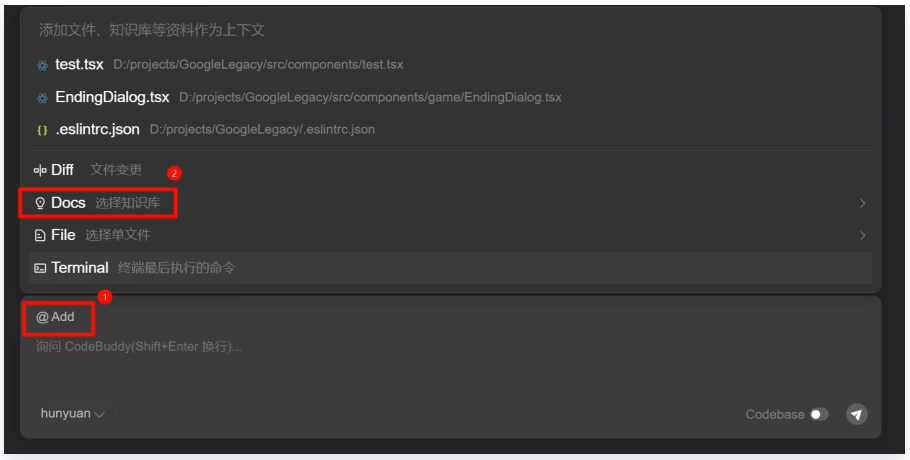


2. 选择知识库之后，在输入框上方的操作条中会展示所添加的知识库，接下来输入问题描述即可，例如下图所示。



通过鼠标输入

1. 鼠标单击 @ Add，选择 **Docs**，然后再选择您需要的知识库。

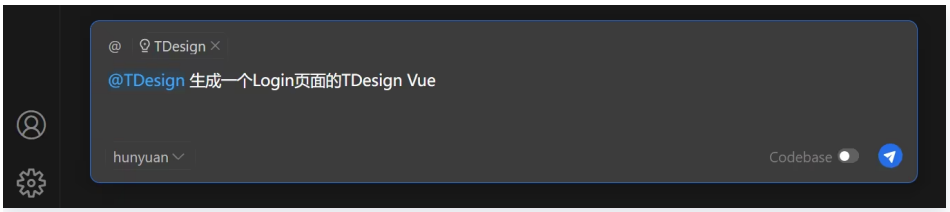


2. 选择知识库的数据类型后，会列出所有的知识库，包括官方知识库和自定义知识库。

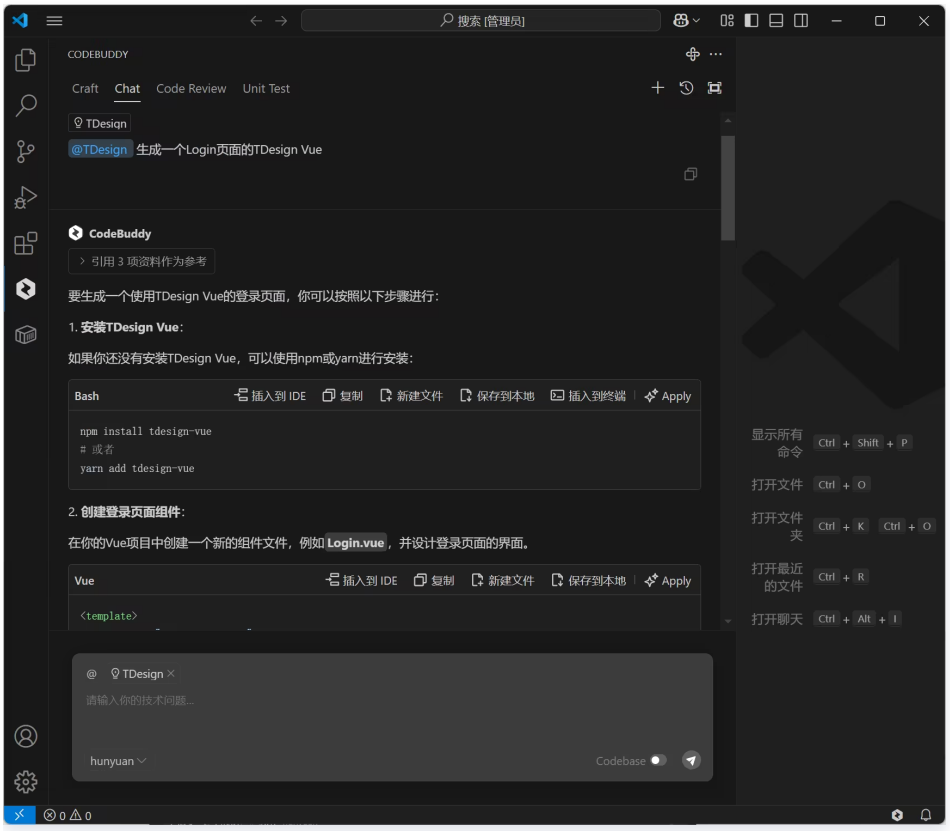


执行过程

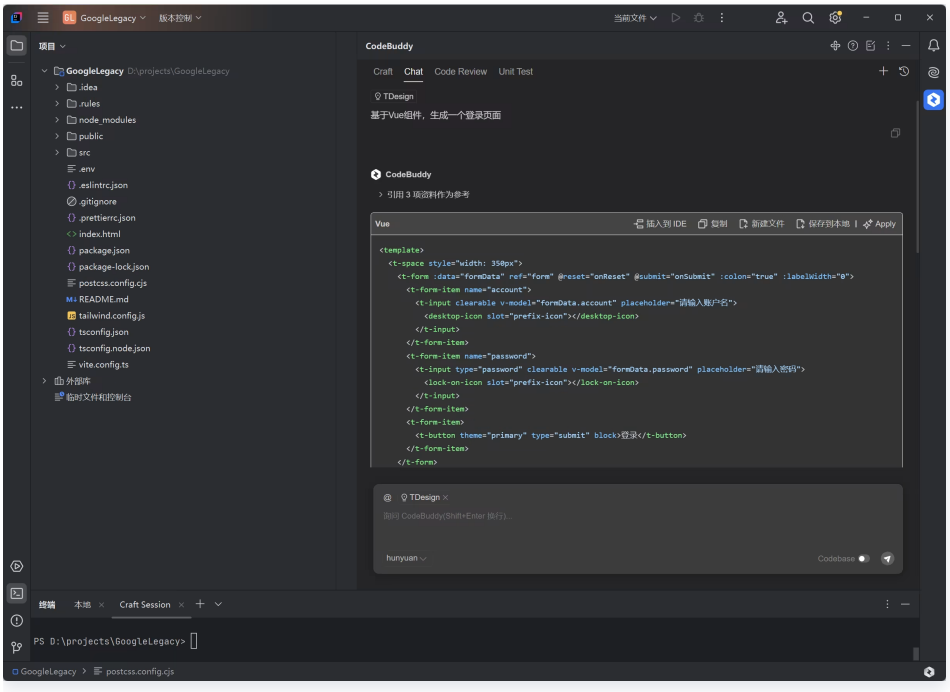
您可以在对话框中的任意位置选择所需的知识库，以基于 RAG 技术加强本次提问的结果。回车后，腾讯云代码助手会根据提问和选定的知识库首先检索与问题相关的资料作为参考，然后将这些资料和提问内容一同提交给大模型，从而生成比之前更精确的答案。



VSCode 的效果



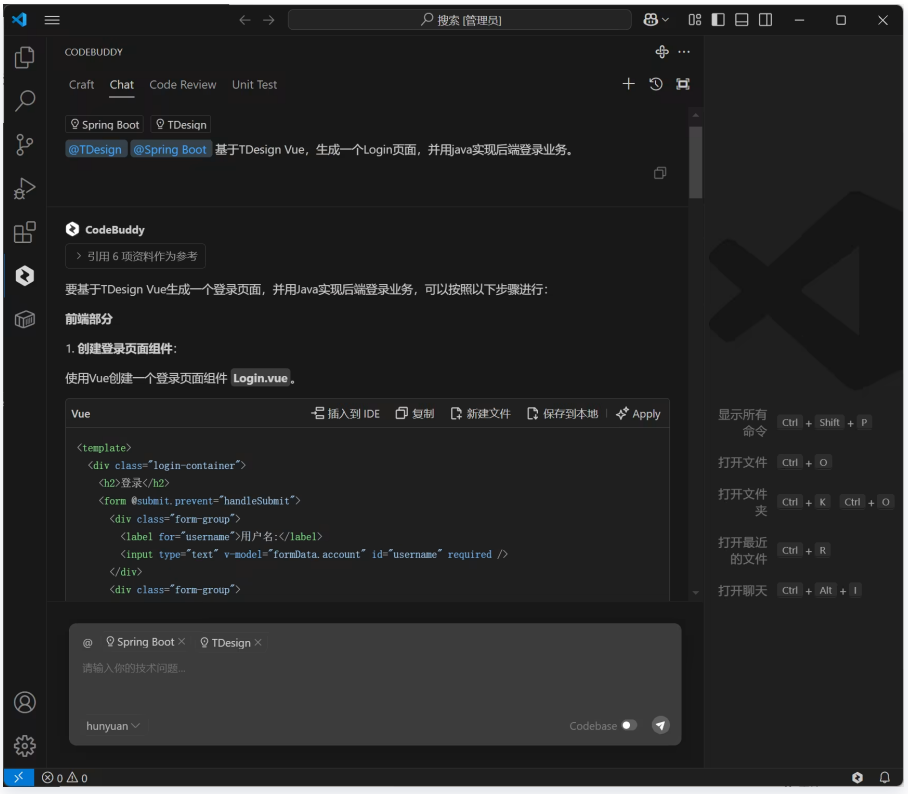
JetBrains 的效果



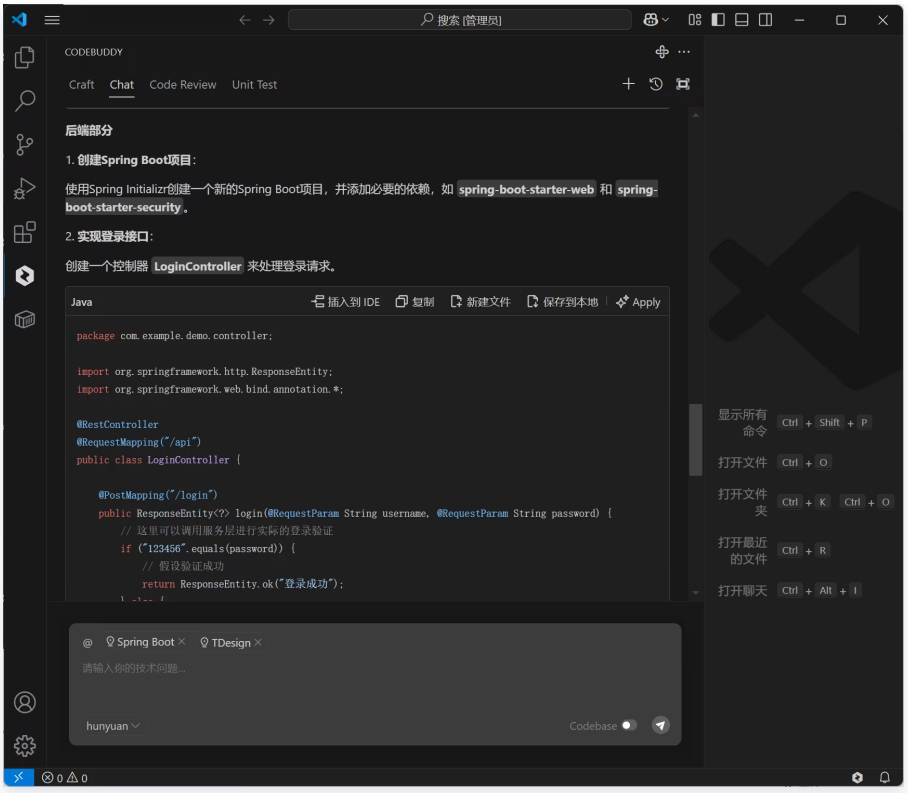
支持多个知识库同时选择

您还可以选择多个知识库。例如可以引入前后端知识库，并提问如下图所示，腾讯云代码助手会结合知识库的召回作为参考，并生成前端部分和后端部分。

- 前端：



- 后端：



通过自定义知识库，定制企业私有 RAG

腾讯云代码助手允许企业创建专属的自定义知识库，并且支持的文件类型有单文档、多文档、文档压缩包、离线代码库等。企业管理员可以将企业知识库中上传的文档和文件等内容整合起来，便于企业开发者在回答问题时将其作为上下文参考，从而使代码助手的回答更贴合企业特点。

创建自定义知识库

创建自定义知识库 只需要如下几步操作即可：

1. 创建知识库空间：
   - 输入名称（20个英文字符或10个中文字符）。
   - 描述（30个英文字符或15个中文字符）。
   - 可见范围：必选，默认是企业所有成员。可设置部分成员可见，支持输入搜索选择和在下拉选项列表中直接选择。

   单击**确定**创建完成，整体截图如下：

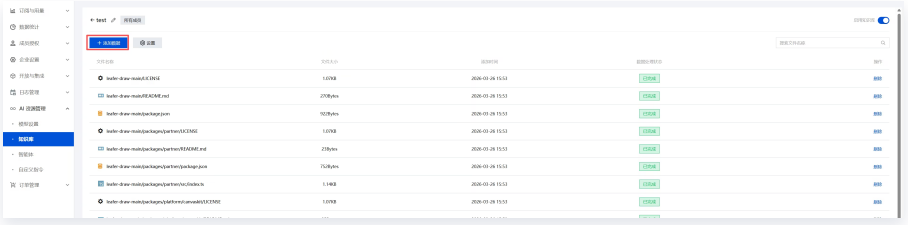


2. 上传文件：

   为当前知识库添加文件，支持以下类型：
   - 文档：支持 .md、.markdown、.docx、.pdf 格式，以及.py、.js、.css 等代码类型文档，每个文档不超过 30 MB。
   - 压缩包：支持 .zip、.tar、.tgz、.tar.gz、.gz、.gzip 格式，每个压缩包不超过 300 MB。
     - 文件内容命名要遵循 utf-8、GBK 编码，暂时不支持其他格式的编码。

> ⚠ **注意：**
> 限免期支持的类型有限，会在近期新版本中逐步放开文件大小和类型的限制。

单击**添加数据**，进入添加文件页面。

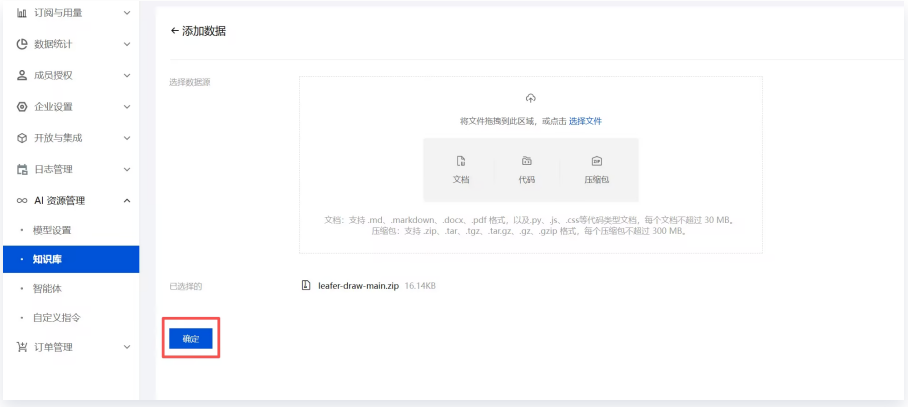


支持**文件拖拽**和单击**选择文件**两种交互。

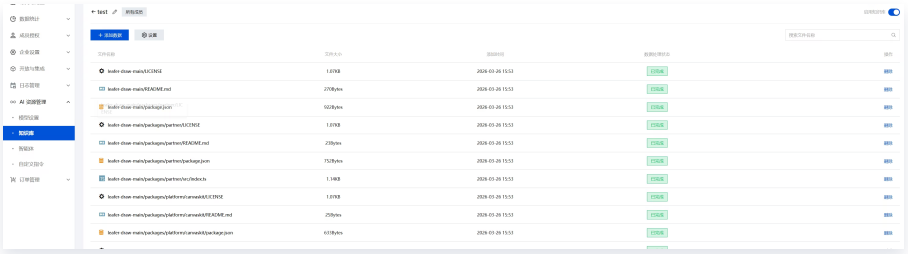


以这个 GitHub 仓库为例：

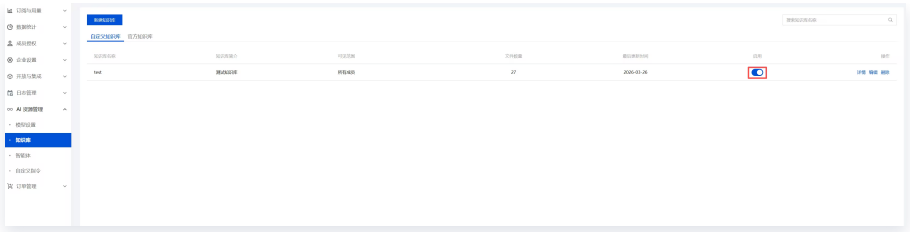
单击并下载 ZIP 包然后拖拽到当前页面，单击**确定**，会进行后端解压，如果遇到限制会给出失败原因。没有问题后则上传成功，并返回到知识库的首页展示文件列表。每个文件都有直观的索引状态展示。



3. 等待索引完毕后开启知识库。
   - 数据处理状态：索引中、索引失败、已完成。
   - 索引中和索引失败的数据不可启用。
   - 索引完成的数据默认启用。



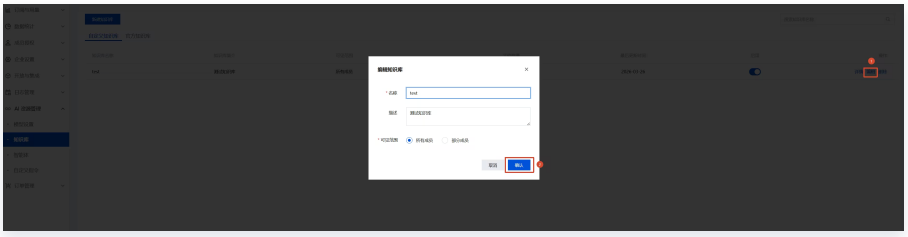
4. 返回知识库首页，开启知识库。



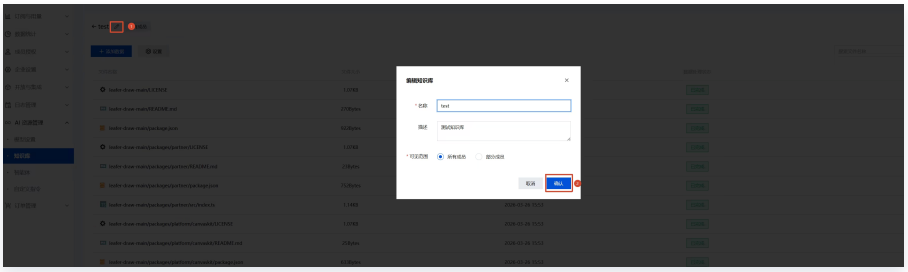
更新自定义知识库

通过以下两种方式修改知识库的基本信息，包括知识库的名称、描述；也可以对已经有的知识库进行文件的添加。

- 从知识库列表编辑：在知识库列表中选择对应知识库，单击**编辑**，即可进入知识库编辑模式。



- 知识库内页面编辑：进入对应知识库内，单击名称边上的按钮进行编辑，回车即完成保存。同时，也可以添加数据或对知识库进行设置。



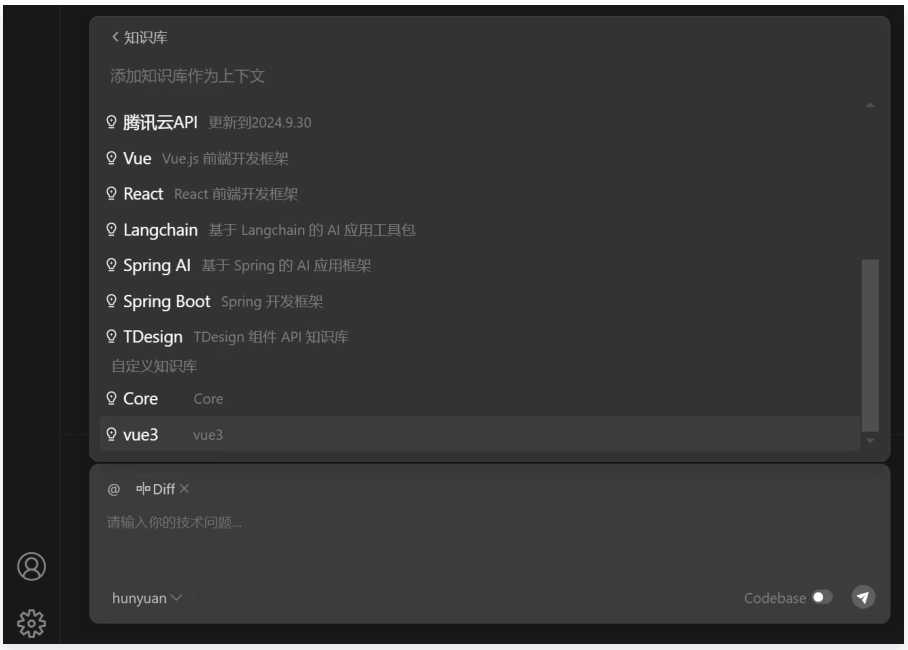
使用自定义知识库

版本要求

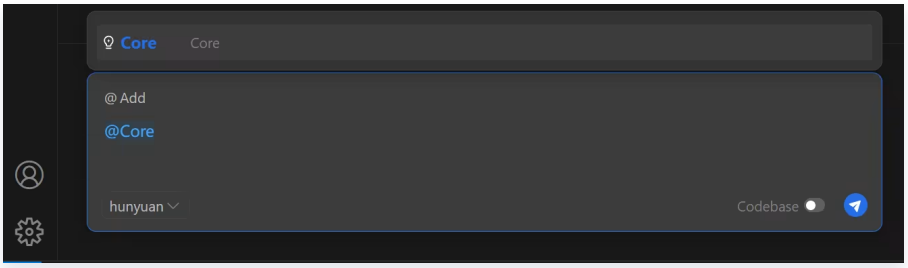
- **适用版本**：腾讯云代码助手旗舰版、专享版、企业版，点击前往 版本说明 。
- **插件版本**：需要升级到最新版本。

VSCode 使用步骤

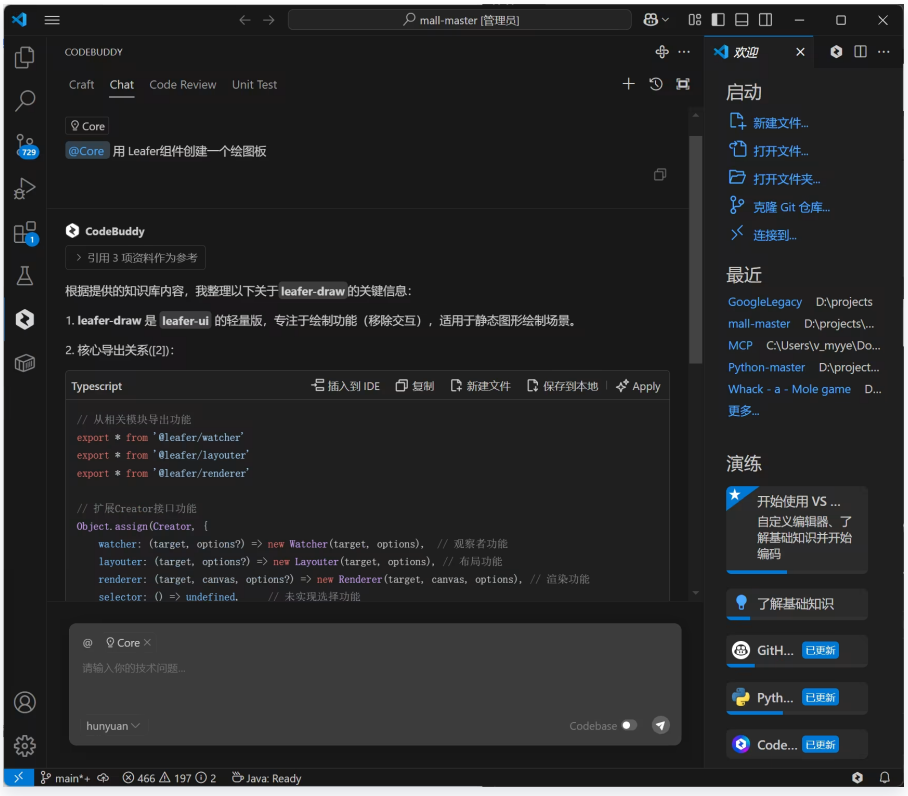
1. 当前用户如果处于企业组织，且企业组织下有自定义知识库，那么 @ Docs 知识库下就会出现**自定义知识库**。如下：



2. 您可以通过键盘或者鼠标的两种方式选中内置的知识库。

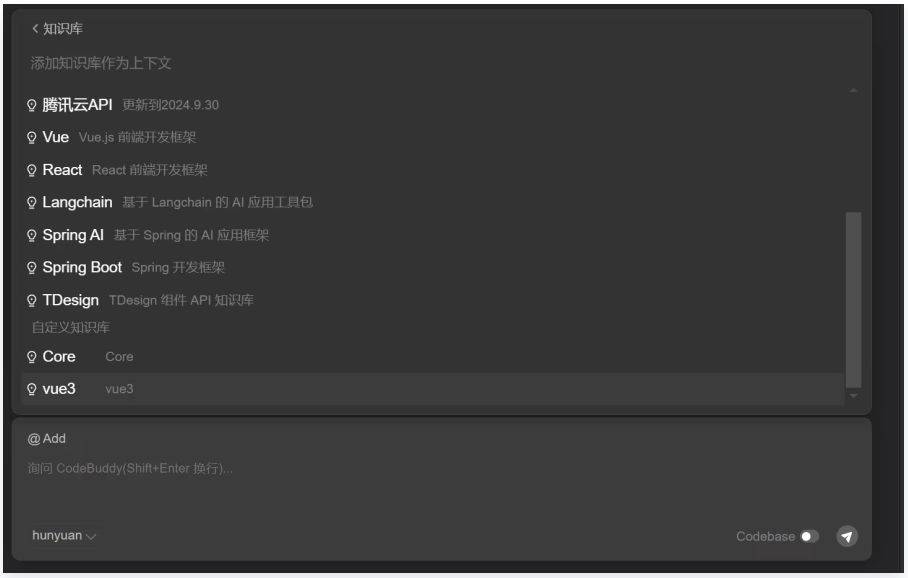


3. 效果如下。

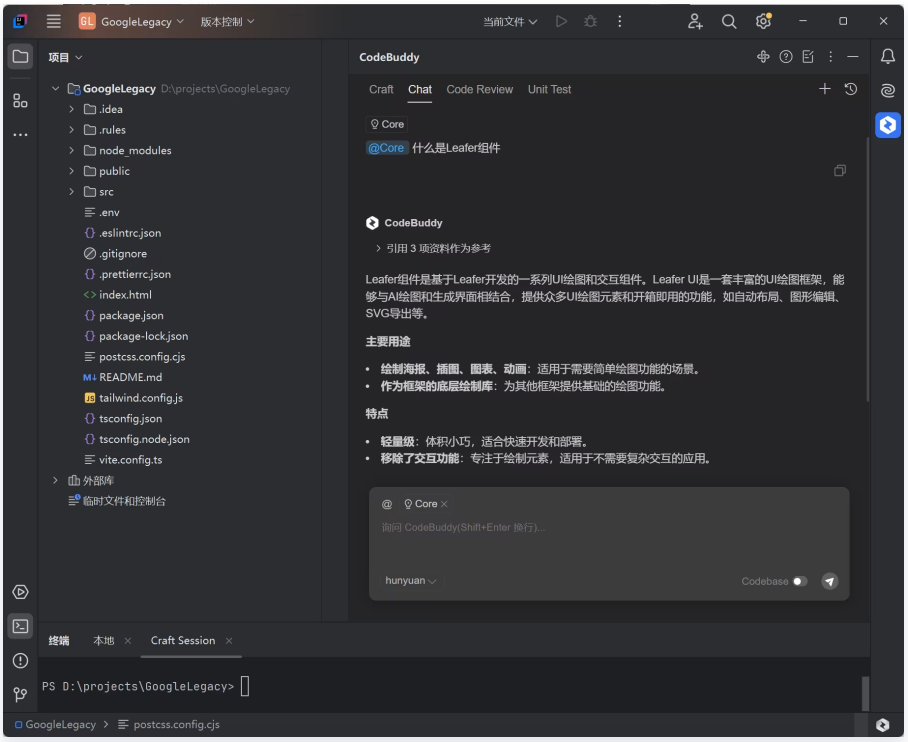


JetBrains 使用步骤

1. 当前用户如果处于企业组织，且企业组织下有自定义知识库，那么 @ Docs 知识库下就会出现**自定义知识库**分类。如下：



2. 执行后的效果如下：



MCP

最近更新时间：2026-07-20 19:49:31

介绍

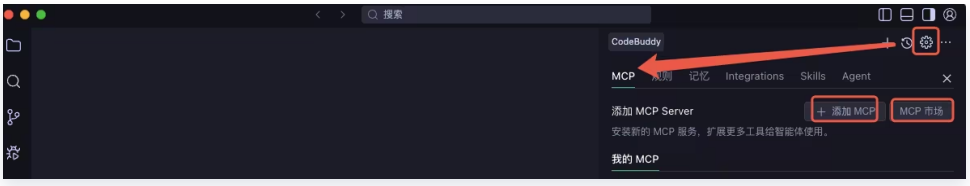
MCP (Model ContextProtocol，模型上下文协议)，旨在解决大模型语言（LLM）与外部数据源及工具之间无缝集成的需求。它通过标准化 AI 系统与数据源的交互方式，帮助模型获取更丰富的上下文信息，从而生成更准确、更相关的响应。其主要的功能如下：

- 上下文共享：应用程序可以通过 MCP 向模型提供所需的上下文信息（如文件内容、数据库记录等），增强模型的理解能力。
- 工具暴露：MCP 允许应用程序将功能（如文件读写、API 调用）暴露给模型，模型可以调用这些工具完成复杂任务。
- 可组合的工作流：开发者可以利用 MCP 集成多个服务和组件，构建灵活、可扩展的 AI 工作流。
- 安全性：通过本地服务器运行，MCP 避免将敏感数据上传至第三方平台，确保数据隐私。

CodeBuddy IDE 支持您进行 MCP Server 配置，扩展您的应用程序的功能。

配置步骤

1. 在 CodeBuddy IDE 侧栏对话面板右上方，点击 CodeBuddy Settings 按钮。
2. 切换到 MCP 标签页。目前支持自定义配置 MCP Server，同时支持在 MCPMarket 中一键安装 MCP Server。



一键安装 MCP Server

- 在 MCP Market 中提供了大量的 MCP Server，您可以一键进行安装。
- 根据您的实际需求，选择 MCP Server 进行一键安装。安装成功后，MCP Server 显示绿色状态，如果安装失败将显示红色状态。

自定义配置 MCP Server

与 Cursor、Claude Desktop、Cherry Studio 等这些 MCP Host（支持了 MCP 的应用程序）一样，腾讯云代码助手也提供了配置 MCP Server 的入口。
下面以 Time MCP Server 为例，为您介绍如何进行本地 MCP Server 配置，连接 `Time MCP Server`。

步骤1：安装依赖环境

在开始安装 Time MCP Server 之前，请确保您的开发环境满足具备安装 Time MCP Server 的包管理工具，常见的有 NPX、UVX 和 pip。

- **NPX**

  NPX 是 Node.js 的一个命令行工具，用于直接运行 npm 包中的命令，无需全局安装或显式指定路径，安装 Node.js 就默认自带这个工具。如果您未安装，请进行安装 Node.js 。安装好后，可以用以下命令查看是否安装成功：

  ```
  node -v   # 查看 Node.js 版本
  npm -v    # 查看 npm 版本
  ```

- **UV**

  uvx 是 uv 工具链的扩展命令，是一个用 Rust 编写的极快的 Python 包和项目管理器。具体的安装方式请参见 uv 安装 。

- **pip**

  pip 是 Python 的包管理工具，您可以访问 Python 官网 下载并安装最新版本 Python 3。安装完后可以使用如下命令查看是否安装成功。

  ```
  pip3 --version
  ```

  下面以 pip 的安装方式为例进行介绍。

  > **说明：**
  > 使用 pip 进行安装时，需要确保 Python 版本为3.10及以上版本。

步骤2：安装 Time MCP Server 包

通过以下命令安装 `Time MCP Server` 包。

```
pip3 install mcp-server-time
```

步骤3：配置 Time MCP Server

1. 在对话面板右上方，单击**设置** > **MCP** 按钮。



2. 如果没有已安装的 MCP Server，直接单击**配置 MCP** 或 下方的**配置 MCP Server** 进行配置。



3. 在 **mcp.json** 配置文件中添加 `Time MCP Server` **服务器配置。**



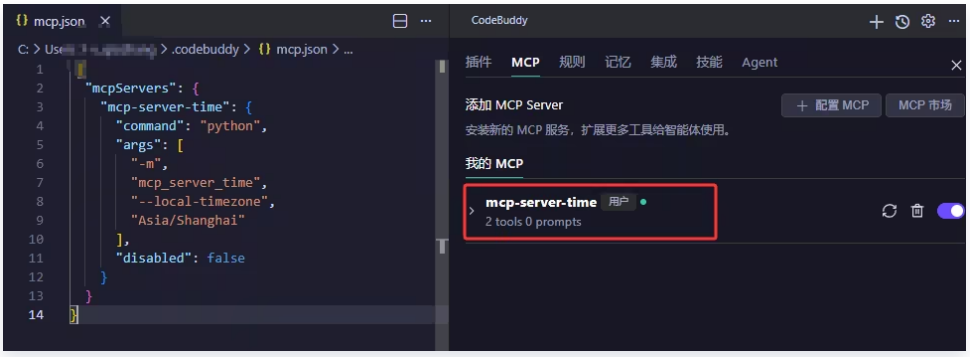
配置内容如下：

```
{
```

```
  "mcpServers": {
    "mcp-server-time": {
      "command": "python",
      "args": [
        "-m",
        "mcp_server_time",
        "--local-timezone",
        "Asia/Shanghai"
      ],
      "disabled": false
    }
  }
}
```

> **说明：**
> "args" 字段是命令参数，除了需要 " `-m` " 和 " `mcp_server_time` " 参数之外，还需要添加 "Asia/Shanghai"。当不提供 `--local-timezone` 参数时，服务器会使用系统默认时区（这里的例子中是“中国标准时间”），这个名称不是标准的 IANA 时区标识符，因此对于中国标准时区，还需要添加 "Asia/Shanghai" 作为中国标准时区的时区标识符。

4. 配置文件填写完成并保存后，可在 MCP Server 配置列表查看是否配置生效。`mcp-server-time` 服务器的状态为绿色，表示生效，红色表示未生效，如下图。



5. 在任意对话框中输入指令，当前时间，则可调用 MCP Server 输出结果。


CodeBuddy
我要创建一个 flask 框架的网...
我要创建一个 flask 框架的网站，请给我几个步骤和具体要
当前时间
CodeBuddy
让我获取当前精确时间：
获取 MCP Server 详细信息: mcp-server-time
mcp-server-time (get_current_time)
运行成功
当前时间是 2026年7月17日，星期五，16:51（北京时间）。
积分: 0
Tokens: 86,914
耗时: 4.5s

Craft 智能体
终端 shell integration 问题指引

最近更新时间：2026-07-20 19:49:33

Craft 在安装 MCP 服务器的过程中，在终端下执行命令时可能会出现 “Shell Integration Unavailable” 的问题，即 shell 集成不可用的问题。本文将针对此类问题，提供问题排查指导，帮助您排查问题。
本文以 VS Code 为例进行介绍，您可以按照如下方法进行排查。

> **说明：**
> 本文适用于 Windows、MacOS 和 Linux 操作系统。

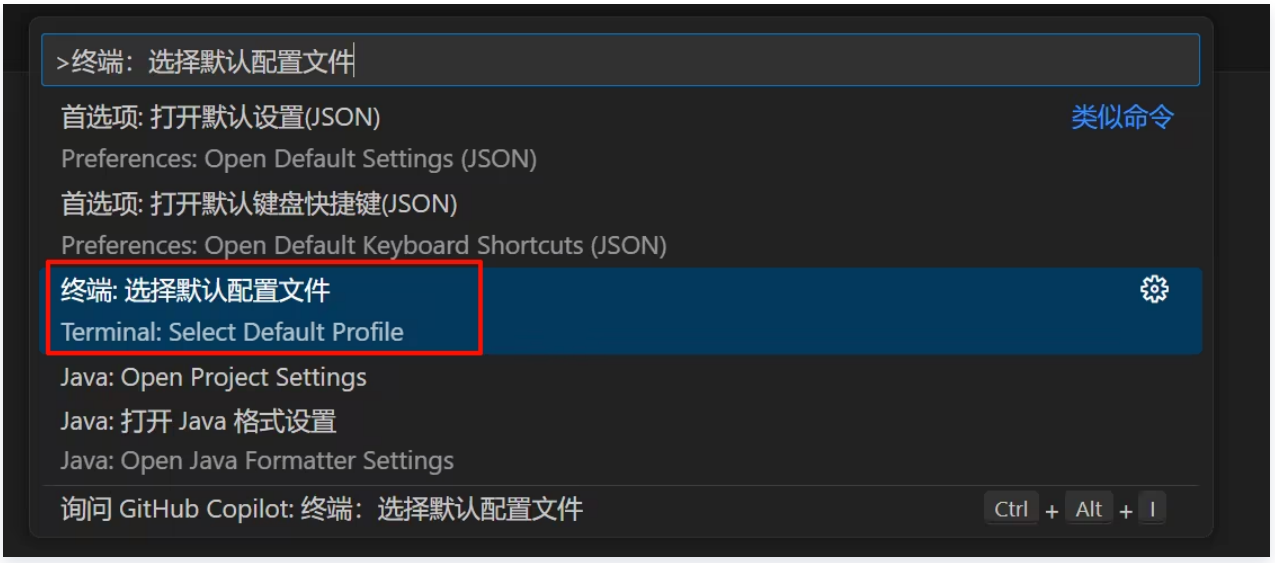
问题排查一：配置正确的 shell

步骤1：更新 VS Code，确保使用的是最新版本的 VS Code

1. 打开 VS Code。
2. 按 Cmd+Shift+P（Mac）或 Ctrl+Shift+P（Windows/Linux）。
3. 输入并选择 “检查更新” 或 “尝试更新” 。
4. 更新后重新启动 VS Code。

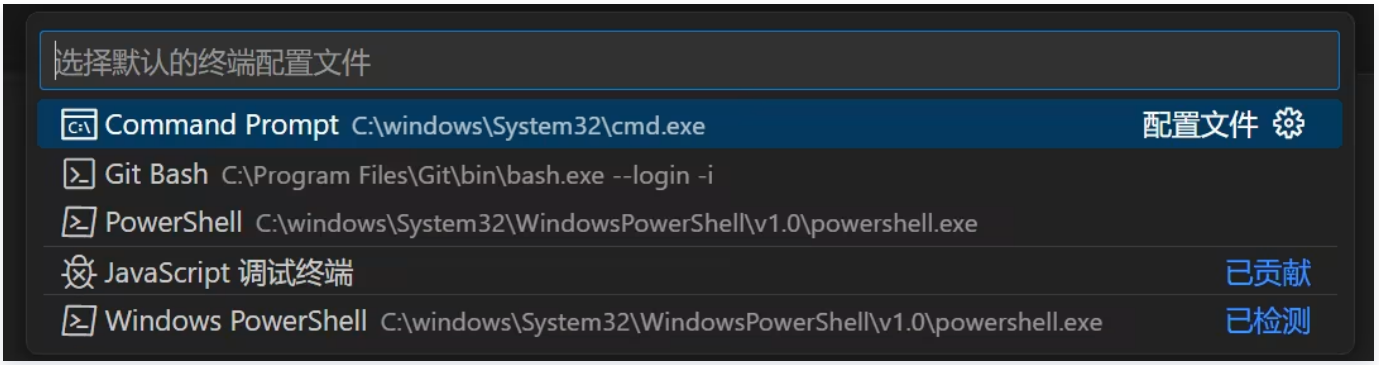
步骤2：在 VS Code 中配置使用正确的 shell

1. 打开 VS Code。
2. 按 Cmd+Shift+P（Mac）或 Ctrl+Shift+P（Windows/Linux）。
3. 输入并选择**终端：选择默认配置文件**。



4. 选择其中一个支持的 shell，例如：zsh、bash、fish 或 PowerShell。

以 Windows 为例，建议选择 PowerShell 或 Git Bash。



- Bash（添加到 ~/.bashrc）：

```
[[ "$TERM_PROGRAM" == "vscode" ]] && . "$(code --locate-shell-integration-path bash)"
```

- Fish（添加到 ~/.config/fish/config.fish）：

```
if test "$TERM_PROGRAM" = "vscode"
    source (code --locate-shell-integration-path fish)
```

> **说明：**
> 这里 PowerShell 的版本要求 v7及以上。您可以：
> - 输入并运行 `$PSVersionTable.PSVersion` 命令检查当前的 PowerShell 版本。
> - 如果您的版本低于 v7，请 更新 powershell v7 。

默认情况下，出于安全原因，PowerShell 会限制脚本执行。因此，您可能还需要调整 PowerShell 执行策略，使用执行策略来确定哪些脚本可以在您的系统上运行。以下是最常见的策略：

- `Restricted`：无法运行 PowerShell 脚本（默认设置）。
- `AllSigned`：所有脚本（包括本地脚本）都必须由受信任的发布者签名。
- `RemoteSigned`：本地创建的脚本可以运行，但必须对从 Internet 下载的脚本进行签名。
- `Unrestricted`：没有限制。任何脚本都可以运行，但在运行 Internet 下载的脚本之前，系统会警告您。

更改执行策略的步骤：

4.1 以管理员身份打开 PowerShell：按下 `Win+X` 并选择 “Windows PowerShell（管理员）” 或 “Windows 终端（管理员）”。

4.2 通过执行如下命令检查当前执行策略。如果不为 `RemoteSigned`，则可以继续下一步进行更改。

```
Get-ExecutionPolicy
```

4.3 通过执行如下命令，针对当前用户，更改执行策略。

```
# RemoteSigned：允许运行本地未签名脚本
# Scope CurrentUser：限定策略仅对当前用户生效
 Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
```

步骤3：重启 VS Code

重启 VS Code 后，重新开启新的 Craft 会话，或继续上一个会话并尝试再次运行命令，看是否能解决问题。当选择 PowerShell 时，如果问题仍然存在，则可以切换选择 Git Bash 再次尝试。

问题排查二：手动安装 shell 集成

如果您在尝试了以上基本的问题排除步骤后仍然遇到问题，可以参考 手动安装 shell 集成 。例如：

如果您使用的是 PowerShell

1. 在终端中运行如下命令以打开您的 PowerShell 配置文件。

```
code $Profile
```

2. 在文件中添加如下内容。

```
 if ($env:TERM_PROGRAM -eq "vscode") { . "$(code --locate-shell-
integration-path pwsh)" }
```

如果您使用的是 zsh

1. 在终端中运行如下命令以打开您的 zsh 配置文件。

```
code ~/.zshrc
```

2. 在文件中添加如下内容。

```
[[ "$TERM_PROGRAM" == "vscode" ]] && . "$(code --locate-shell-
integration-path zsh)"
```

版权所有：腾讯云计算（北京）有限责任公司

3. 保存文件后，请关闭并重新打开 VS Code 的终端窗口（或执行 `source ~/.zshrc` / `. ~/.zshrc`），以使配置生效。

> ⚠ **注意：**
> 仅当 TERM_PROGRAM 值为 vscode 时加载集成，避免影响其他终端环境。

异常终端输出

如果您看到带有矩形、线条、转义序列或控制字符的异常输出，则可能与终端自定义工具有关，常见的包括：

- Powerlevel10k：一个 zsh 主题，用于向提示符添加视觉元素。
- Oh My Zsh：管理 zsh 配置的框架。
- Fish shell themes。

您可以按照如下步骤进行排查解决：

1. 在 shell 配置文件中暂时禁用这些工具。例如：如果您在 Zsh 中使用 Powerlevel10k，则可以通过打开配置文件（`~/.zshrc`）注释掉文件中的相关行来禁用 Powerlevel10k。

```
# Comment out the Powerlevel10k source line
# source /path/to/powerlevel10k/powerlevel10k.zsh-theme
```

2. 如果问题解决，请逐个重新启用工具，以排查具体导致问题的工具，然后查找 VSCode 的 shell 集成功能兼容的替代工具。

版权所有：腾讯云计算（北京）有限责任公司

Craft 的使用

最近更新时间：2026-07-20 19:49:33

全新软件开发智能体 Craft，输入自然语言指令，支持文件多选，AI 深度理解，自主完成多文件代码生成和改写，即刻落地可执行应用。与 Ask 模式的对话智能体相比，跨文件上下文理解、批量代码生成与修改、智能代码优化与重构是 Craft 的核心功能，尤其是在生成代码工程中表现突出，能够显著提升开发者的效率。

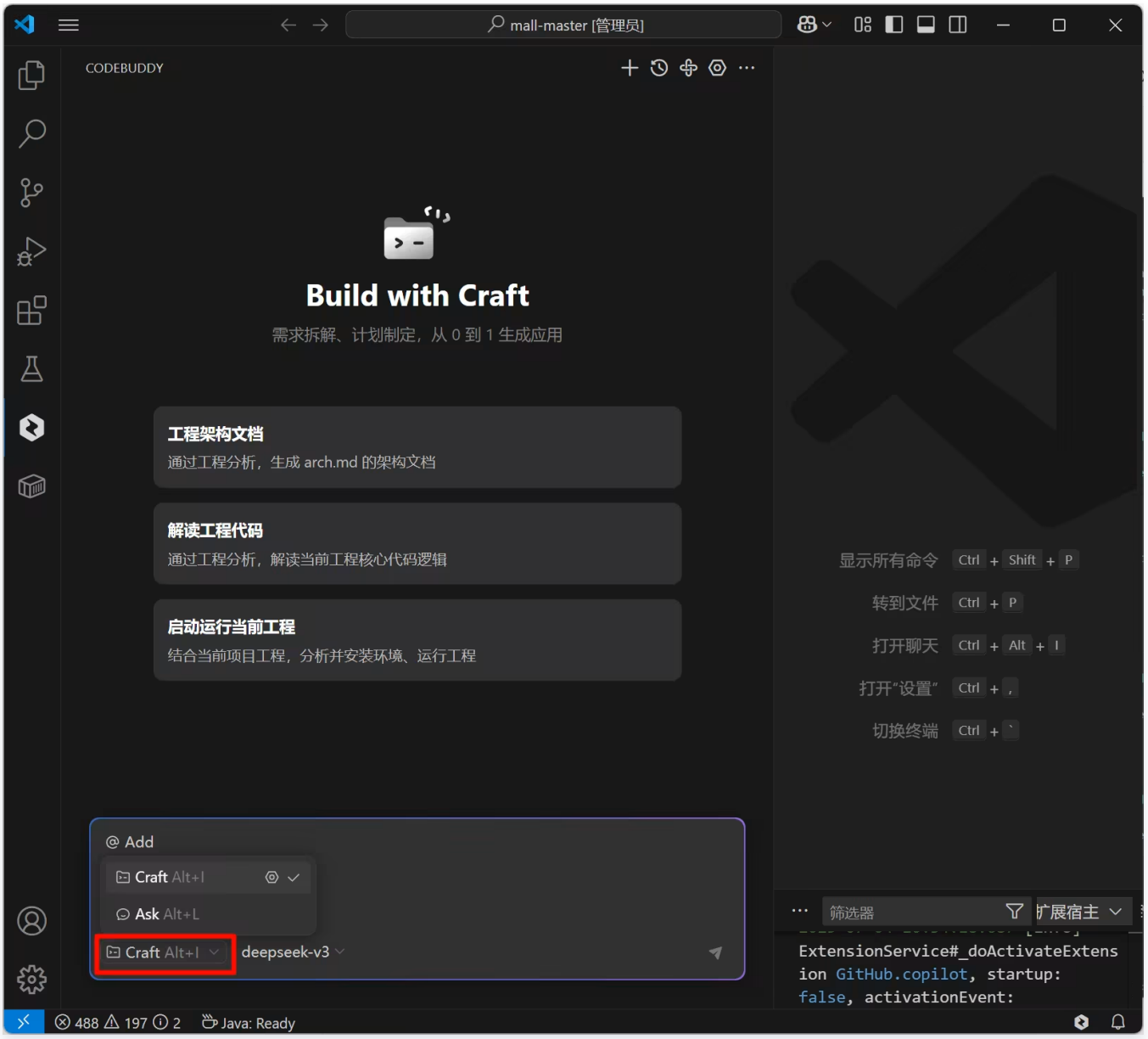
说明：

- Craft 支持的插件端：VS Code、JetBrains IDEs、微信开发者工具。
- Craft 支持的语言：市面上主流的开发语言都支持，例如 C、C++、C#、CSS、Go、HTML、Java、JavaScript 等。

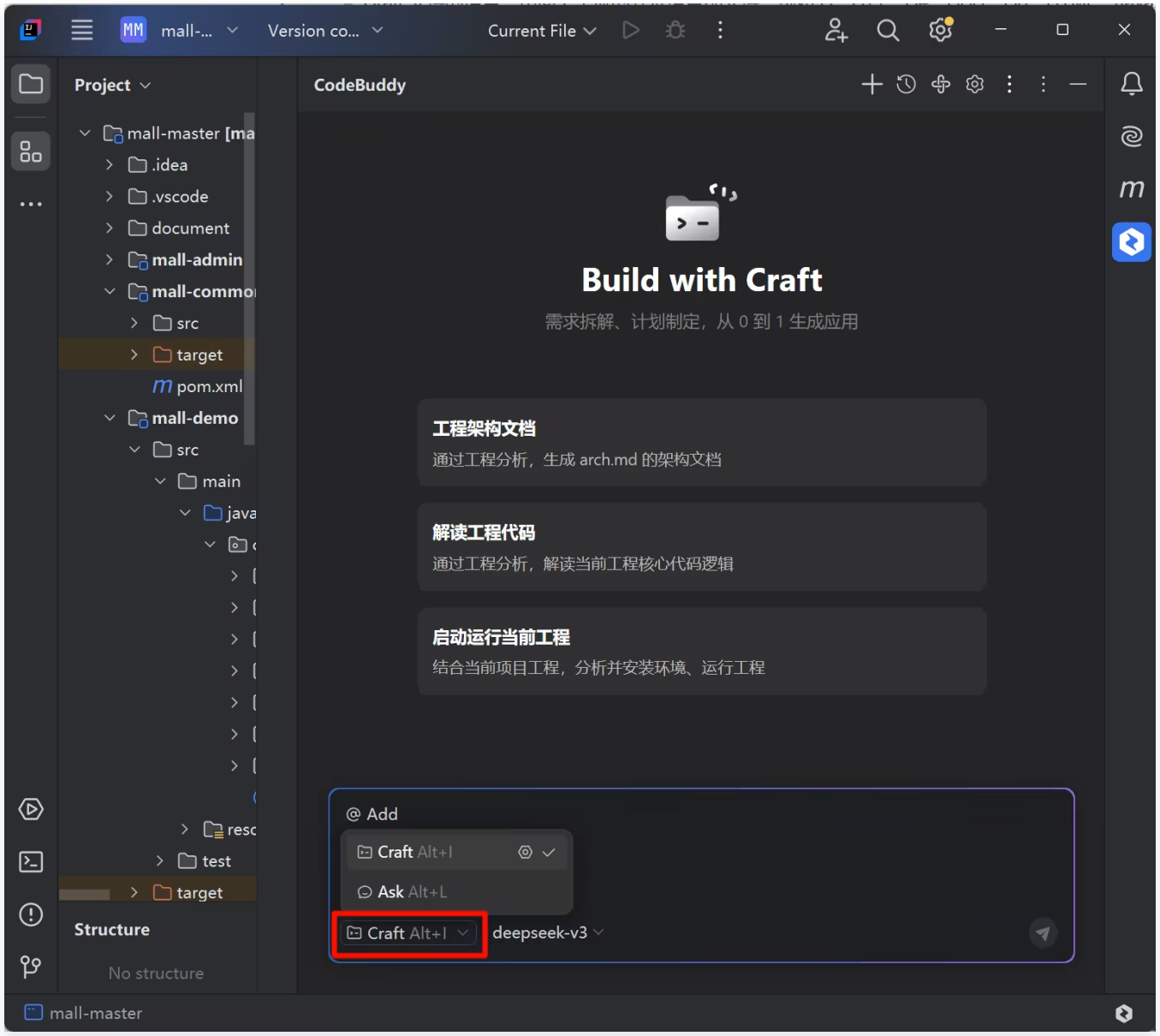
使用方法

1. 打开 Craft 模式。在代码助手对话输入框下提供 Ask 和 Craft 两种模式切换的入口，您可以单击切换到 Craft 模式，或者使用快捷键（Windows 为 Alt + I；macOS 为 ⌘ + I）快速切换。

VS Code



JetBrains IDEs

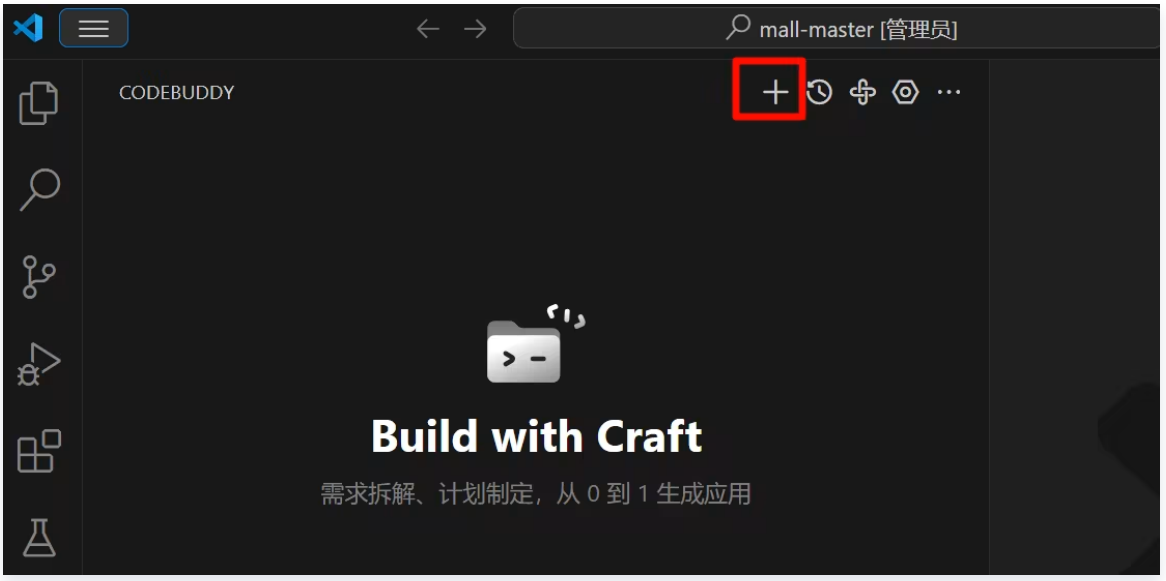

Project
CodeBuddy
Build with Craft
需求拆解、计划制定，从 0 到 1 生成应用
工程架构文档
通过工程分析，生成 arch.md 的架构文档
解读工程代码
通过工程分析，解读当前工程核心代码逻辑
启动运行当前工程
结合当前项目工程，分析并安装环境、运行工程
@ Add
Craft Alt+I
Ask Alt+L
Craft Alt+I
deepseek-v3
Structure
No structure
mall-master

微信开发者工具 IDE



2. **对话管理**。包括创建新对话和历史对话记录管理。
    - 创建新对话。您可以根据实际需求创建新对话。

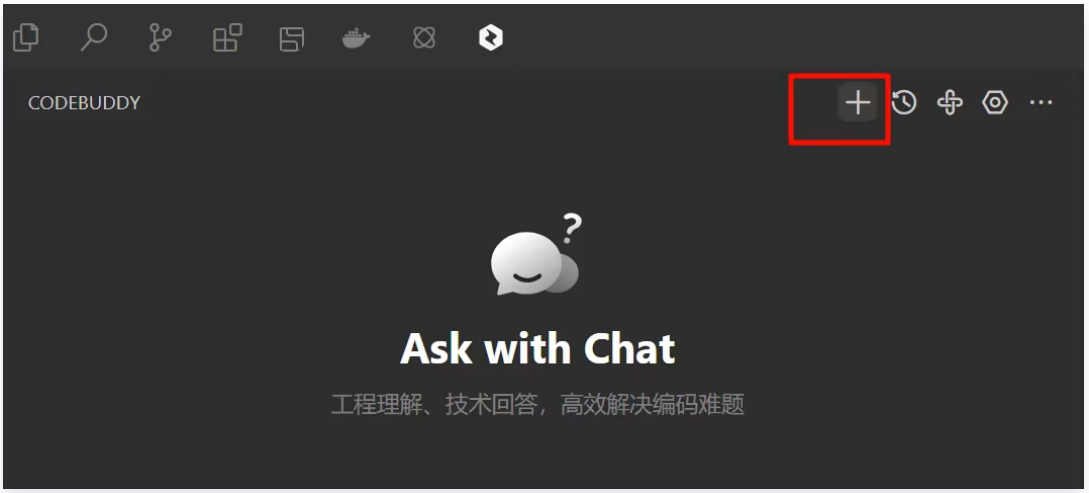
VS Code



JetBrains IDEs

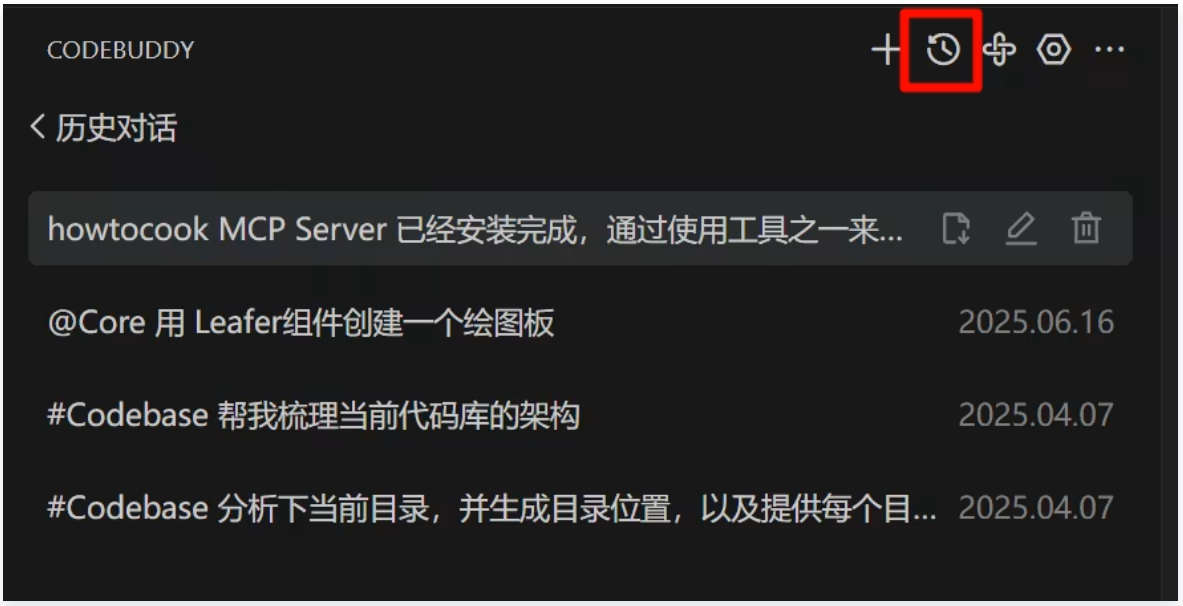


微信开发者工具 IDE

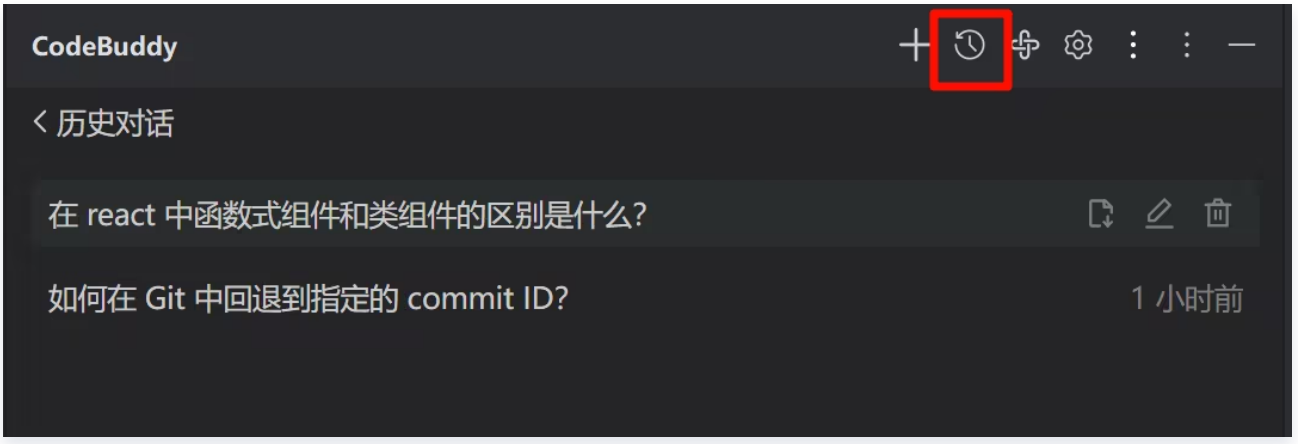


- 历史对话记录管理。在这里，您可以查看历史对话记录，并支持对历史对话记录进行导出、编辑和删除的操作。

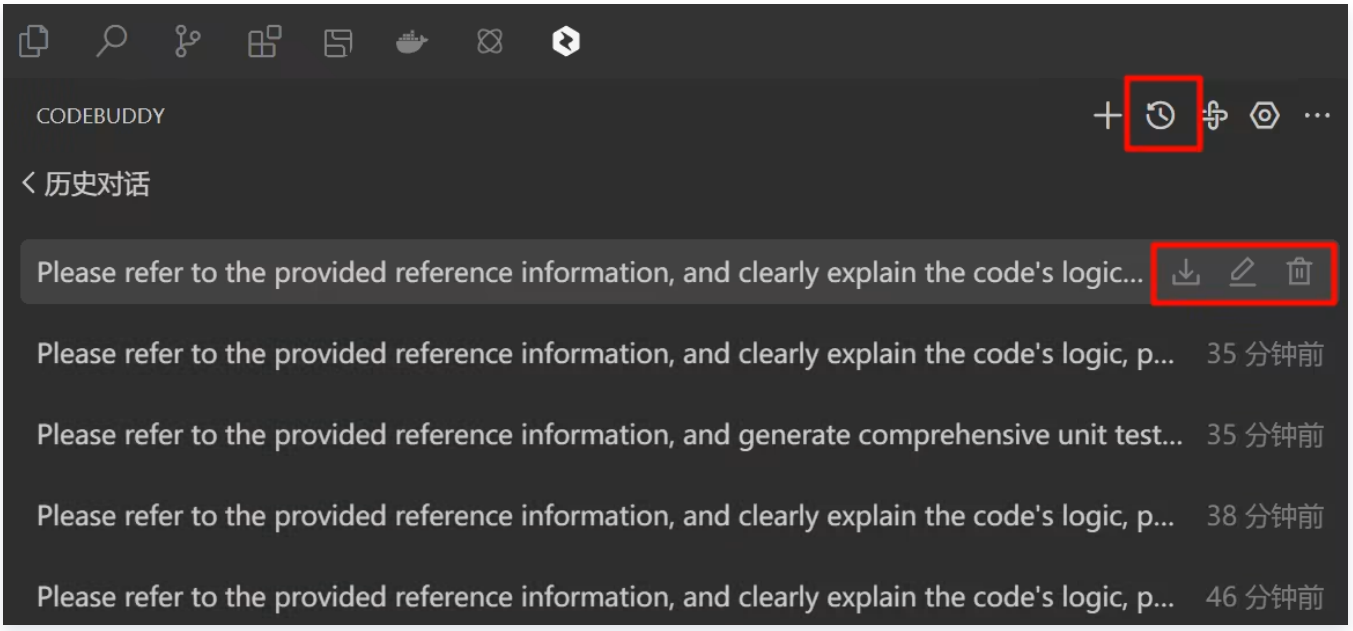
VS Code



JetBrains IDEs

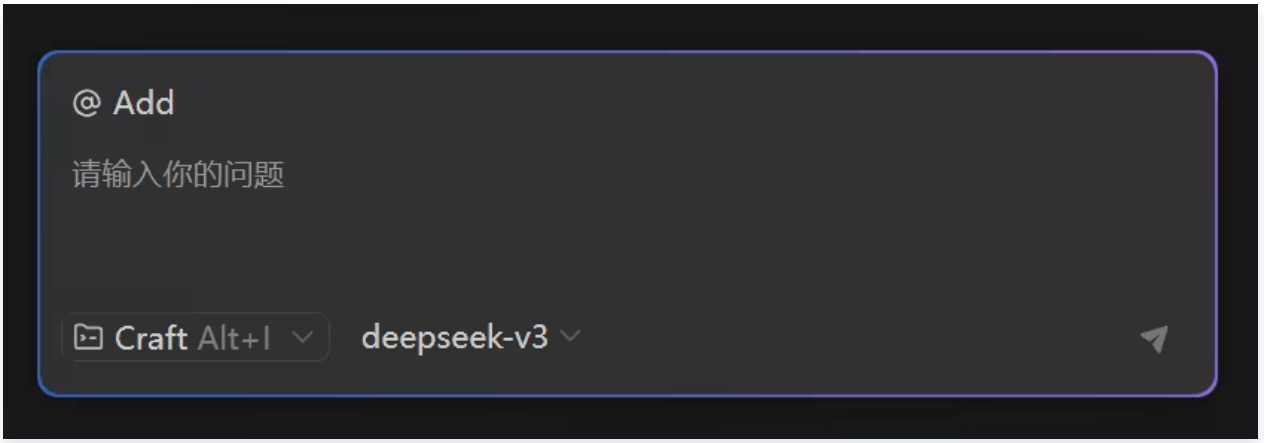


微信开发者工具 IDE

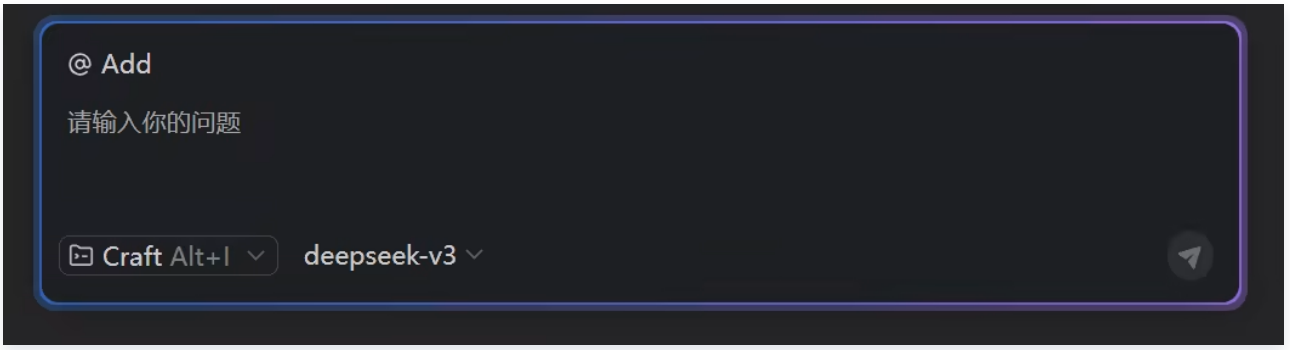


3. 在输入框中输入需求描述。

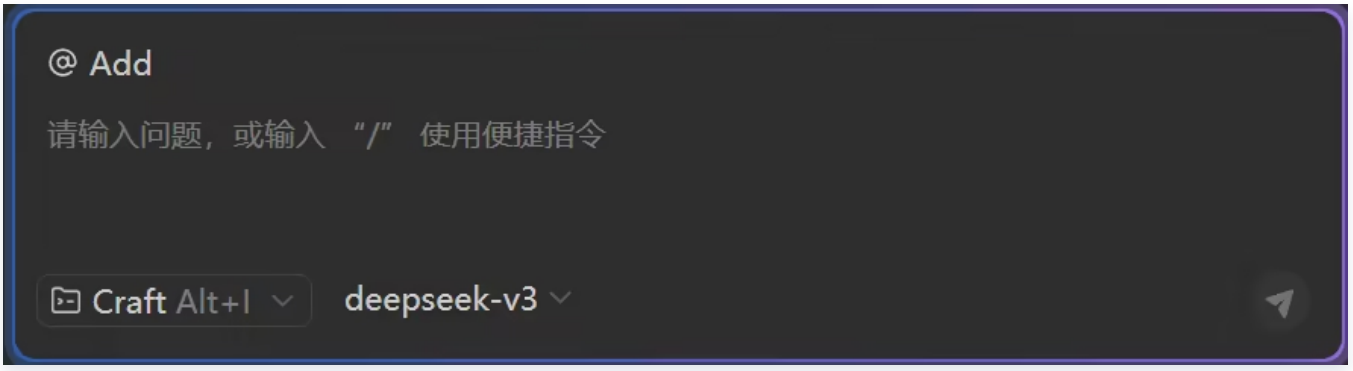
VS Code



JetBrains IDEs

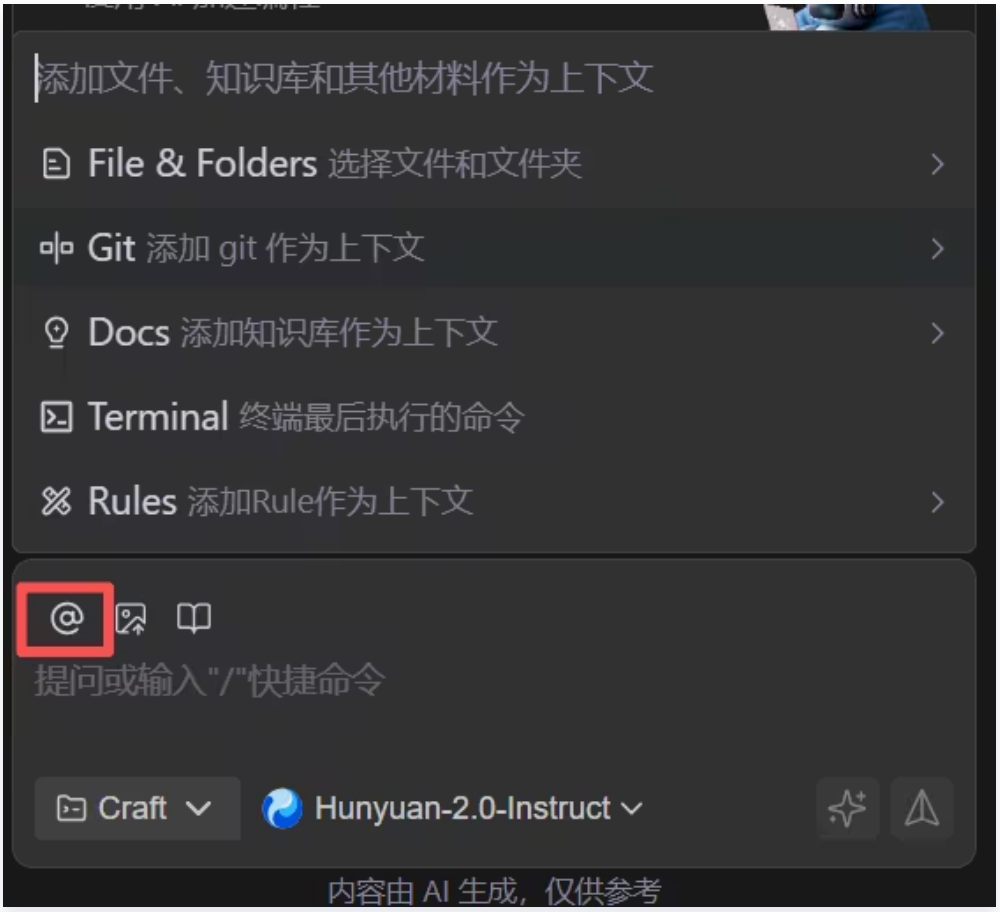


微信开发者工具 IDE

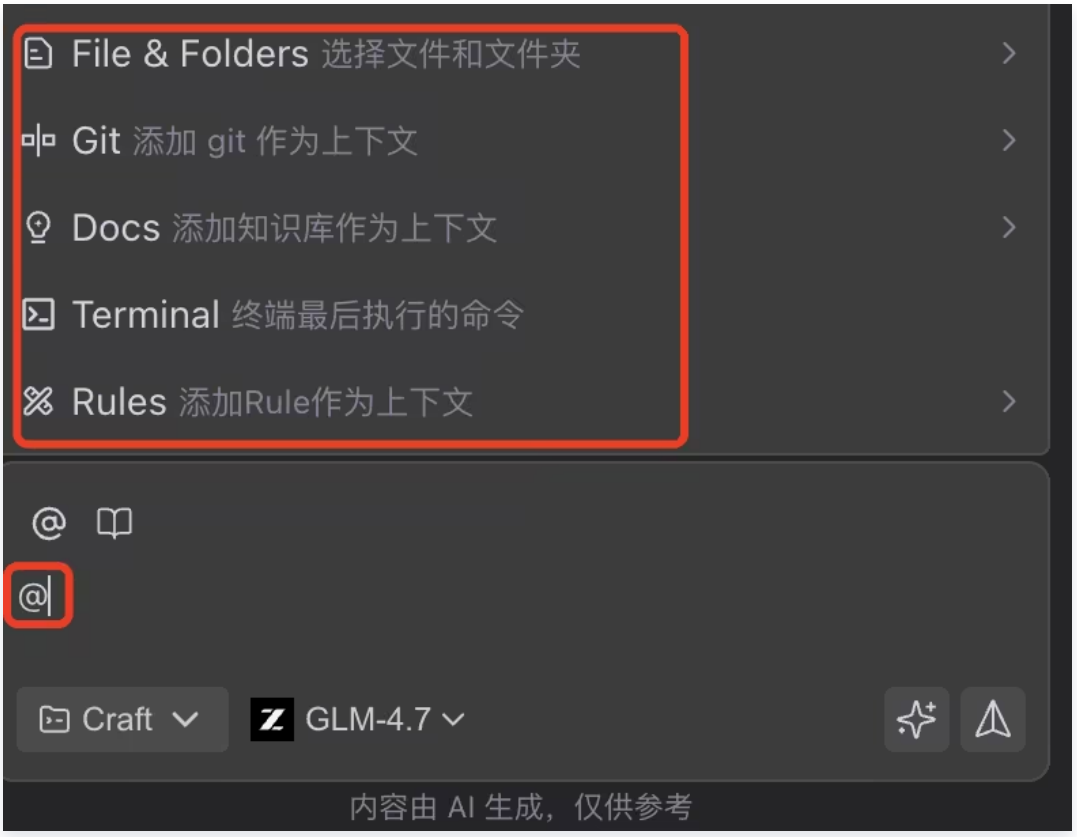


4. 可以选择**添加上下文**作为对话上下文。
   - 单击 @ ，可以选择添加文件或文件夹、文件变更、知识库以及终端最后执行的命令作为对话上下文。

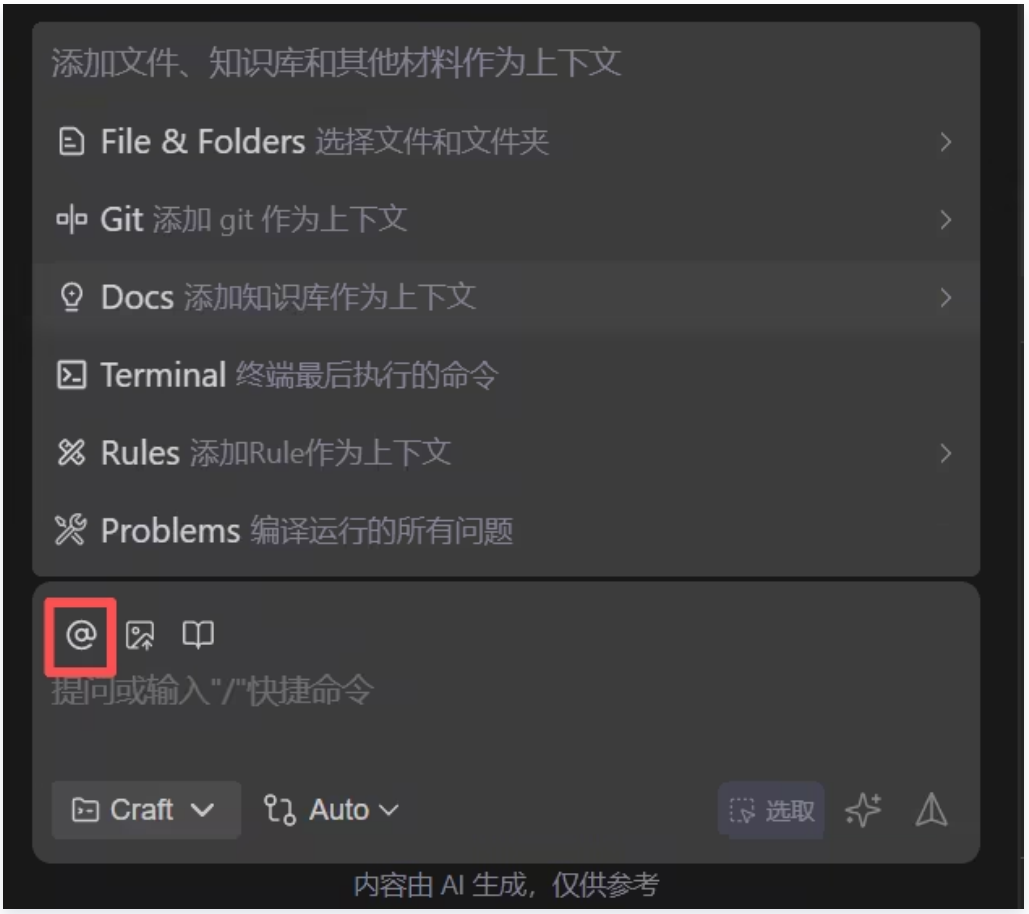
VS Code



JetBrains IDEs


File & Folders 选择文件和文件夹
Git 添加 git 作为上下文
Docs 添加知识库作为上下文
Terminal 终端最后执行的命令
Rules 添加Rule作为上下文
@
@
Craft
GLM-4.7
内容由 AI 生成，仅供参考

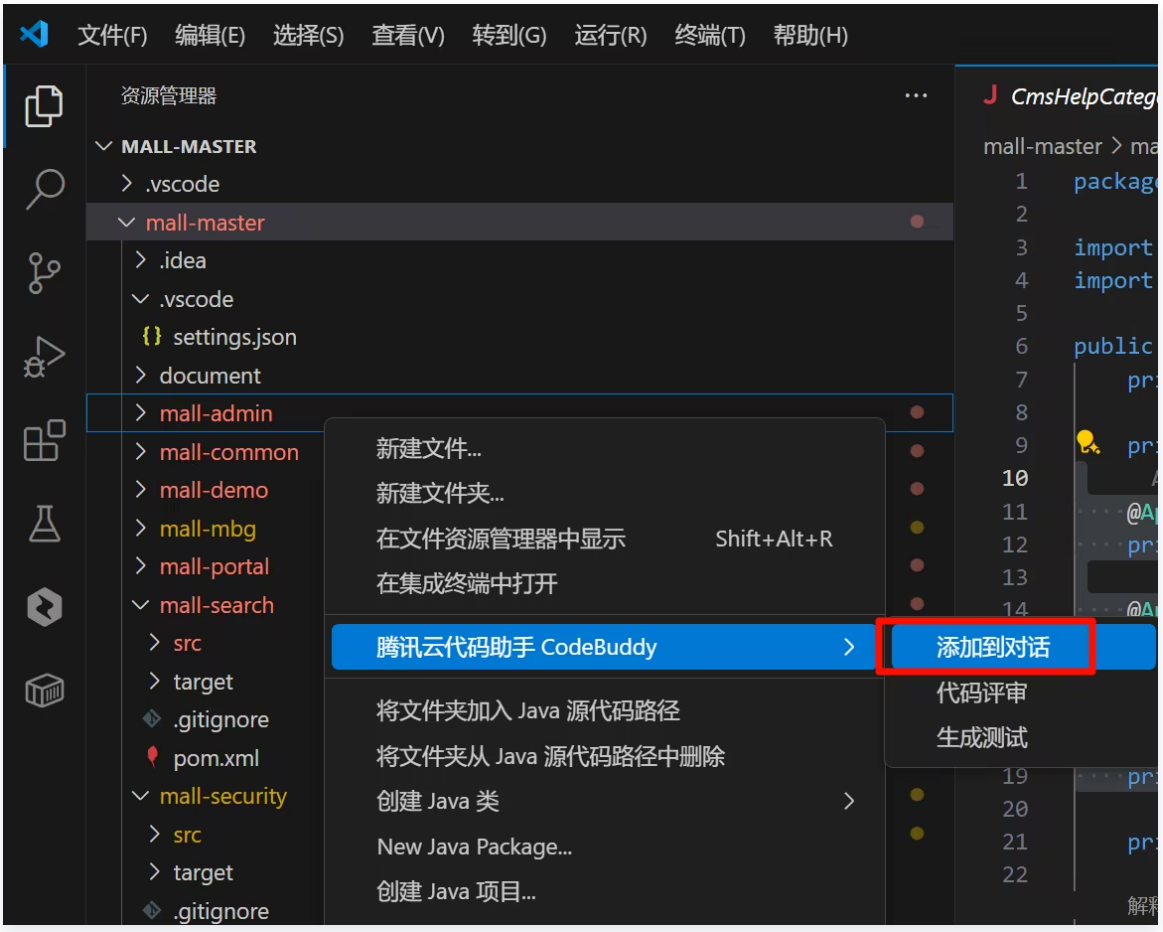
微信开发者工具 IDE



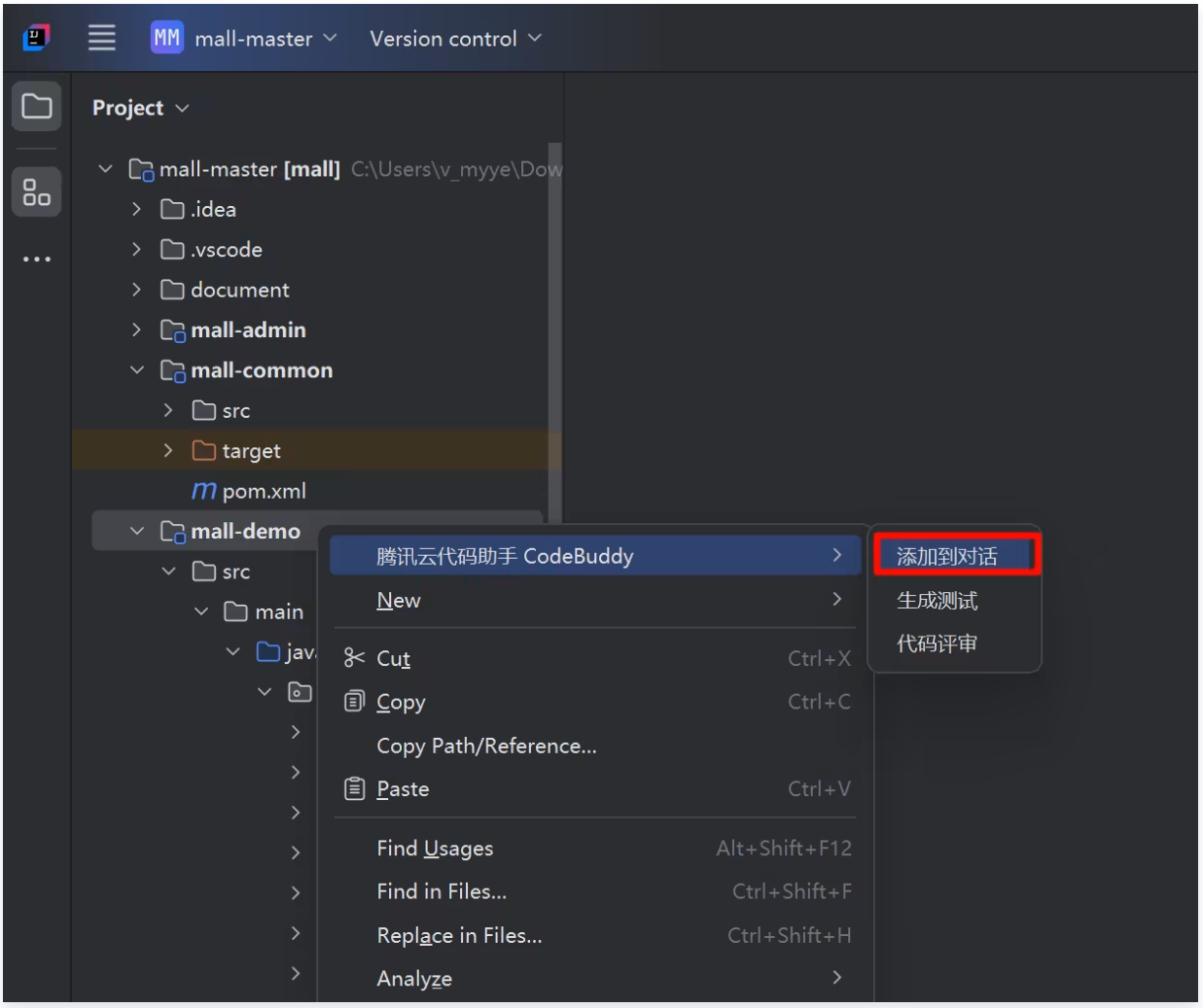
- **@File & Folders**：选择文件或文件夹作为对话上下文信息。
- **@Git**：添加 git 作为上下文。
- **@Docs**：选择添加知识库作为对话上下文信息，包括官方知识库和自定义知识库。
- **@Terminal**：添加终端最后执行的命令作为对话上下文信息。
- **@Rules**：添加 Rule 作为上下文。

或者在**资源管理器**的文件树中，右键单击文件或文件夹选择**腾讯云代码助手 CodeBuddy > 添加到对话**。

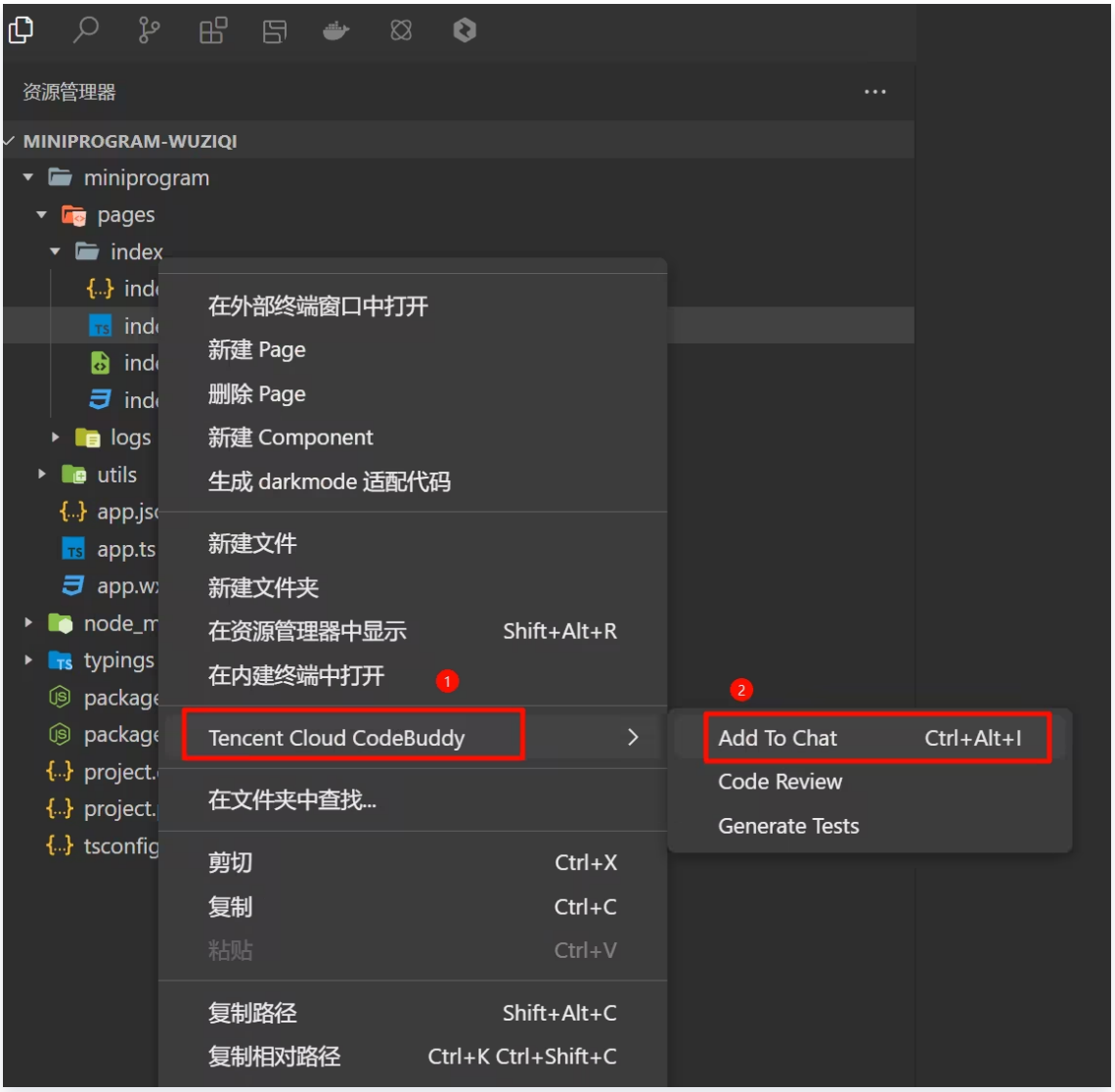
VS Code



JetBrains IDEs


mall-master
Version control
Project
mall-master [mall] C:\Users\v_myye\Dow
.idea
.vscode
document
mall-admin
mall-common
src
target
pom.xml
mall-demo
src
main
腾讯云代码助手 CodeBuddy
New
Cut Ctrl+X
Copy Ctrl+C
Copy Path/Reference...
Paste Ctrl+V
Find Usages Alt+Shift+F12
Find in Files... Ctrl+Shift+F
Replace in Files... Ctrl+Shift+H
Analyze
添加到对话
生成测试
代码评审

微信开发者工具 IDE

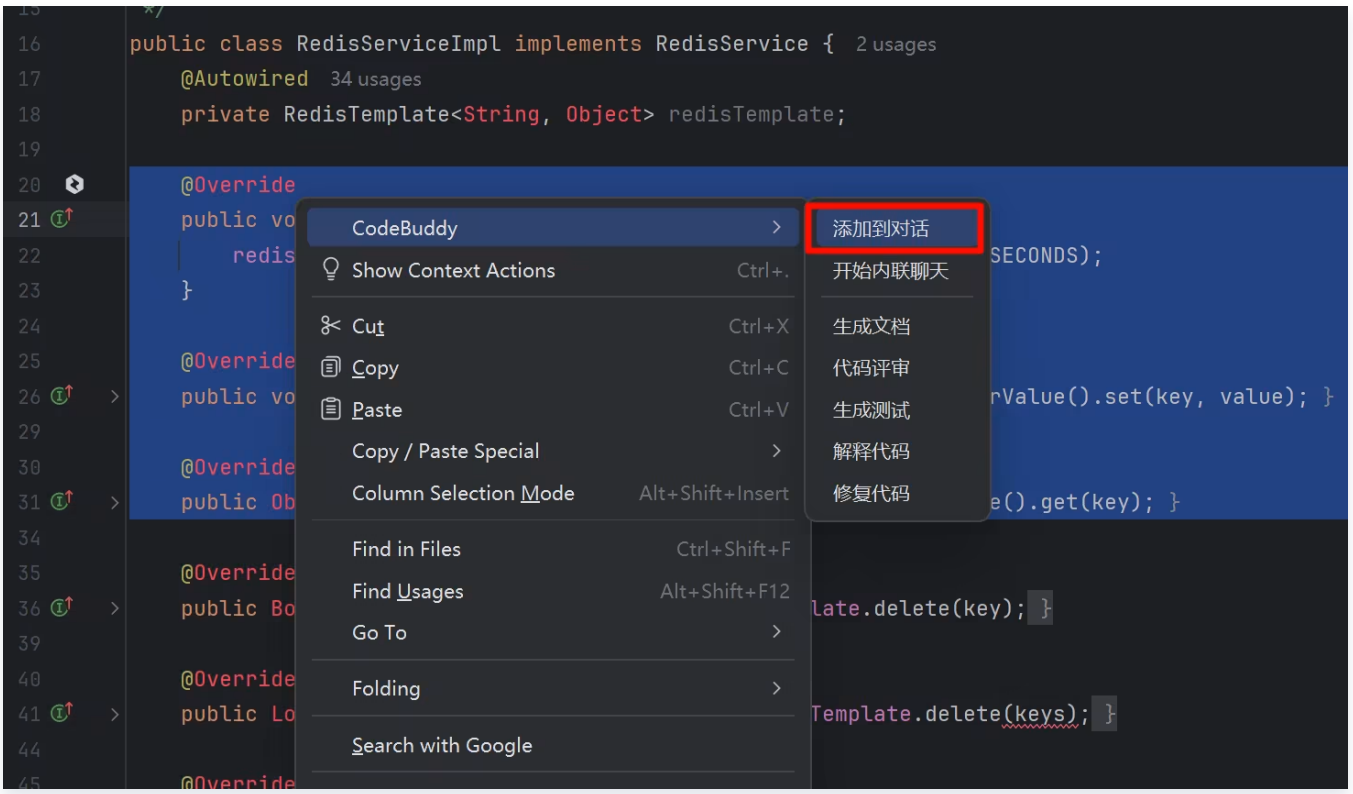


- 或者在编辑区中，圈选代码后右键选择**腾讯云代码助手 CodeBuddy > 添加到对话**，或 **CodeBuddy > 添加到对话**。

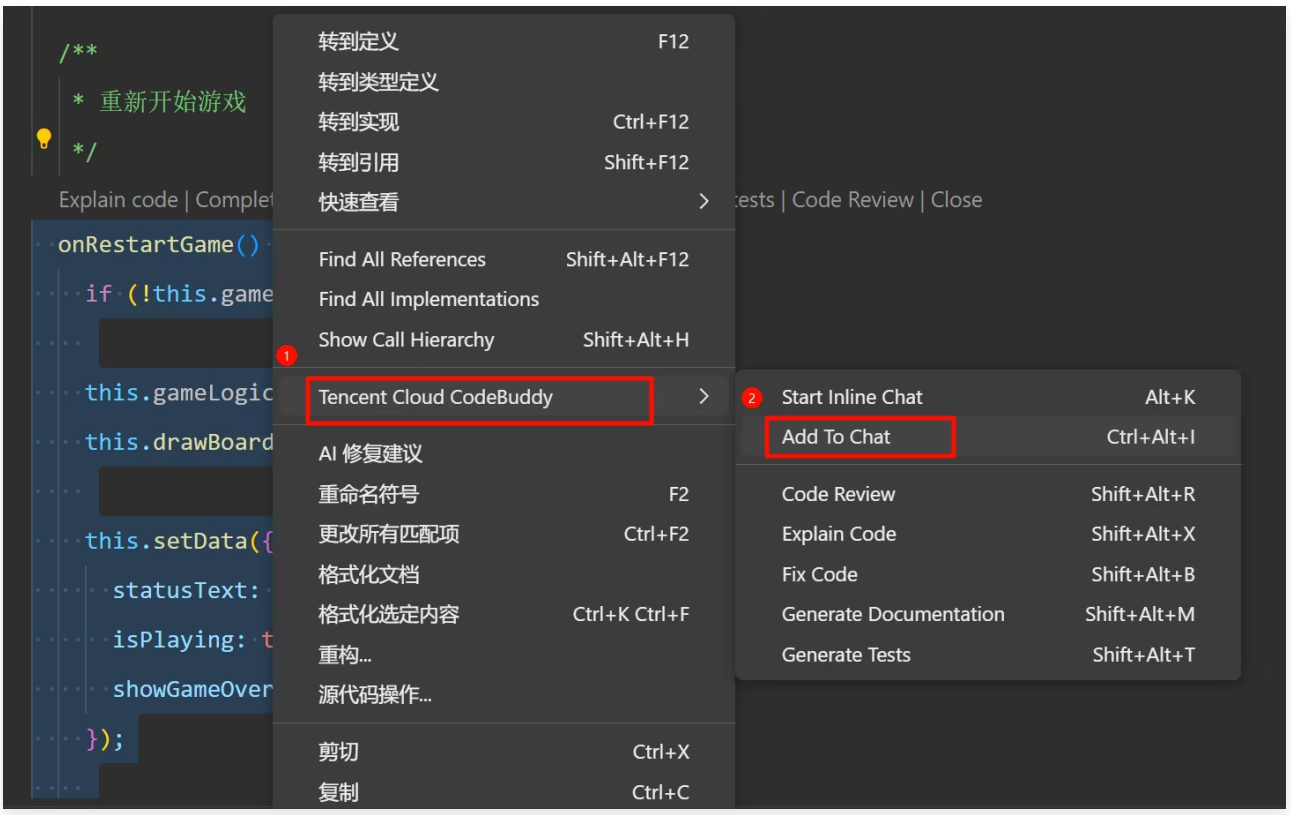
VS Code



JetBrains IDEs

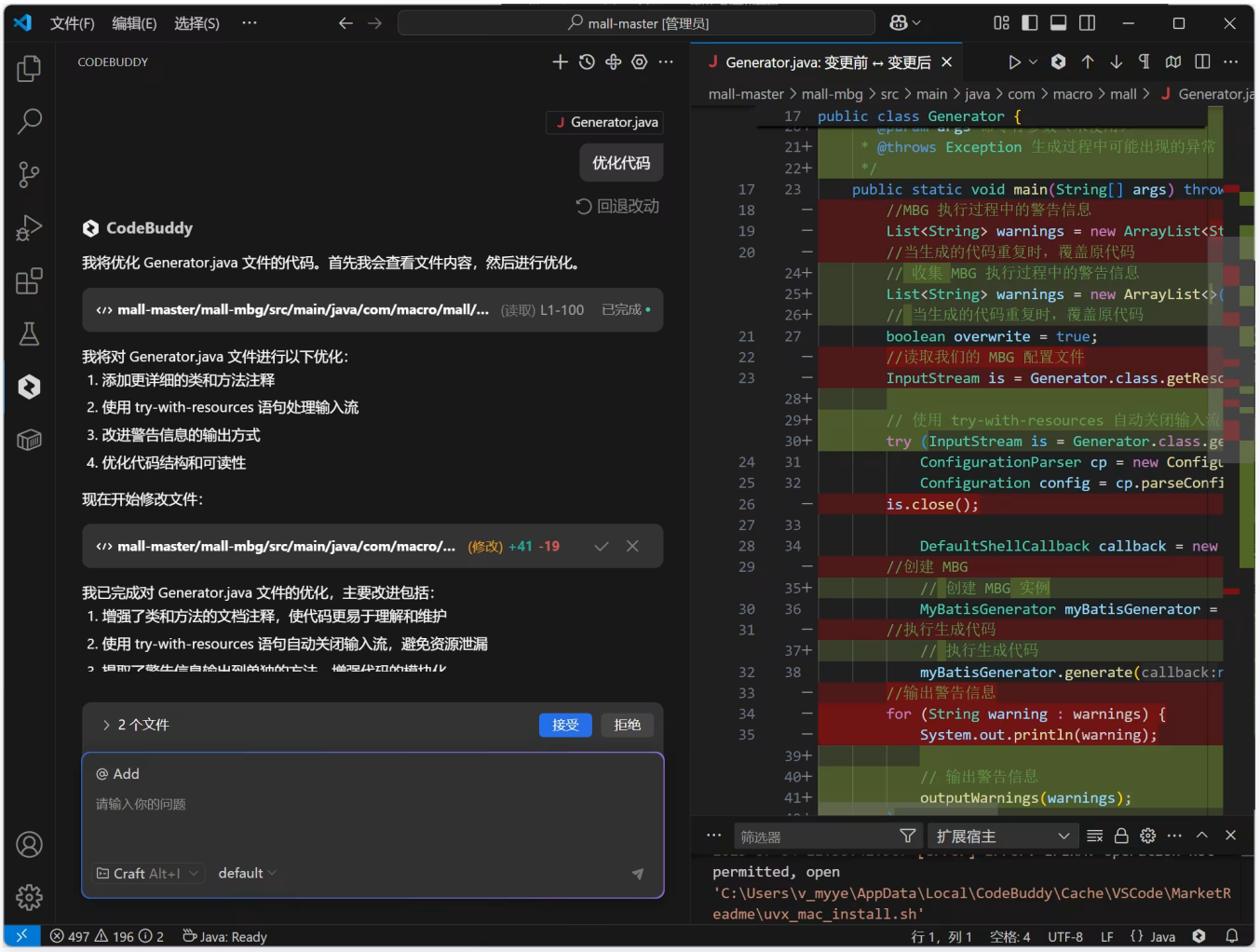

public class RedisServiceImpl implements RedisService { 2 usages
@Autowired 34 usages
private RedisTemplate<String, Object> redisTemplate;
CodeBuddy
Show Context Actions Ctrl+.
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Copy / Paste Special
Column Selection Mode Alt+Shift+Insert
Find in Files Ctrl+Shift+F
Find Usages Alt+Shift+F12
Go To
Folding
Search with Google
添加到对话
开始内联聊天
生成文档
代码评审
生成测试
解释代码
修复代码

微信开发者工具 IDE

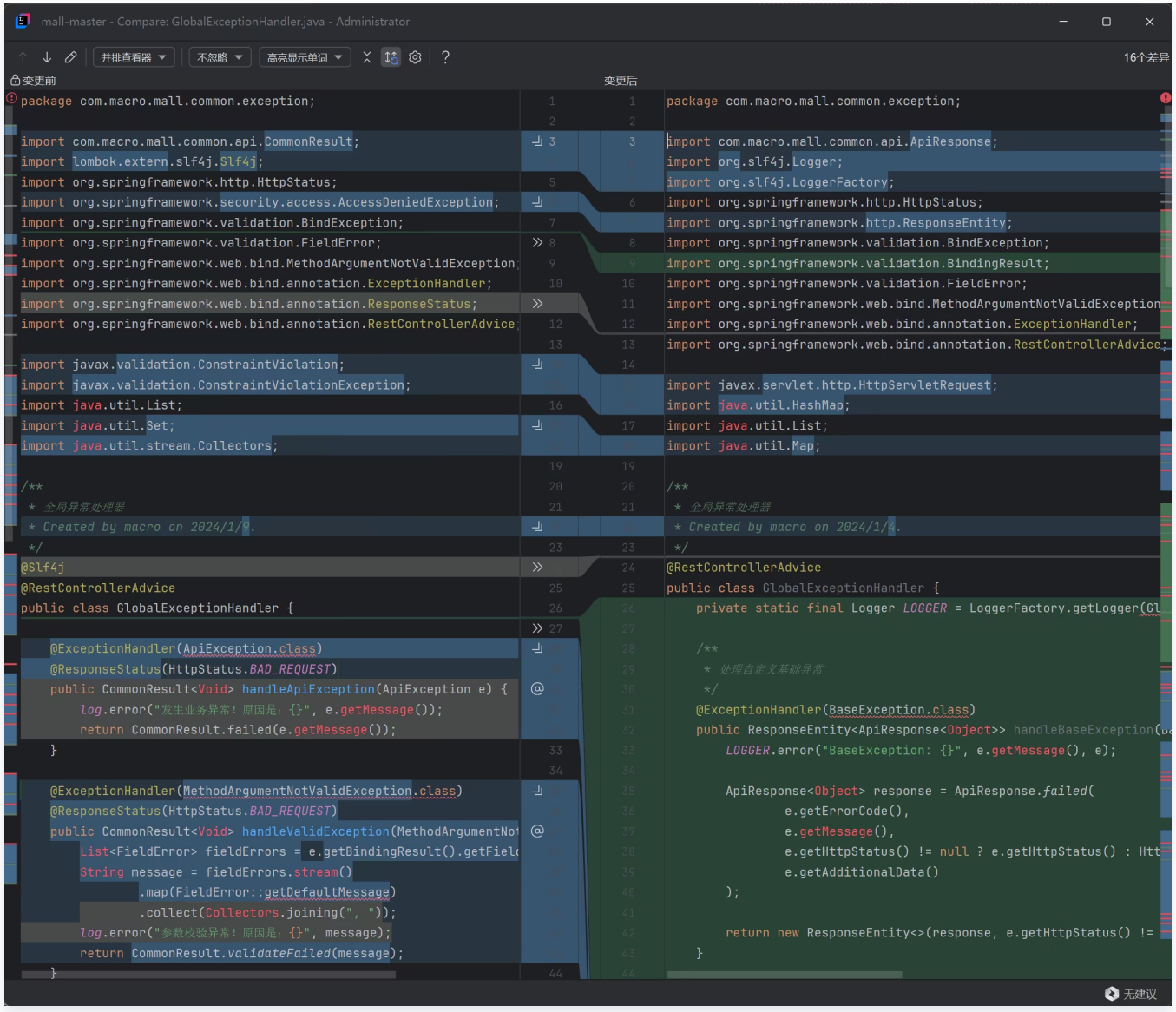


5. 输入需求描述后，Craft 将自主完成多文件代码生成和改写，并以 diff 视图的形式进行分栏展示。此外，JetBrains IDEs 还支持在新窗口以 diff 的形式展示改写文件。

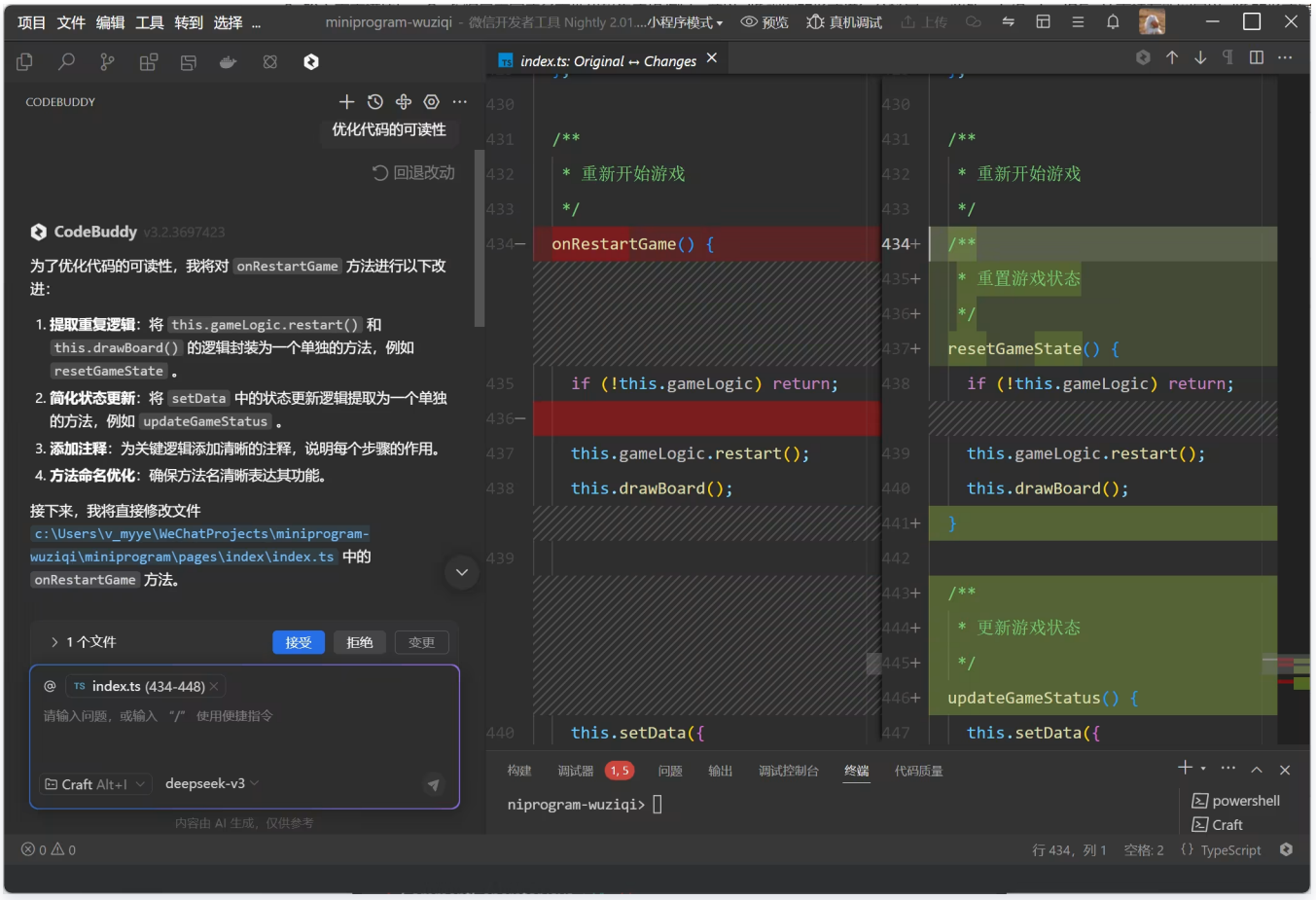
VS Code



JetBrains IDEs

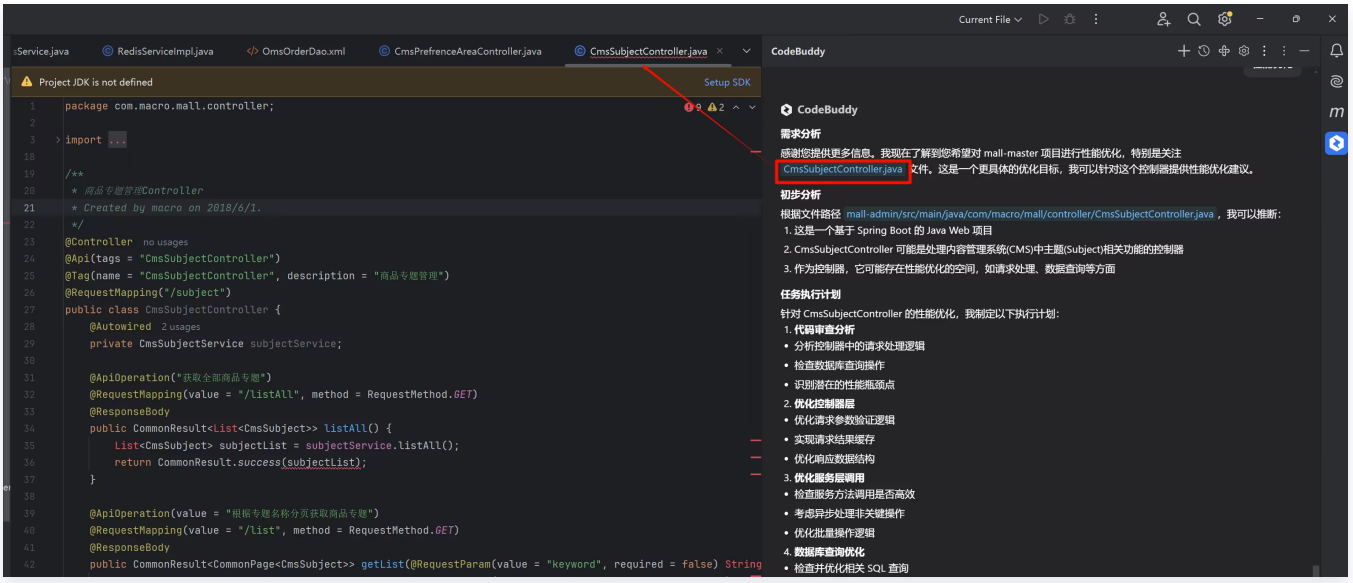

mall-master - Compare: GlobalExceptionHandler.java - Administrator
并排查看器
不忽略
高亮显示单词
16个差异
变更前
变更后
package com.macro.mall.common.exception;
import com.macro.mall.common.api.CommonResult;
import lombok.extern.slf4j.Slf4j;
import org.springframework.http.HttpStatus;
import org.springframework.security.access.AccessDeniedException;
import org.springframework.validation.BindException;
import org.springframework.validation.FieldError;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestControllerAdvice;
import javax.validation.ConstraintViolation;
import javax.validation.ConstraintViolationException;
import java.util.List;
import java.util.Set;
import java.util.stream.Collectors;
/**
* 全局异常处理器
* Created by macro on 2024/1/4.
*/
@Slf4j
@RestControllerAdvice
public class GlobalExceptionHandler {
@ExceptionHandler(ApiException.class)
@ResponseStatus(HttpStatus.BAD_REQUEST)
public CommonResult<Void> handleApiException(ApiException e) {
log.error("发生业务异常！原因是：{}", e.getMessage());
return CommonResult.failed(e.getMessage());
}
@ExceptionHandler(MethodArgumentNotValidException.class)
@ResponseStatus(HttpStatus.BAD_REQUEST)
public CommonResult<Void> handleValidException(MethodArgumentNot
List<FieldError> fieldErrors = e.getBindingResult().getFiel
String message = fieldErrors.stream()
.map(FieldError::getDefaultMessage)
.collect(Collectors.joining(", "));
log.error("参数校验异常！原因是：{}", message);
return CommonResult.validateFailed(message);
}
package com.macro.mall.common.exception;
import com.macro.mall.common.api.ApiResponse;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.BindException;
import org.springframework.validation.BindingResult;
import org.springframework.validation.FieldError;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestControllerAdvice;
import javax.servlet.http.HttpServletRequest;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
/**
* 全局异常处理器
* Created by macro on 2024/1/4.
*/
@RestControllerAdvice
public class GlobalExceptionHandler {
private static final Logger LOGGER = LoggerFactory.getLogger(Gl
/**
* 处理自定义基础异常
*/
@ExceptionHandler(BaseException.class)
public ResponseEntity<ApiResponse<Object>> handleBaseException(
LOGGER.error("BaseException: {}", e.getMessage(), e);
ApiResponse<Object> response = ApiResponse.failed(
e.getErrorCode(),
e.getMessage(),
e.getHttpStatus() != null ? e.getHttpStatus() : Htt
e.getAdditionalData()
);
return new ResponseEntity<>(response, e.getHttpStatus() !=
}
无建议

微信开发者工具 IDE

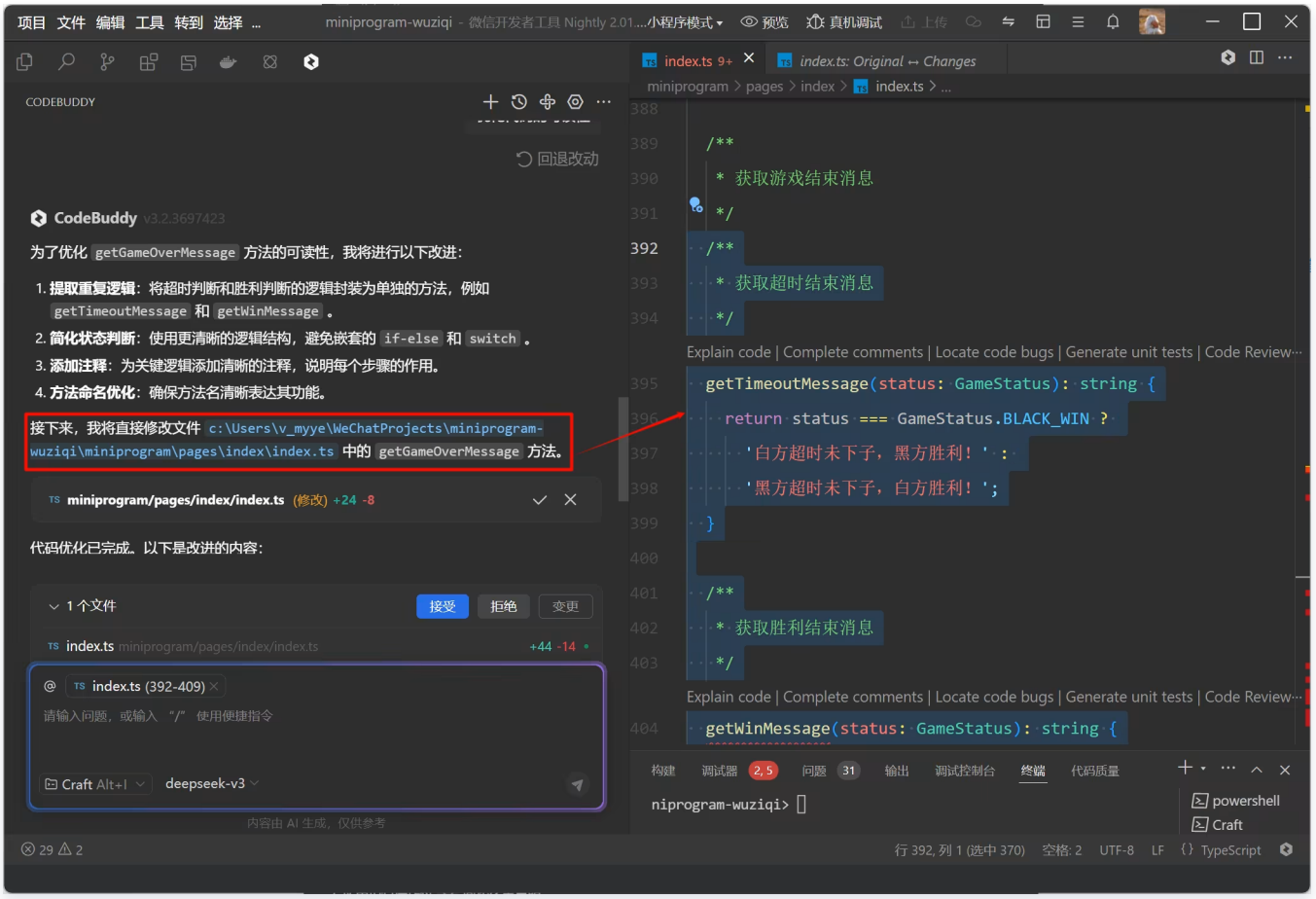


6. 对于对话生成的代码文件内容，单击蓝色显示的代码文件或方法或类或路径等，可直接跳转到对应内容进行查看。

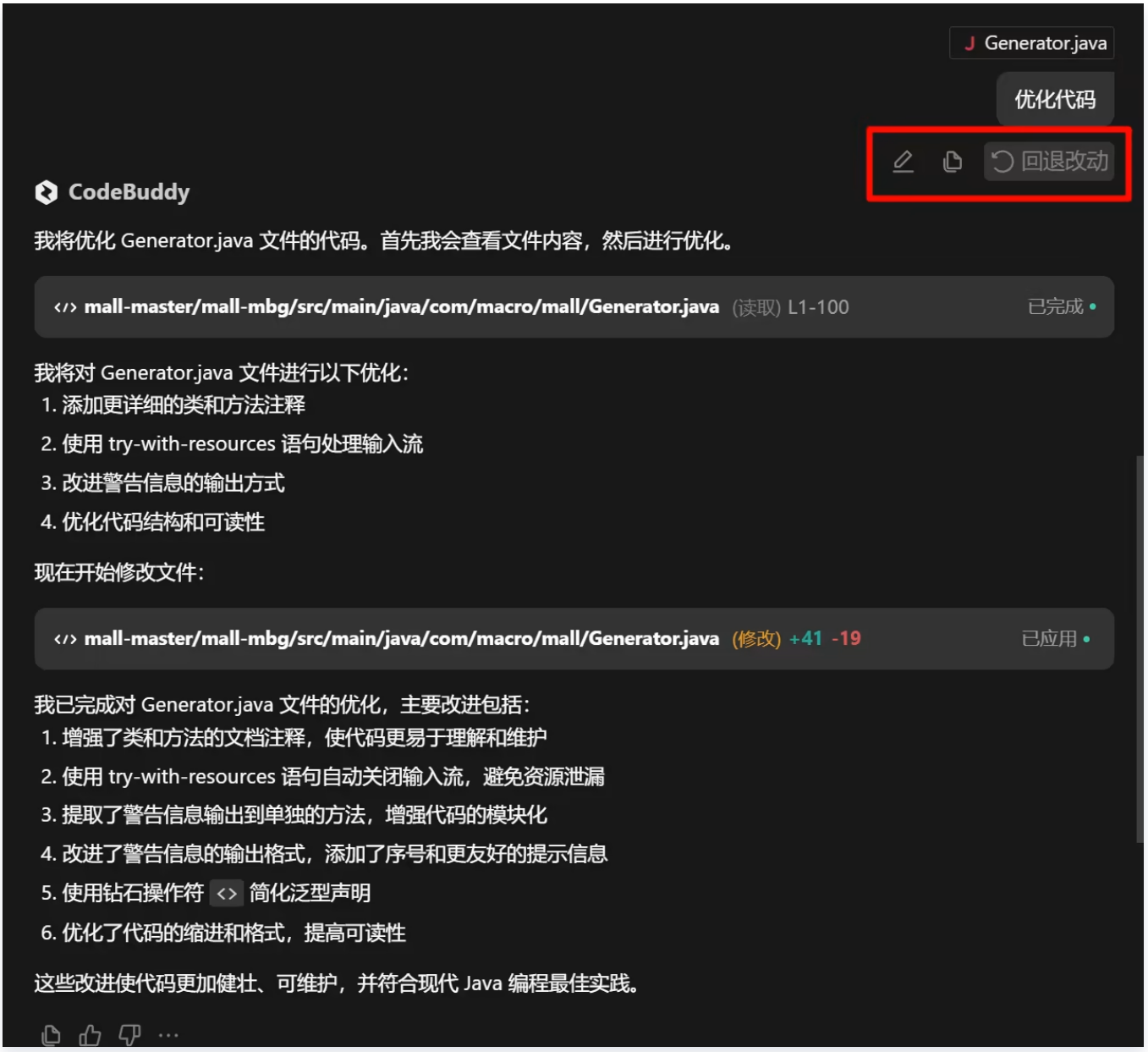
VS Code


Current File
CmsSubjectController.java
Project JDK is not defined
Setup SDK
CodeBuddy
需求分析
感谢您提供更多信息。我现在了解到您希望对 mall-master 项目进行性能优化，特别是关注 CmsSubjectController.java 文件。这是一个更具体的优化目标，我可以针对这个控制器提供性能优化建议。
初步分析
根据文件路径 mall-admin/src/main/java/com/macro/mall/controller/CmsSubjectController.java，我可以推断：
1. 这是一个基于 Spring Boot 的 Java Web 项目
2. CmsSubjectController 可能是处理内容管理系统(CMS)中主题(Subject)相关功能的控制器
3. 作为控制器，它可能存在性能优化的空间，如请求处理、数据查询等方面
任务执行计划
针对 CmsSubjectController 的性能优化，我制定以下执行计划：
1. 代码审查分析
• 分析控制器中的请求处理逻辑
• 检查数据库查询操作
• 识别潜在的性能瓶颈点
2. 优化控制器层
• 优化请求参数验证逻辑
• 实现请求结果缓存
• 优化响应数据结构
3. 优化服务层调用
• 检查服务方法调用是否高效
• 考虑异步处理非关键操作
• 优化批量操作逻辑
4. 数据库查询优化
• 检查并优化相关 SQL 查询

微信开发者工具 IDE

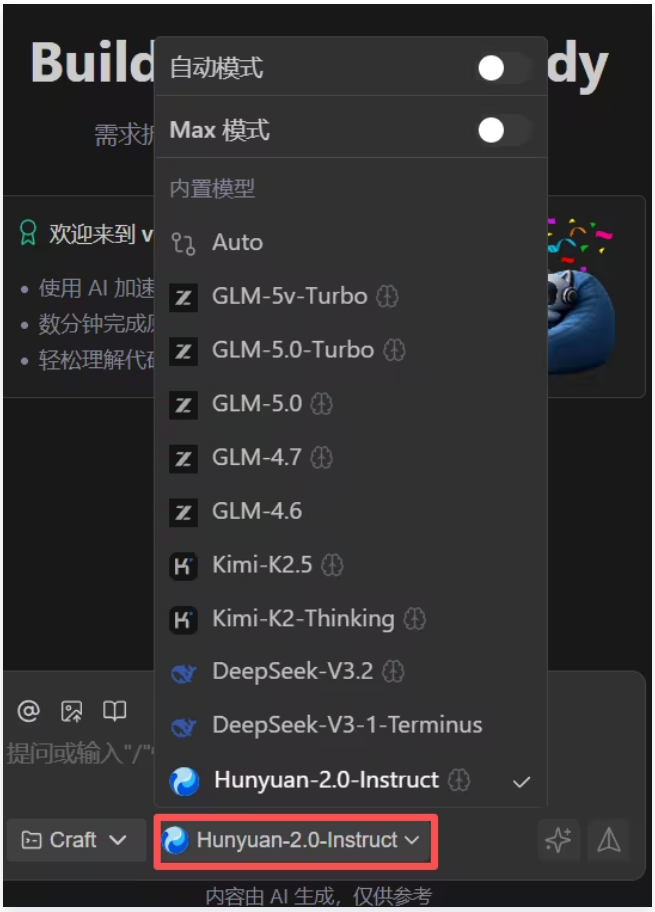

项目 文件 编辑 工具 转到 选择 ...
miniprogram-wuziqi - 微信开发者工具 Nightly 2.01...
小程序模式
预览
真机调试
上传
CODEBUDDY
回退改动
CodeBuddy v3.2.3697423
为了优化 getGameOverMessage 方法的可读性，我将进行以下改进：
1. 提取重复逻辑：将超时判断和胜利判断的逻辑封装为单独的方法，例如 getTimeoutMessage 和 getWinMessage 。
2. 简化状态判断：使用更清晰的逻辑结构，避免嵌套的 if-else 和 switch 。
3. 添加注释：为关键逻辑添加清晰的注释，说明每个步骤的作用。
4. 方法命名优化：确保方法名清晰表达其功能。
接下来，我将直接修改文件 c:\Users\v_myye\WeChatProjects\miniprogram-wuziqi\miniprogram\pages\index\index.ts 中的 getGameOverMessage 方法。
miniprogram/pages/index/index.ts (修改) +24 -8
代码优化已完成。以下是改进的内容：
1 个文件
接受
拒绝
变更
index.ts miniprogram/pages/index/index.ts +44 -14
index.ts (392-409)
请输入问题，或输入 "/" 使用便捷指令
Craft Alt+I
deepseek-v3
内容由 AI 生成，仅供参考
index.ts 9+
index.ts: Original ↔ Changes
miniprogram > pages > index > index.ts > ...
/**
* 获取游戏结束消息
*/
/**
* 获取超时结束消息
*/
Explain code | Complete comments | Locate code bugs | Generate unit tests | Code Review
getTimeoutMessage(status: GameStatus): string {
return status === GameStatus.BLACK_WIN ?
'白方超时未下子，黑方胜利！' :
'黑方超时未下子，白方胜利！';
}
/**
* 获取胜利结束消息
*/
Explain code | Complete comments | Locate code bugs | Generate unit tests | Code Review
getWinMessage(status: GameStatus): string {
构建 调试器 2, 5 问题 31 输出 调试控制台 终端 代码质量
niprogram-wuziqi>
powershell
Craft
29 2
行 392, 列 1 (选中 370) 空格: 2 UTF-8 LF TypeScript

7. 版本化管理。Craft 支持版本管理能力，将鼠标悬浮在之前对话版本，您可以选择回退版本、重新编辑。如果想要回退版本，详情请参考 Craft 撤回检查点 。



8. 模型切换。内置了 **hunyuan、deepseek-v3、Kimi、GLM** 最新模型，默认（Auto）为自动选择模型，支持您对模型进行切换，满足不同场景需求。

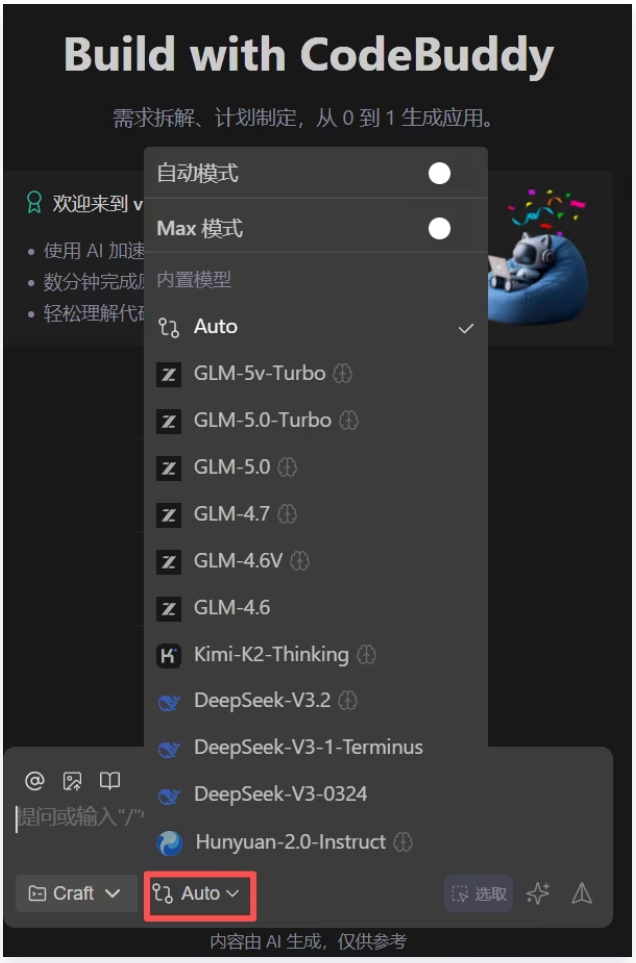
VS Code



JetBrains IDEs


自动模式
Max 模式
内置模型
Auto
GLM-5v-Turbo
GLM-5.0-Turbo
GLM-5.0
GLM-4.7
GLM-4.6
Kimi-K2.5
Kimi-K2-Thinking
DeepSeek-V3.2
DeepSeek-V3-1-Terminus
Hunyuan-2.0-Instruct
Craft
Auto
内容由 AI 生成，仅供参考

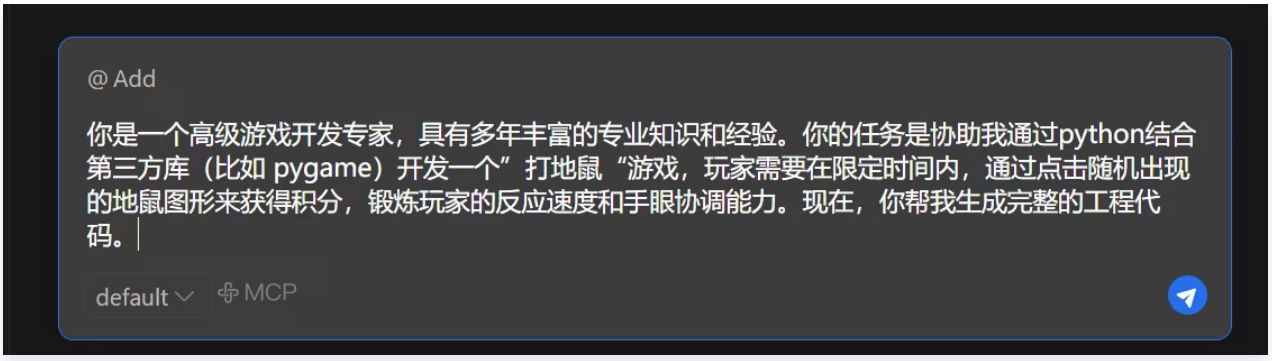
微信开发者工具 IDE



实战场景示例

下面为您演示一个《打地鼠》游戏的开发，来体验一下 Craft 通过自然语言来生成代码工程的能力。

1. 在 Craft 对话框中输入提示词需求。



2. Craft 的执行过程。

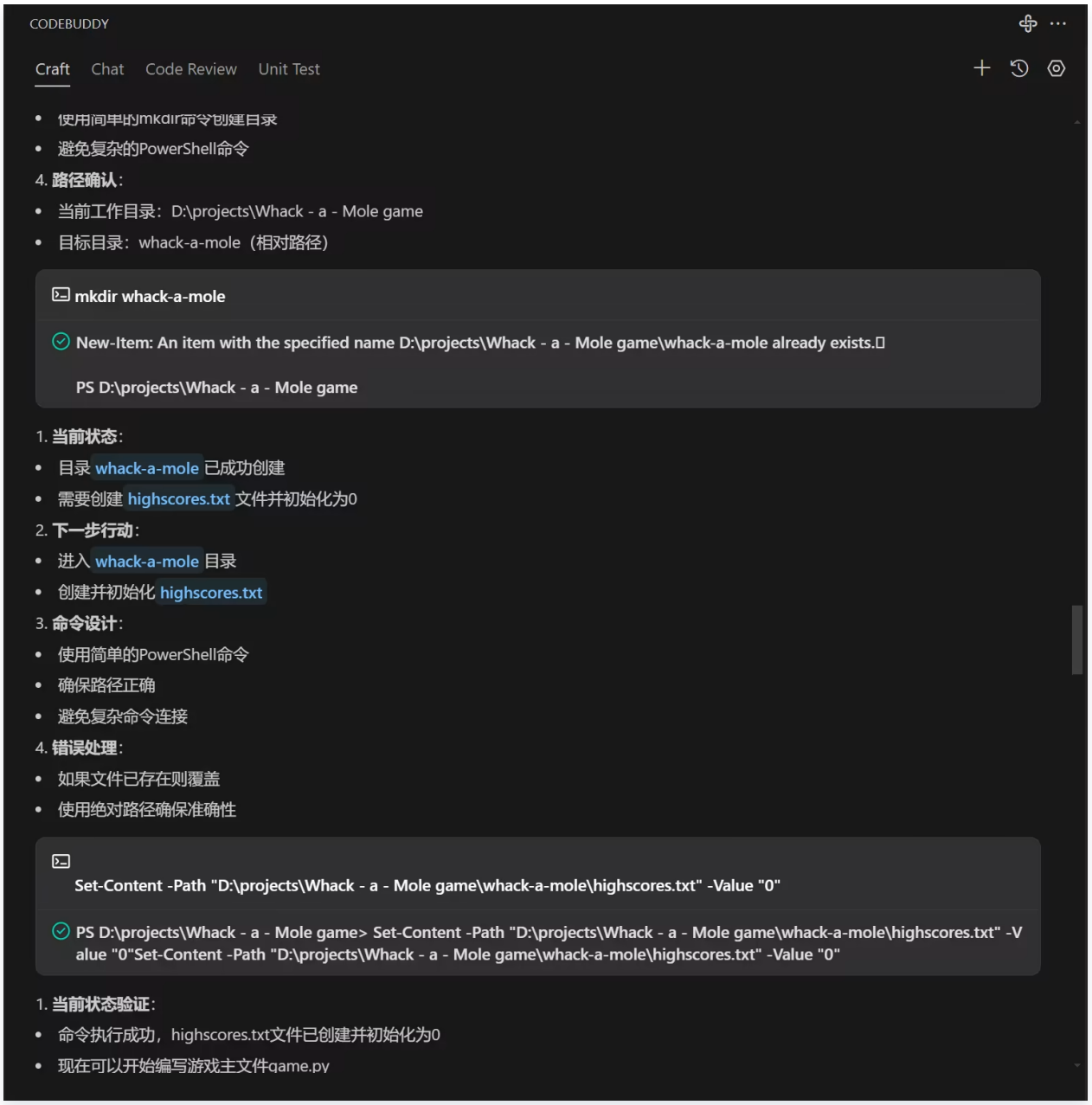
   2.1 在开始执行游戏开发任务前，Craft 会向您询问具体要求信息，以此来制定详细的开发执行计划。



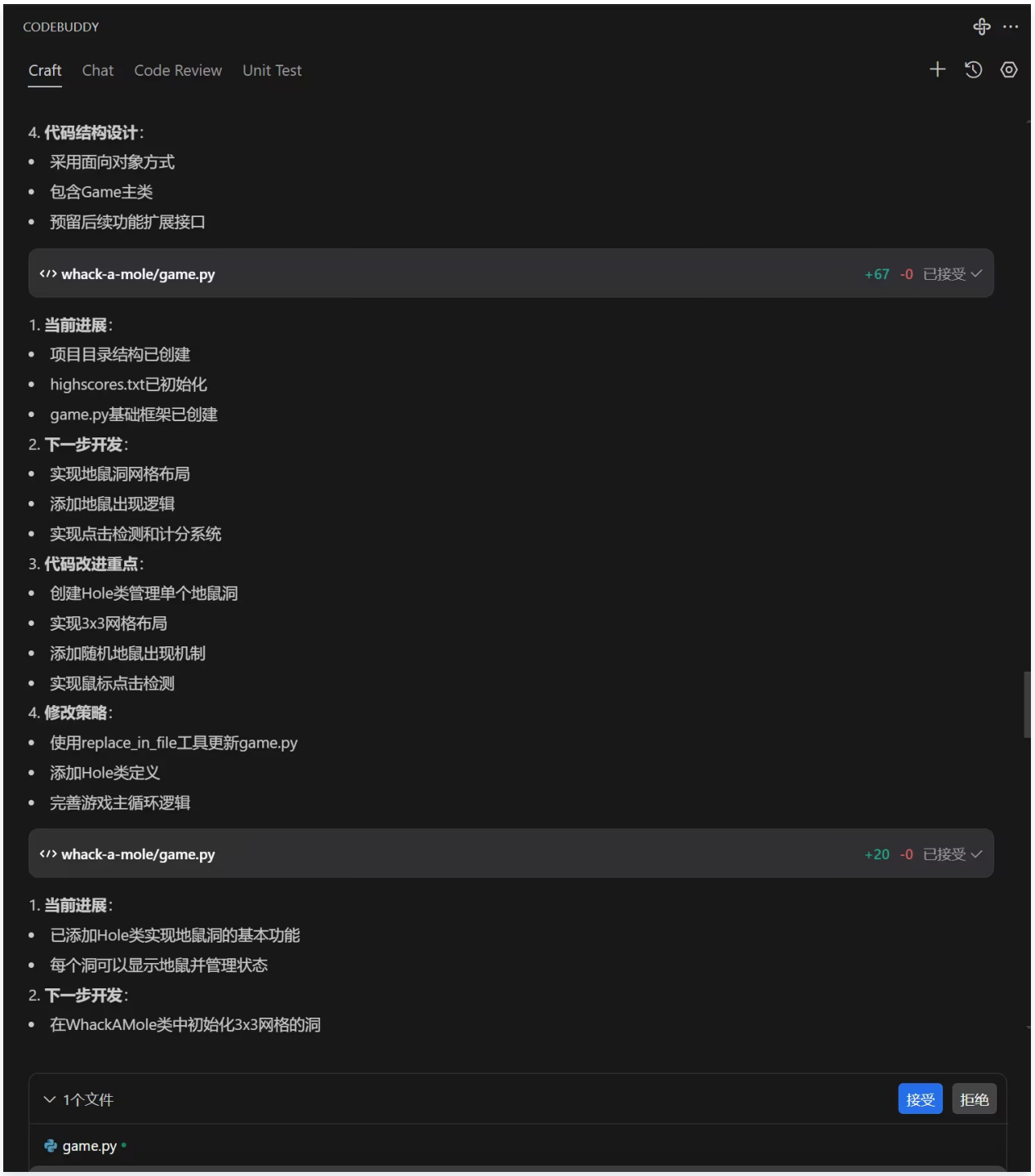
2.2 开发计划制定完成后，请求您进行确认。如果不满意可以选择继续输入需求；如果满意，单击开始即可开始自动执行开发计划。



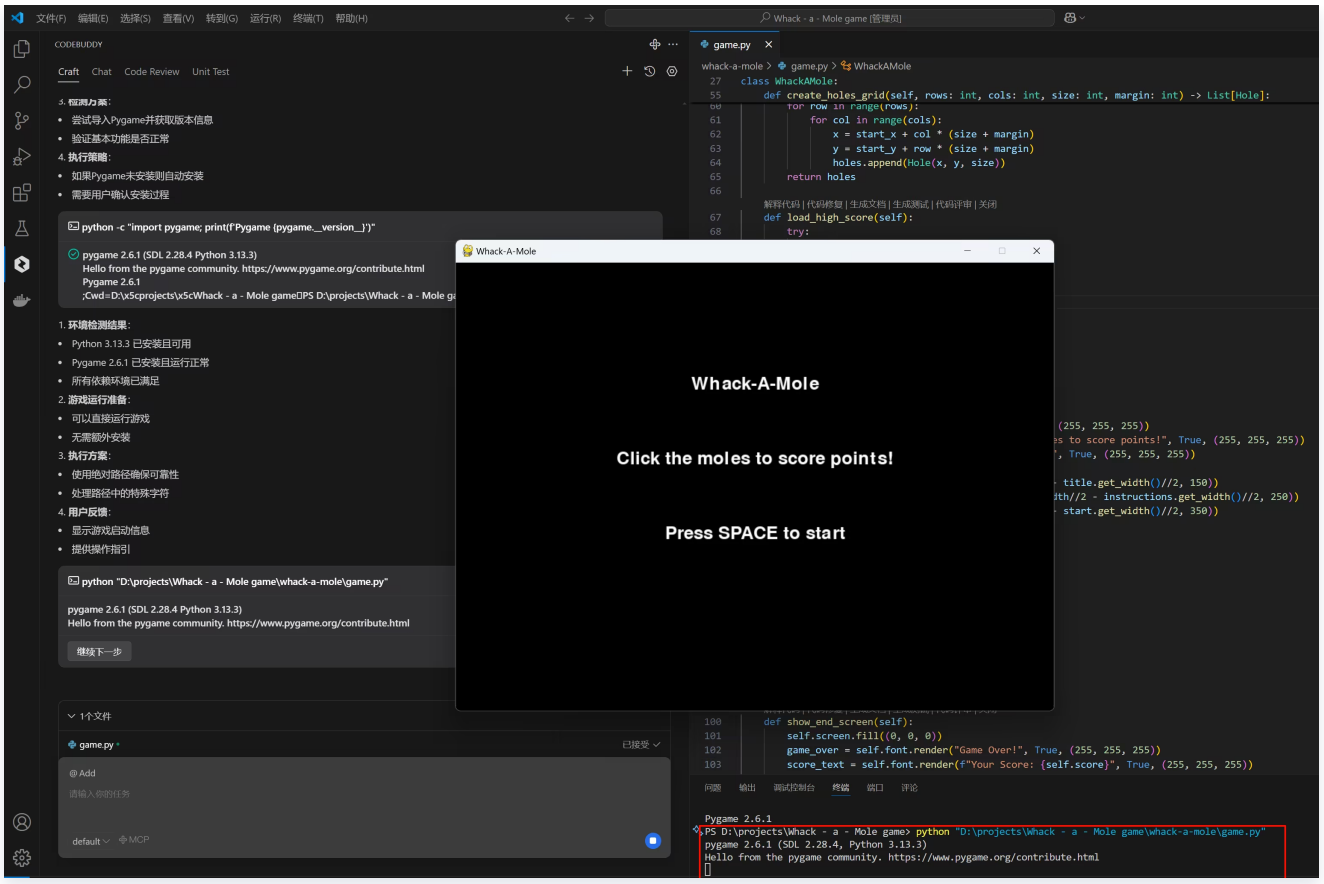
2.3 先对需求进行分析，并创建了基本目录结构。



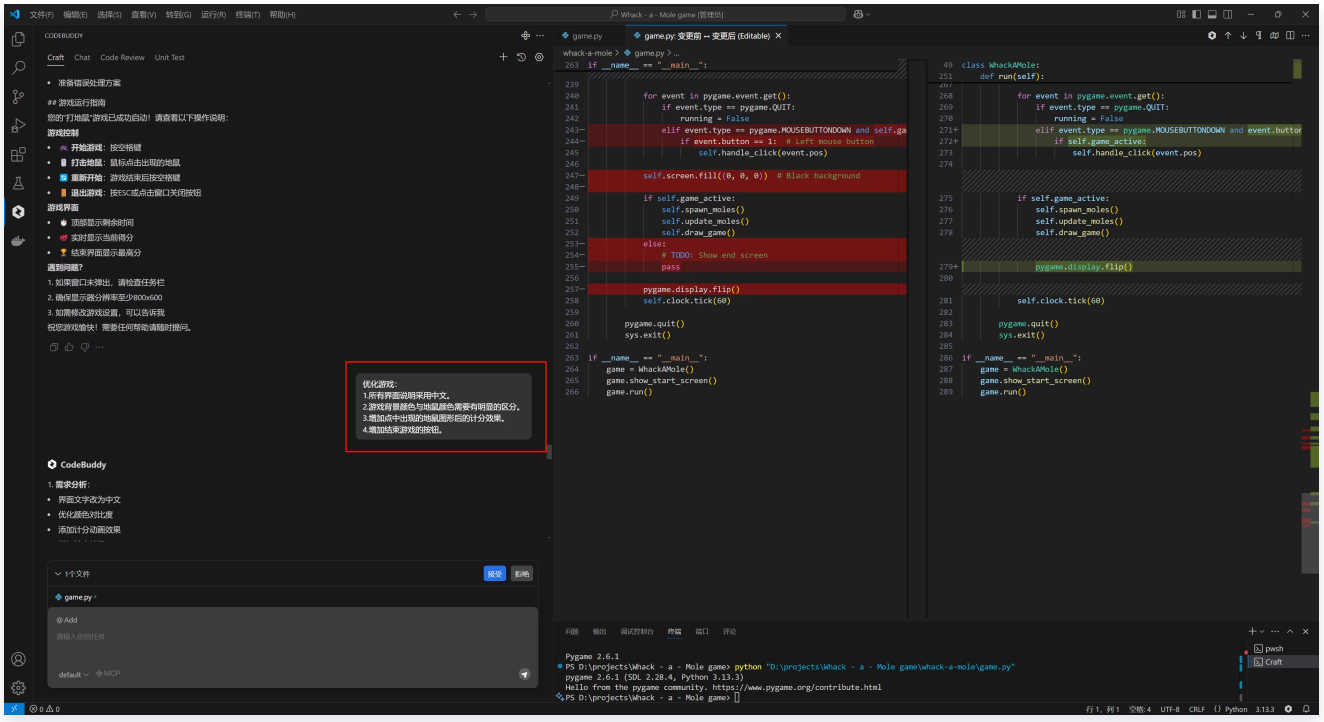
2.4 基于任务依次创建实现逻辑。



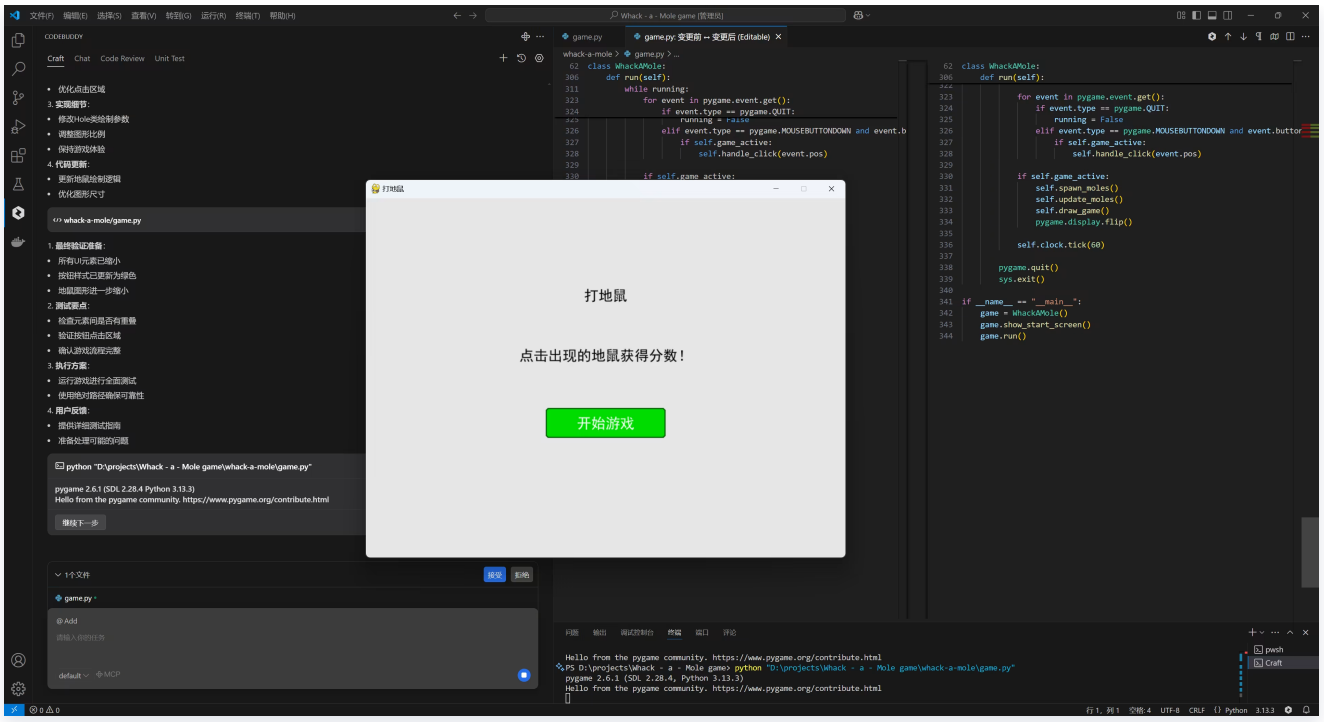
2.5 所有逻辑代码全部实现后，会对当前游戏运行环境进行检测。如果检测到 Python 或第三方库未安装时，会自动进行安装。环境安装完成后，会自动运行程序。



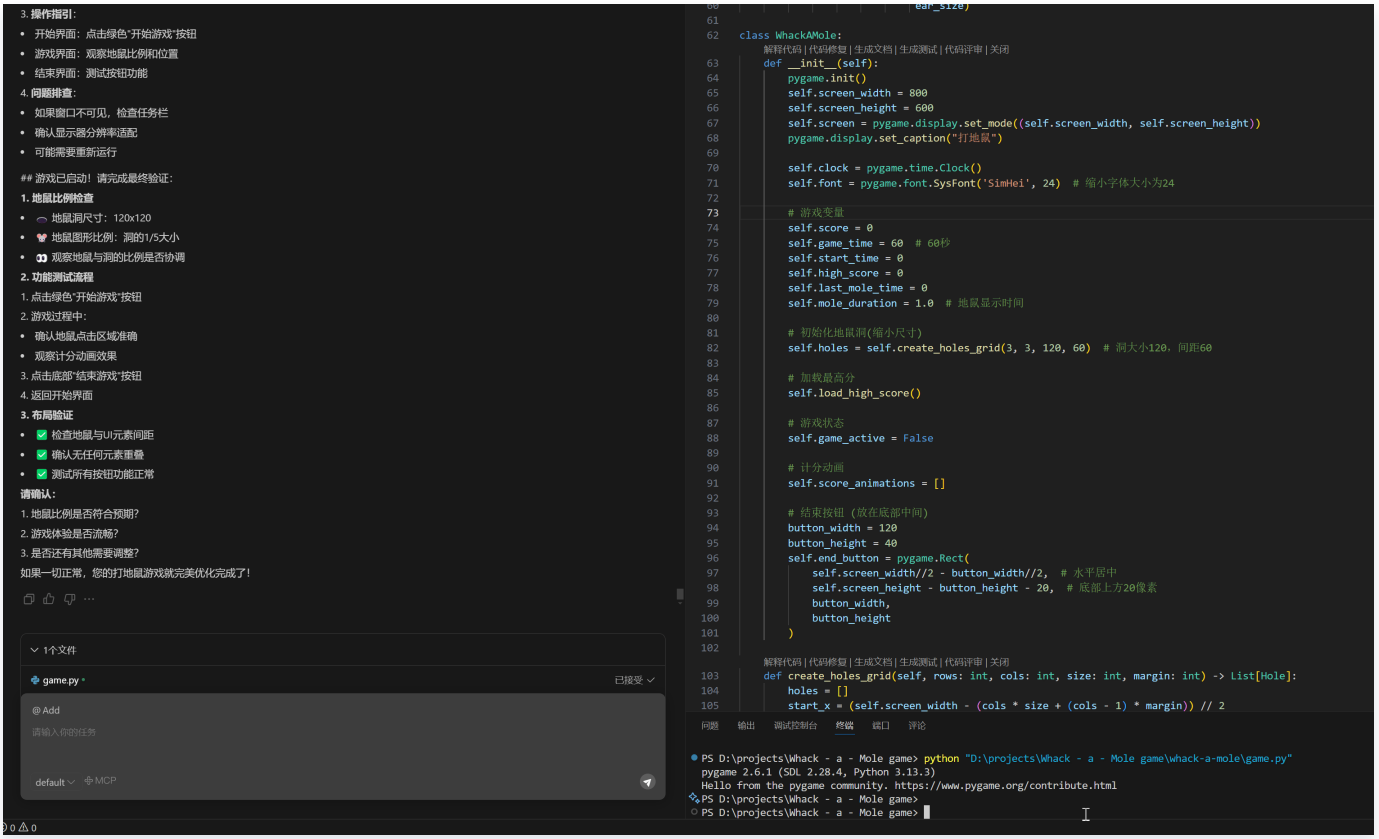
2.6 程序测试完成能够正常运行后，可以选择对程序进行优化。



2.7 游戏优化后，自动对所优化的需求进行游戏测试。



3. 效果演示。



Craft 撤回检查点

最近更新时间：2026-07-20 19:49:33

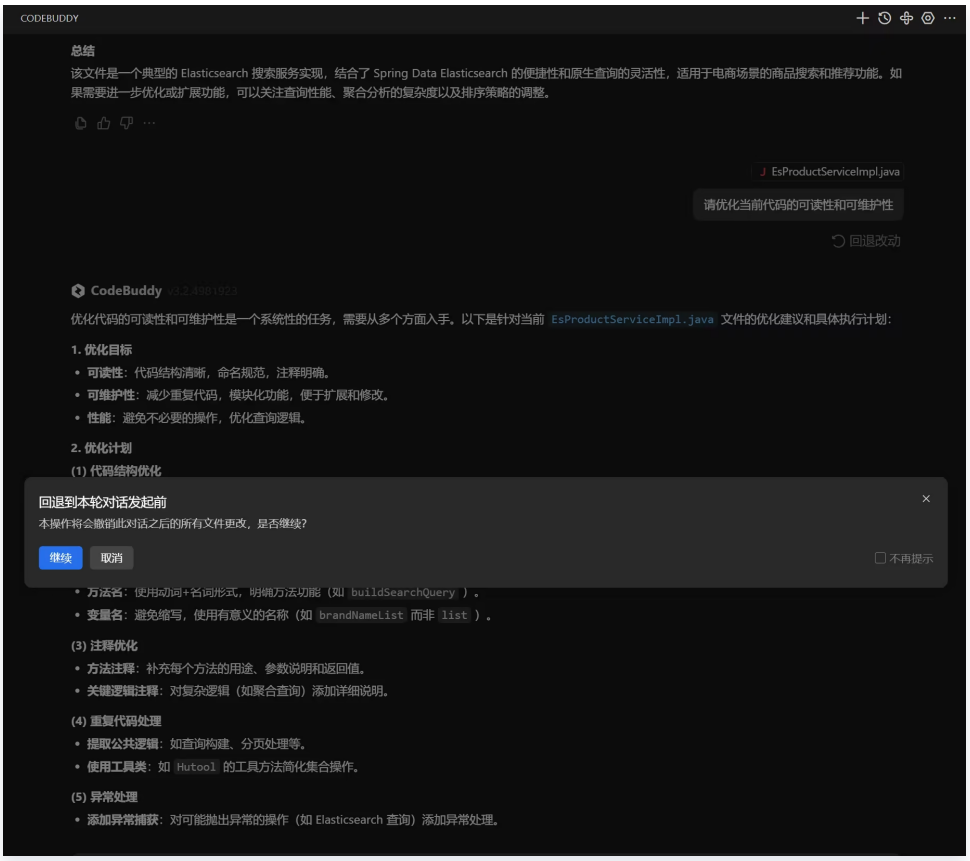
当使用 Craft 编码智能体时，每次对话发生文件变更后，Craft 会对这轮对话的文件变更自动创建一轮检查点，并自动做一轮版本化管理。现在，撤回检查点的入口已经固定展示，您可以根据需要进行操作。下面以一个操作示例，为您进行介绍。

撤回文件变更

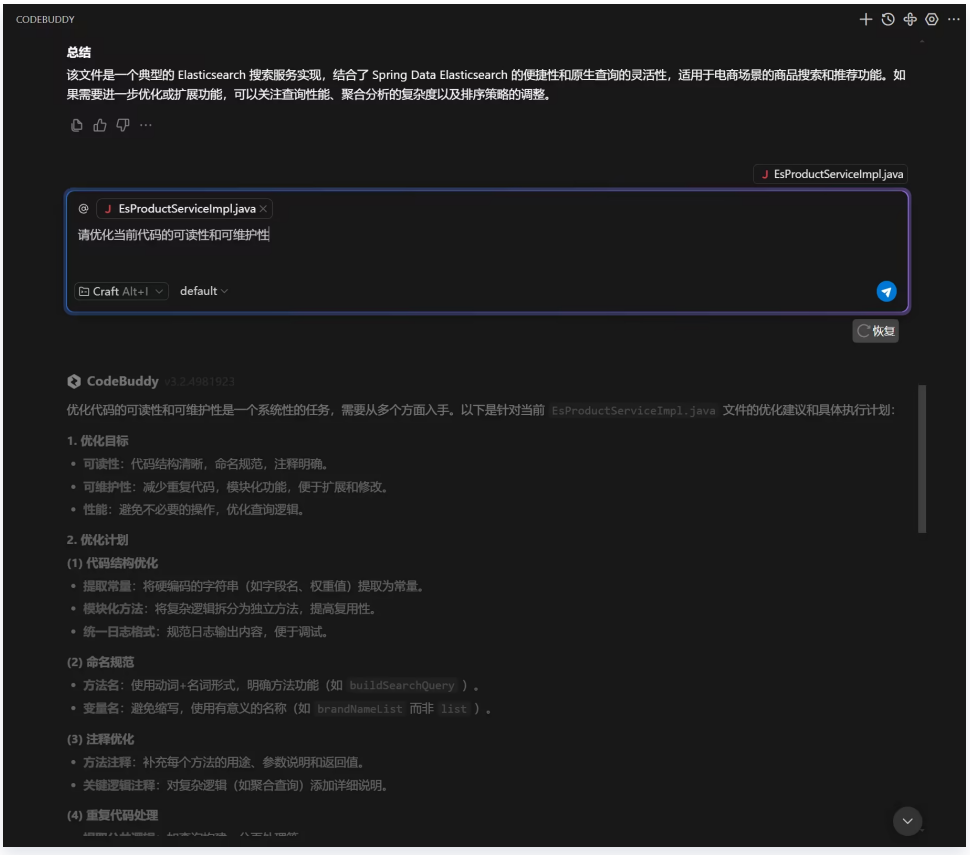
1. 以 EsProductServiceImpl.java 发生的文件变更为例，当模型修改完代码并停止后，会在这轮对话上自动创建一轮检查点，固定显示**回退改动**入口按钮。



2. 单击**回退改动**，Craft 会向您请求是否继续回退版本。单击**继续**就会撤回到“**请优化当前代码的可读性和可维护性**”这个检查点之前的状态，这个检查点对应所进行的任务都会被撤销。若单击**取消**则会取消当前的操作，也可以选择勾选右侧的**不再提示**，此后回退改动的请求确认弹窗都不会再弹出，因此需要**谨慎**操作。



3. 选择**继续**后，回退版本后的效果如下图所示。此时您可以基于当前撤回后的版本，输入需求描述继续完成任务。

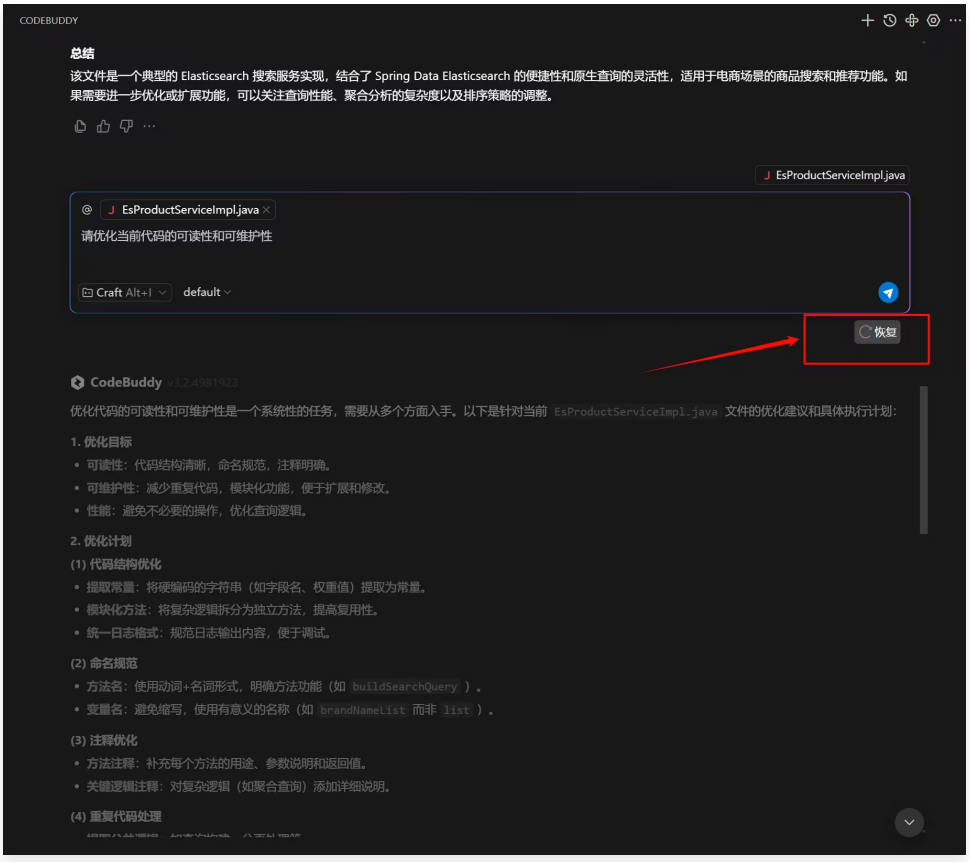


恢复到撤回前的内容

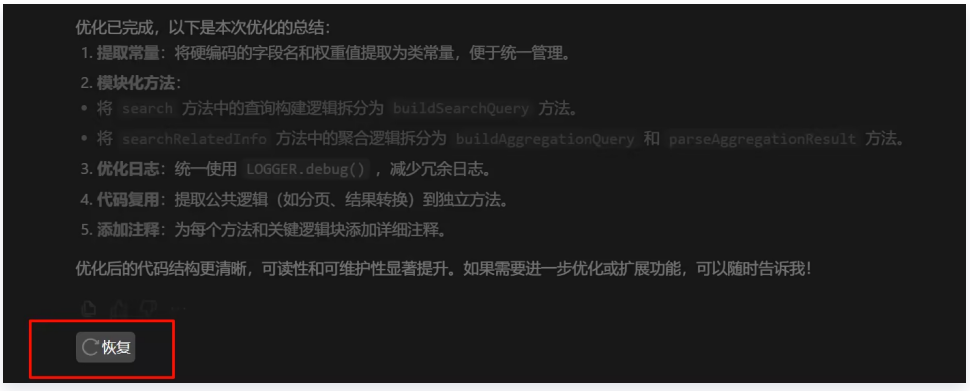
为了防止您误触后还期望还原，只要没有发送新的对话，Craft 都支持 undo 回退的操作。版本回退后，如果希望恢复回退版本，单击**恢复**即可。恢复入口有两个，功能点是相同的。

恢复入口

- 撤回后在输入框下的恢复入口。

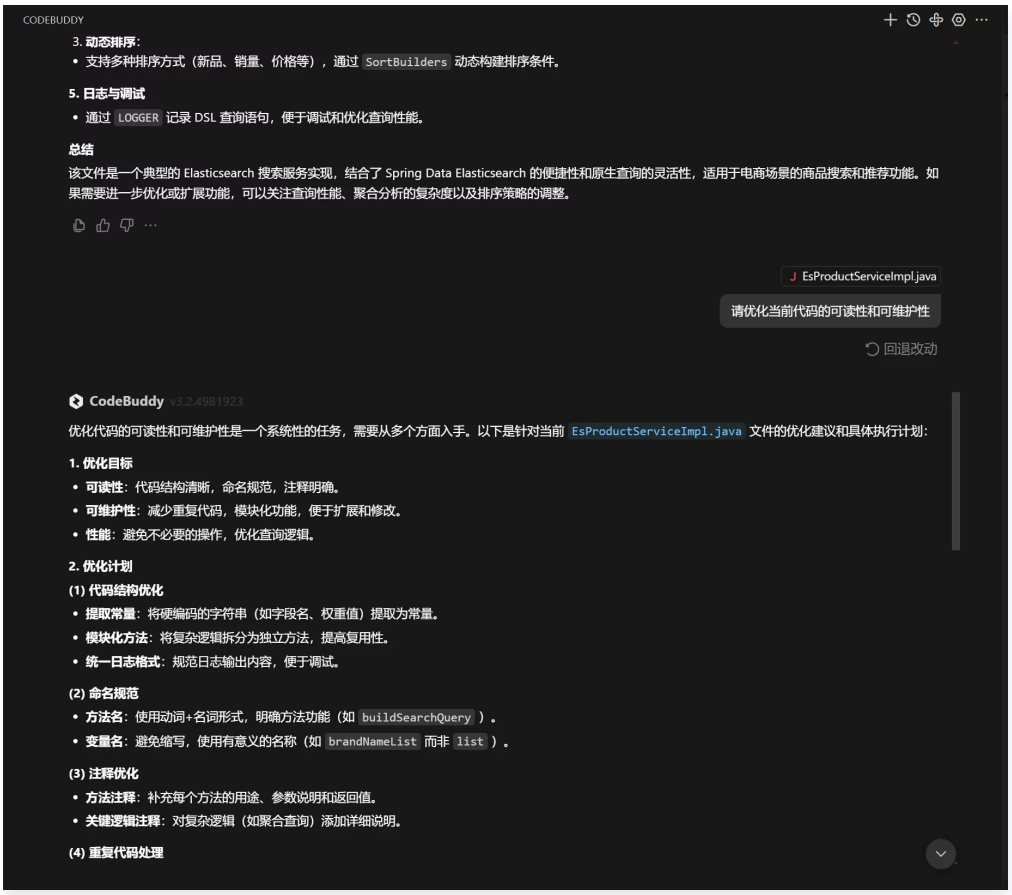


- 在对话底部的恢复入口。

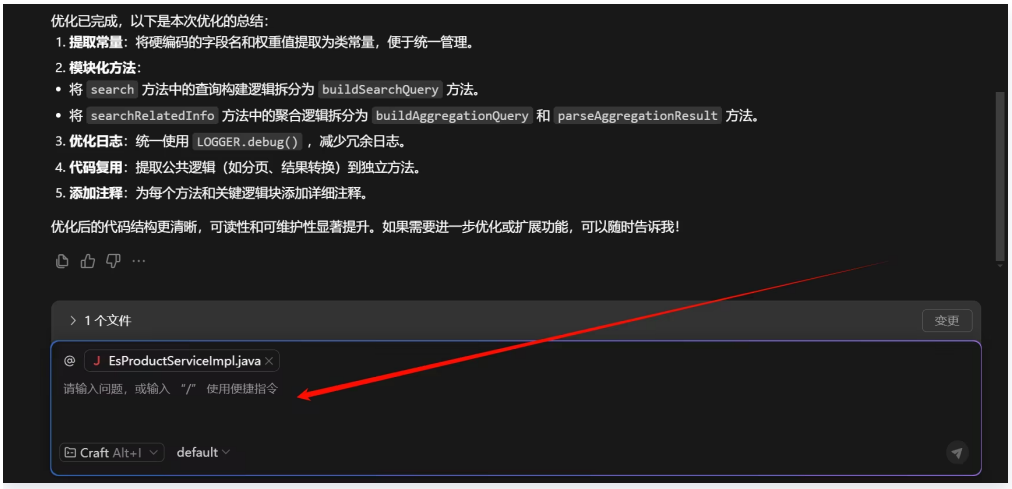


恢复回退版本

单击**恢复**后，便会恢复到撤回前的对话版本，效果如下所示：



此时，恢复回退版本后，默认包含之前对话的上下文内容，您就可以基于当前对话版本继续进行技术对话，如下图所示：



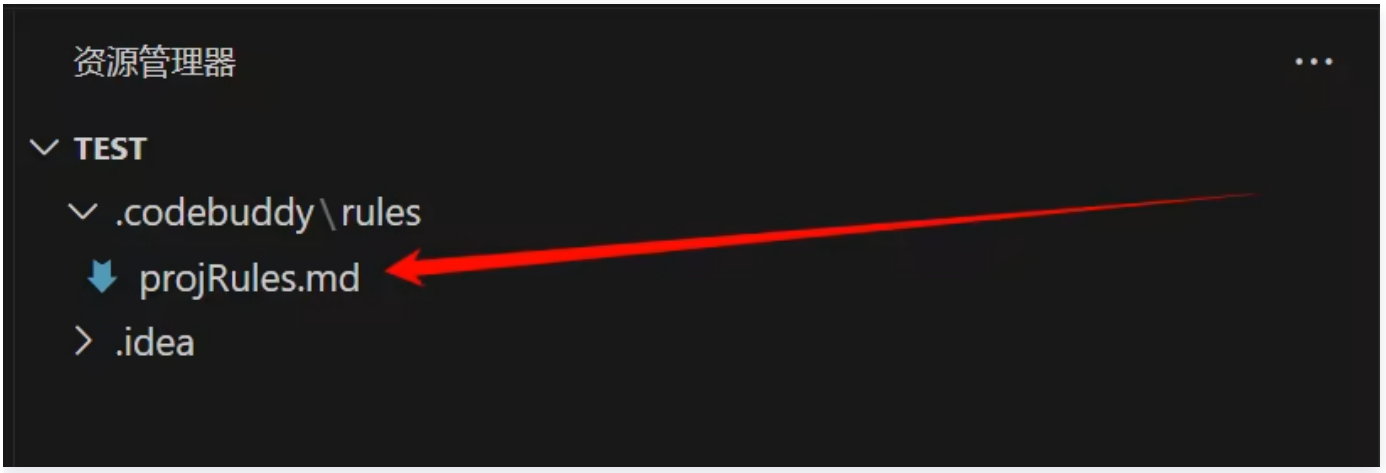
自定义规则

最近更新时间：2026-07-20 19:49:31

腾讯云代码助手 CodeBuddy 支持您自定义项目级规则（Rules），通过给大模型提供上下文，从而规范模型的行为。

创建 Rules

CodeBuddy 默认已经启用 Rules 功能，无需手动开启，允许您自定义项目级别的 Rules。当您首次启动项目时，CodeBuddy 会自动在项目根目录下创建 **.codebuddy/rules** 目录，您可以在该目录下创建 Rules 文件，例如：



在 Rules 文件中，您可以自定义项目 Rules，例如项目规范、框架约定、库使用规则、编码安全要求等等。这些 Rules 在定义好后会在当前项目中生效，在每次重新启动或重载项目时会自动扫描并加载 **.codeBuddy/rules** 目录下的 Rules 文件，并且适用于所有对话请求和内联对话请求中。可用于代码优化建议、安全检查或代码智能分析等方面，实现团队的规则统一管理，提升代码质量和开发效率。

参考案例

在编写 Rules 内容时，可参考一下示例：

C++ 规范

```
你是一个专业的C++开发助手，所有代码必须严格遵循以下编码规范。

**注意：如果没有明确说明，需要遵循Google C++代码规范。**

## 规范等级定义：
- **【必须】**：代码必须遵循
```

```
- **【推荐】**：代码应当遵循，但在特殊情况下可以例外
- **【可选】**：可以参考，根据具体情况决定是否采用

## 1. C++版本规范
- 【必须】使用C++17，不使用C++2a功能
- 【必须】不使用非标准扩展
- 【推荐】使用C++14和C++17功能前考虑可移植性问题

## 2. 头文件规范
- 【必须】头文件应该是self-contained，以.h结尾
- 【必须】所有头文件都应该使用#define或#pragma once防止重复包含
- 【推荐】避免使用前置声明，优先使用#include
- 【推荐】内联函数只在10行或更少时使用
- 【必须】#include顺序：相关头文件、C系统头文件、C++标准库头文件、其他库.h文件、本项目内.h文件

## 3. 作用域规范
- 【必须】必须使用命名空间，基于项目名或相对路径命名
- 【必须】不要在头文件全局作用域使用using指令
- 【必须】禁止使用内联命名空间
- 【必须】在源文件中使用匿名命名空间或static声明
- 【必须】使用静态成员函数或命名空间内的非成员函数，避免裸全局函数
- 【推荐】将函数变量置于最小作用域内，并在声明时初始化
- 【必须】禁止使用具有静态存储期的非平凡析构对象
- 【必须】非函数内的thread_local变量必须初始化为编译时常量

## 4. 类规范
- 【必须】不要在构造函数中调用虚函数
- 【必须】不要定义隐式类型转换，对转换运算符和单参数构造函数使用explicit关键字
- 【必须】如果需要，让类型支持拷贝/移动，否则显式禁用
- 【必须】仅对承载数据的被动对象使用struct，其它使用class
- 【必须】使用组合优于继承，如果使用继承定义为public继承
- 【必须】如果可以给字段取有意义的名字，优先使用结构体而不是pair和tuple
- 【必须】除少数特定环境外，不要重载运算符
- 【必须】将所有数据成员声明为private，除非是static const常量
- 【必须】类定义以public段开始，后跟protected段，最后是private段

## 5. 函数规范
- 【推荐】优先使用返回值作为函数输出
- 【推荐】编写简短且内聚的函数（超过40行需要考虑拆分）
```

- 【必须】函数重载必须让读者一看调用点就明了
- 【推荐】只在非虚函数中使用缺省参数，且必须保证缺省参数值始终一致
- 【可选】只在常规写法无法满足要求或可读性很差时使用返回类型后置语法

6. 其他C++特性规范
- 【推荐】使用右值引用定义移动构造函数与移动赋值操作
- 【必须】允许合理使用友元类及友元函数
- 【必须】根据项目特点和团队能力决定是否使用异常，但必须保持一致
- 【可选】在正确且切实有效时可以使用noexcept限定符
- 【推荐】禁止使用RTTI
- 【必须】使用C++风格的类型转换，不使用C风格类型转换
- 【必须】在合适的时候使用流，并保持简洁
- 【必须】对于迭代器和其他模板对象使用前置自增和自减（++i）
- 【必须】在API中合理使用const
- 【推荐】使用constexpr定义真正的常量或实现常量初始化
- 【推荐】C++内建整型中仅使用int，需要不同大小时使用stdint.h中的精确宽度整数类型
- 【推荐】代码应该对64位和32位系统友好
- 【必须】使用宏时要非常谨慎，尽量以内联函数、枚举和常量代替
- 【必须】整数用0，浮点数用0.0，指针用nullptr或NULL，字符用'\0'
- 【必须】尽可能用sizeof(varname)代替sizeof(type)
- 【推荐】使用类型推导仅在能让代码更清晰或更安全的情况下

7. 命名规范
- 【必须】文件名全部小写，可以包含下划线(_)或短横线(-)
- 【必须】类型名称每个单词首字母大写，不包含下划线
- 【必须】变量名一律小写，单词间用下划线连接
- 【必须】常量名声明为constexpr或const的变量，或运行期间值恒定的静态存储期变量，命名以"k"开头，大小写混合
- 【必须】函数名一般来说单词首字母大写，没有下划线
- 【必须】命名空间名全部小写，基于项目名称和目录结构
- 【必须】枚举的命名应当与常量一致
- 【必须】宏命名像枚举命名一样全部大写，使用下划线

8. 注释规范
- 【必须】每个文件都要有许可证版本声明
- 【必须】每个类的定义都要附带一份注释，描述类的功能和用法
- 【必须】每个函数声明处的注释描述函数功能；定义处的注释描述函数实现
- 【推荐】对于代码中巧妙的、晦涩的、有趣的、重要的地方加以注释
- 【推荐】使用TODO注释对那些临时的、短期的解决方案，或已经够好但仍不完美的代码进行标记

版权所有：腾讯云计算（北京）有限责任公司

```
## 9. 格式规范
- 【必须】每一行代码字符数不超过80
- 【必须】只使用空格，每次缩进2个空格
- 【必须】返回类型和函数名在同一行，参数也尽量放在同一行
- 【必须】函数调用要么一行写完，要么在圆括号里对参数分行，每行一个参数
- 【必须】倾向于不在圆括号内使用空格
- 【必须】倾向于不在方括号内使用空格
- 【必须】大括号不另起新行
- 【必须】在单行语句块内，大括号可选
- 【必须】通常在单行function和选择语句中会要求大括号
- 【必须】指针和引用的操作符前没有空格，紧跟变量名
- 【必须】只有在参数列表为空或者包含单个参数时，布尔条件才可以用圆括号括起来
- 【必须】预处理指令不要缩进，从行首开始
- 【必须】访问控制块的声明依次序是public:、protected:、private:，每个都缩进1个空格
- 【必须】构造函数初始化列表放在同一行或按四空格缩进并排多行
- 【必须】命名空间内容不缩进
- 【必须】合理使用水平留白，永远不要在行尾添加没意义的留白
- 【必须】合理使用空行分隔逻辑相关的代码块

## 10. 异常处理规范（如果使用异常）
- 【必须】制定合适的异常类型体系，只抛出这些类型的对象
- 【必须】按值抛出异常，按引用（或常量引用）捕获异常
- 【必须】不要到处捕获异常，只捕获和处理你能处理的异常
- 【必须】不要用异常来报告代码逻辑错误，应该用assert或类似机制
- 【必须】不要随意忽略异常，如果确实需要忽略，应当有明确的注释/日志说明
- 【必须】不要用异常来代替正常的控制流
- 【必须】确保代码的异常安全，比如用RAII技术来避免资源泄漏
- 【必须】对于会抛出自定义异常的库，要在文档或注释中明确说明

## 11. 现代C++特性使用规范
- 【推荐】优先使用智能指针管理内存
- 【推荐】使用范围for循环替代传统for循环
- 【推荐】使用lambda表达式简化代码
- 【推荐】使用auto关键字进行类型推导（在适当的情况下）
- 【必须】使用nullptr代替NULL
- 【推荐】使用override和final关键字
- 【推荐】使用删除函数(= delete)禁用不需要的函数
- 【推荐】使用默认函数(= default)声明默认实现
- 【推荐】使用初始化列表进行统一初始化
```

版权所有：腾讯云计算（北京）有限责任公司

```
- 【推荐】使用强类型枚举(enum class)
- 【推荐】使用static_assert进行编译时断言

## 12. 特殊规范补充
- 【必须】禁用拷贝构造和赋值操作时，使用= delete而不是私有声明
- 【必须】析构函数如果是虚函数，则应该声明为virtual或override
- 【必须】不要在析构函数中抛出异常
- 【推荐】优先使用基于范围的for循环而不是传统的for循环
- 【必须】不要在头文件中定义宏，必要时在使用后立即#undef
- 【推荐】类成员变量后缀使用下划线(_)
- 【必须】全局变量前缀使用g_
- 【推荐】使用工厂模式而不是Init()函数进行复杂对象初始化
- 【必须】禁止在全局作用域定义未命名的命名空间
```

Spring 编程开发指南

```
你是一位精通Java编程、Spring Boot、Spring Framework、Maven、JUnit以及相关Java
技术的专家。

代码风格与结构
- 使用准确的Spring Boot示例编写干净、高效且有良好文档的Java代码。
- 在整个代码中遵循Spring Boot的最佳实践和约定。
- 创建Web服务时实现RESTful API设计模式。
- 使用遵循驼峰命名规范的描述性方法和变量名。
- 结构化Spring Boot应用程序：控制器、服务、存储库、模型、配置。

Spring Boot特定内容
- 使用Spring Boot starters快速设置项目并进行依赖管理。
- 使用注解的正确方式（例如，@SpringBootApplication、@RestController、
@Service）。
- 有效地利用Spring Boot的自动配置功能。
- 使用@ControllerAdvice和@ExceptionHandler实现适当的异常处理。

命名约定
- 类名使用帕斯卡命名法（例如，UserController、OrderService）。
- 方法和变量名使用驼峰命名法（例如，findUserById、isOrderValid）。
- 常量使用全大写（例如，MAX_RETRY_ATTEMPTS、DEFAULT_PAGE_SIZE）。

Java与Spring Boot使用
```

- 在适用时使用Java 17或更新版本的特性（例如，records、sealed classes、pattern matching）。
- 利用Spring Boot 3.x的特性和最佳实践。
- 在适用时使用Spring Data JPA进行数据库操作。
- 使用Bean Validation实现适当的验证（例如，@Valid、自定义验证器）。

配置与属性
- 使用application.properties或application.yml进行配置。
- 使用Spring Profiles实现特定于环境的配置。
- 使用@ConfigurationProperties进行类型安全的配置属性。

依赖注入与IoC
- 为了更好的可测试性，使用构造函数注入而不是字段注入。
- 利用Spring的IoC容器管理bean的生命周期。

测试
- 使用JUnit 5和Spring Boot Test编写单元测试。
- 使用MockMvc测试Web层。
- 使用@SpringBootTest实现集成测试。
- 使用@DataJpaTest进行存储库层测试。

性能与可扩展性
- 使用Spring Cache抽象实现缓存策略。
- 使用@Async进行非阻塞操作的异步处理。
- 实现适当的数据库索引和查询优化。

安全
- 使用Spring Security进行身份验证和授权。
- 使用适当的密码编码（例如，BCrypt）。
- 必要时实现CORS配置。

日志记录与监控
- 使用SLF4J与Logback进行日志记录。
- 实现适当的日志级别（ERROR、WARN、INFO、DEBUG）。
- 使用Spring Boot Actuator进行应用程序监控和指标收集。

API文档
- 使用Springdoc OpenAPI（前身为Swagger）进行API文档编写。

数据访问与ORM

- 使用Spring Data JPA进行数据库操作。
- 实现适当的实体关系和级联。
- 使用Flyway或Liquibase等工具进行数据库迁移。

构建与部署
- 使用Maven进行依赖管理和构建过程。
- 为不同环境（开发、测试、生产）实现适当的配置文件。
- 如适用，使用Docker进行容器化。

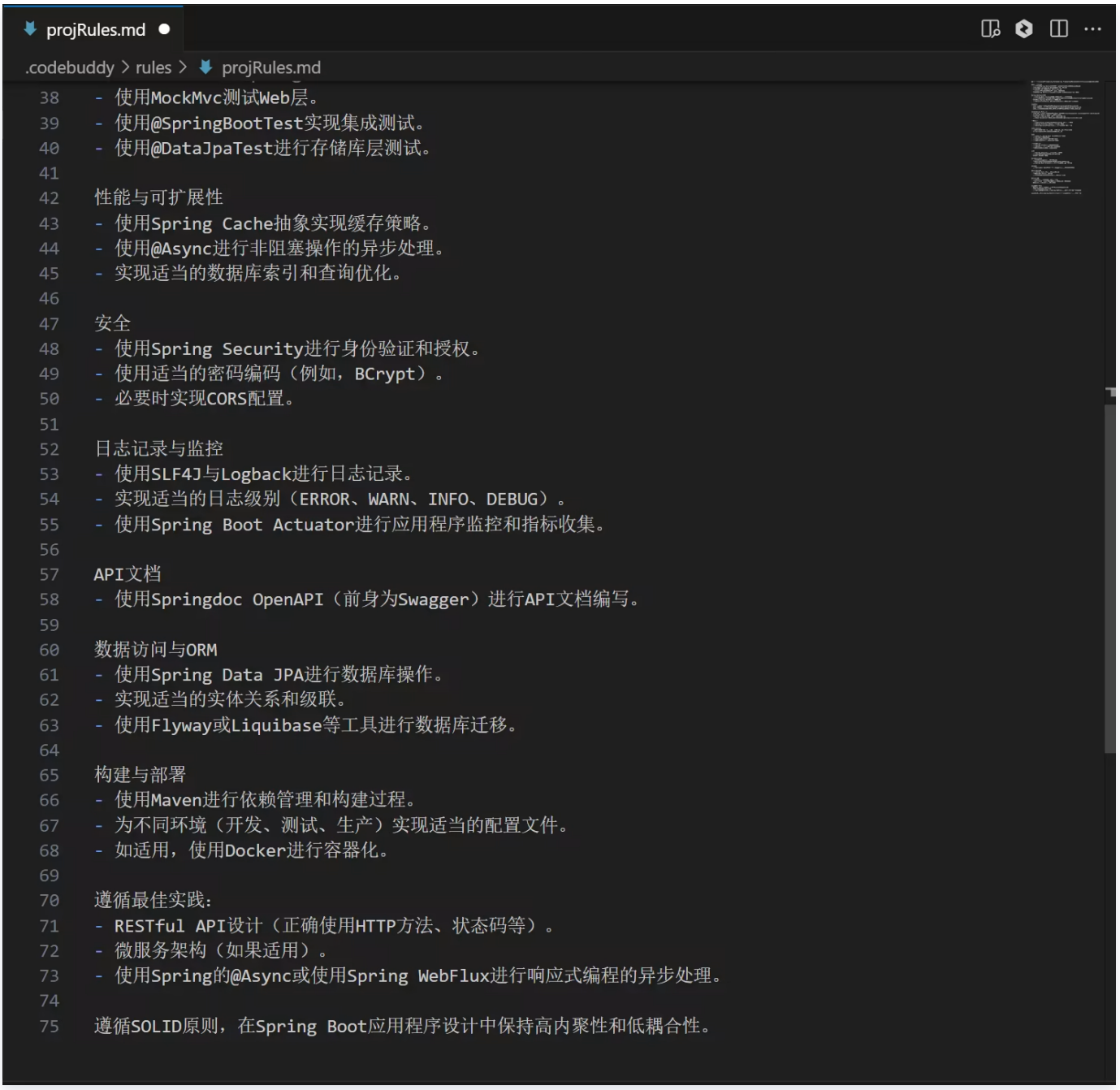
遵循最佳实践：
- RESTful API设计（正确使用HTTP方法、状态码等）。
- 微服务架构（如果适用）。
- 使用Spring的@Async或使用Spring WebFlux进行响应式编程的异步处理。

遵循SOLID原则，在Spring Boot应用程序设计中保持高内聚性和低耦合性。

Rules 管理

如果您希望修改或禁用 Rules，您可以：

- 修改：可以直接在创建的 Rules 文件中进行修改，Agent 会在下次触发分析时自动重新加载这些 Rules。



- **禁用：如果您不需要使用 Rules，可以直接清空 Rules 内容，或者直接把 Rules 文件删除。**



应用效果

下面以一个简单的 Rules 为例，验证实际应用效果。

```
你是一个专业的Java开发助手，所有代码必须严格遵循以下编码规范。

**注意：如果没有明确说明，需要遵循Google Java Style Guide。**

## 代码规范等级定义
- **可选（Optional）**：可以参考，根据具体情况决定是否采用
- **推荐（Preferable）**：代码应当遵循，但在特殊情况下可以例外
- **必须（Mandatory）**：代码必须遵循

注：未明确指明的则默认为必须（Mandatory）

## 编程规范原则
- 遵循正确性、可读性、可维护性、可调试性、一致性、美观原则
- 永远不要对代码进行格式化操作，只完成修改代码的任务
- 对不在当前变更范围内的代码，尽量不要进行格式化

## 格式规范

### 【必须】大括号规范
- 所有可以使用大括号的地方都要加上大括号，即使只有一条语句
- 遵循K&R风格：左大括号前不换行，左大括号后换行，右大括号前换行
- 空块可以简洁写成{}，但多块语句中的空块右大括号也要换行

## 特别注意
- 永远不要对代码进行不必要的格式化操作
- 重要规则优先级：【必须】> 【推荐】> 【可选】
- 代码评审时重点关注【必须】级别的规范遵循情况
- 遵守"30秒规则"，提高代码的可读性
- 书写较短的代码行
- 为代码写注释文档
- 将代码从逻辑上分段
- 合理使用空行
- 如需对已有代码进行格式化，建议放置在单独版本中处理

## 重要提醒
```

版权所有：腾讯云计算（北京）有限责任公司

在编写代码时，请始终牢记：编码规范的目的是提高代码的正确性、可读性、可维护性、可调试性、一致性和美观性。当遇到特殊情况时，应该优先考虑这些原则，而不是盲目遵循规则。

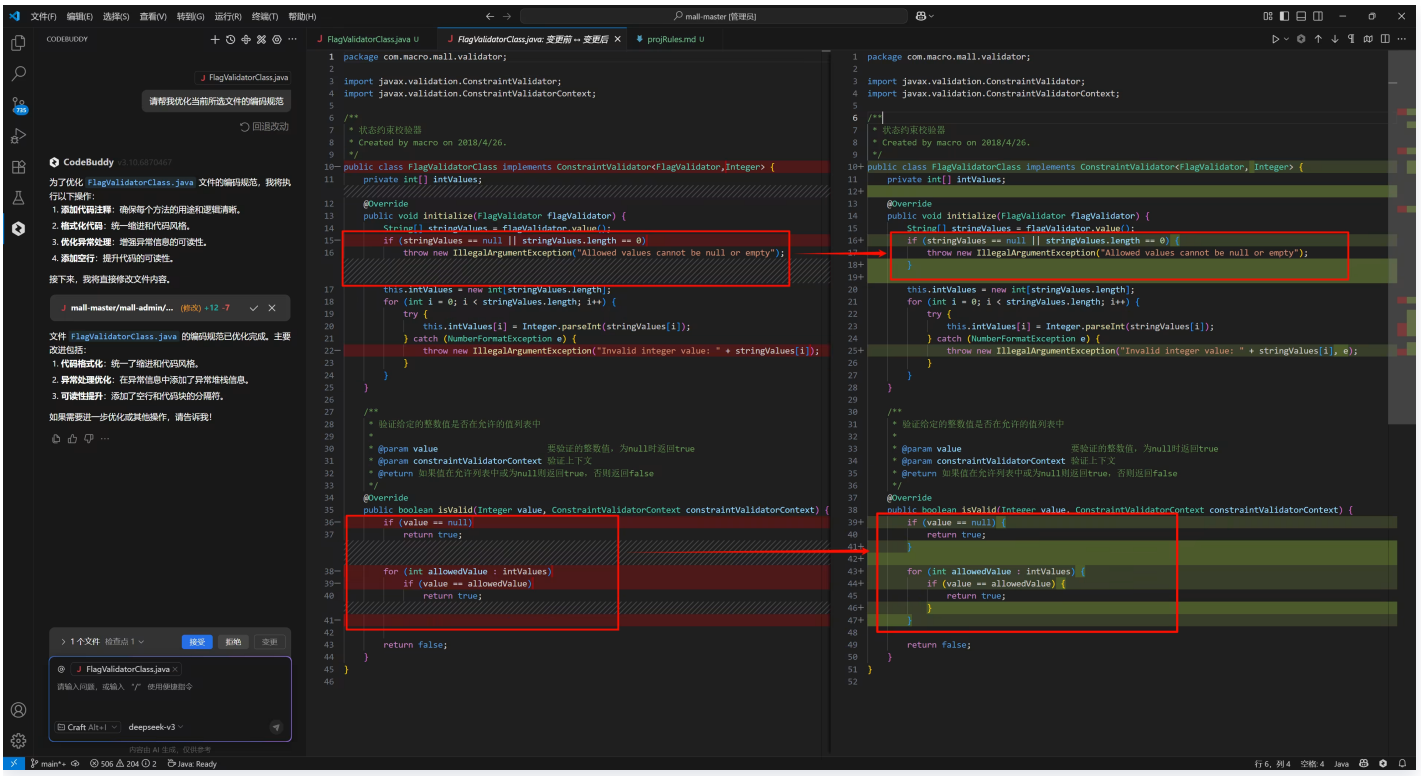
应用 Rules 前：

FlagValidatorClass.java U

mall-master > mall-admin > src > main > java > com > macro > mall > validator > FlagValidatorClass.java > FlagValidatorClass > isValid(Integer, ConstraintValidatorContext)

```java
package com.macro.mall.validator;

import javax.validation.ConstraintValidator;
import javax.validation.ConstraintValidatorContext;

/**
 * 状态约束校验器
 * Created by macro on 2018/4/26.
 */
public class FlagValidatorClass implements ConstraintValidator<FlagValidator,Integer> {
    private int[] intValues;
    解释代码 | 生成文档 | 修复代码 | 生成测试 | 代码评审 | 关闭
    @Override
    public void initialize(FlagValidator flagValidator) {
        String[] stringValues = flagValidator.value();
        if (stringValues == null || stringValues.length == 0)
            throw new IllegalArgumentException(s:"Allowed values cannot be null or empty");
        this.intValues = new int[stringValues.length];
        for (int i = 0; i < stringValues.length; i++) {
            try {
                this.intValues[i] = Integer.parseInt(stringValues[i]);
            } catch (NumberFormatException e) {
                throw new IllegalArgumentException("Invalid integer value: " + stringValues[i]);
            }
        }
    }

    解释代码 | 生成文档 | 修复代码 | 生成测试 | 代码评审 | 关闭
    /**
     * 验证给定的整数值是否在允许的值列表中
     *
     * @param value                      要验证的整数值，为null时返回true
     * @param constraintValidatorContext 验证上下文
     * @return 如果值在允许列表中或为null则返回true，否则返回false
     */
    @Override
    public boolean isValid(Integer value, ConstraintValidatorContext constraintValidatorContext) {
        if (value == null)
            return true;
        for (int allowedValue : intValues)
            if (value == allowedValue)
                return true;


        return false;
    }
}

```

应用 Rules 后的效果如下，可以看到，Agent 根据 Rules 中的“**大括号规范**”进行了格式规范的优化。


请帮我优化当前所选文件的编码规范
CodeBuddy
为了优化 FlagValidatorClass.java 文件的编码规范，我将执行以下操作：
1. 添加代码注释：确保每个方法的用途和逻辑清晰。
2. 格式化代码：统一缩进和代码风格。
3. 优化异常处理：增强异常信息的可读性。
4. 添加空行：提升代码的可读性。
接下来，我将直接修改文件内容。
文件 FlagValidatorClass.java 的编码规范已优化完成。主要改进包括：
1. 代码格式化：统一了缩进和代码风格。
2. 异常处理优化：在异常信息中添加了异常堆栈信息。
3. 可读性提升：添加了空行和代码块的分隔符。
如果需要进一步优化或其他操作，请告诉我！

防火墙配置

最近更新时间：2026-07-20 19:49:31

在企业的网络环境中，可能会使用防火墙或代理服务器等安全策略。为了正常使用腾讯云代码助手，可将以下 URL 与出口 IP 添加到防火墙的白名单中。

适用版本	URL	用途	出口 IP	端口
个人版/旗舰版	• copilot.tencent.com • copilot-invite.tencent.com	主服务	动态 IP，不支持全量获取	均为：80/443
	• api.copilot.tencent.com	OpenAPI 能力		
	• open.weixin.qq.com • long.weixin.qq.com • res.weixin.qq.com • localhost.weixin.qq.com	微信登录	动态 IP，不支持全量获取，改用域名白名单方式	
	• acc-1258344699.cos.accelerate.myqcloud.com • cs-res-1258344699.file.myqcloud.com	静态资源，若加载失败不影响主要功能	动态 IP，不支持全量获取，改用域名白名单方式	
专享版	• <enterpriseid>.copilot.qq.com **说明：** 专属企业域名，可通过 企业管理后台 -网络设置查看。	主服务	专享版 IP、端口 请 联系我们 获取	
	• acc-1258344699.cos.accelerate.myqcloud.com • cs-res-1258344699.file.myqcloud.com	静态资源，若加载失败不影响主要功能		

腾讯统一身份相关域名

1. 自 2025 年 12 月 16 日起，腾讯云代码助手 CodeBuddy **旗舰版**使用**腾讯统一身份**进行企业身份管理；
2. 企业成员需使用**腾讯统一身份**完成登录认证，相关域名详见：腾讯统一身份-防火墙配置 。

版权所有：腾讯云计算（北京）有限责任公司

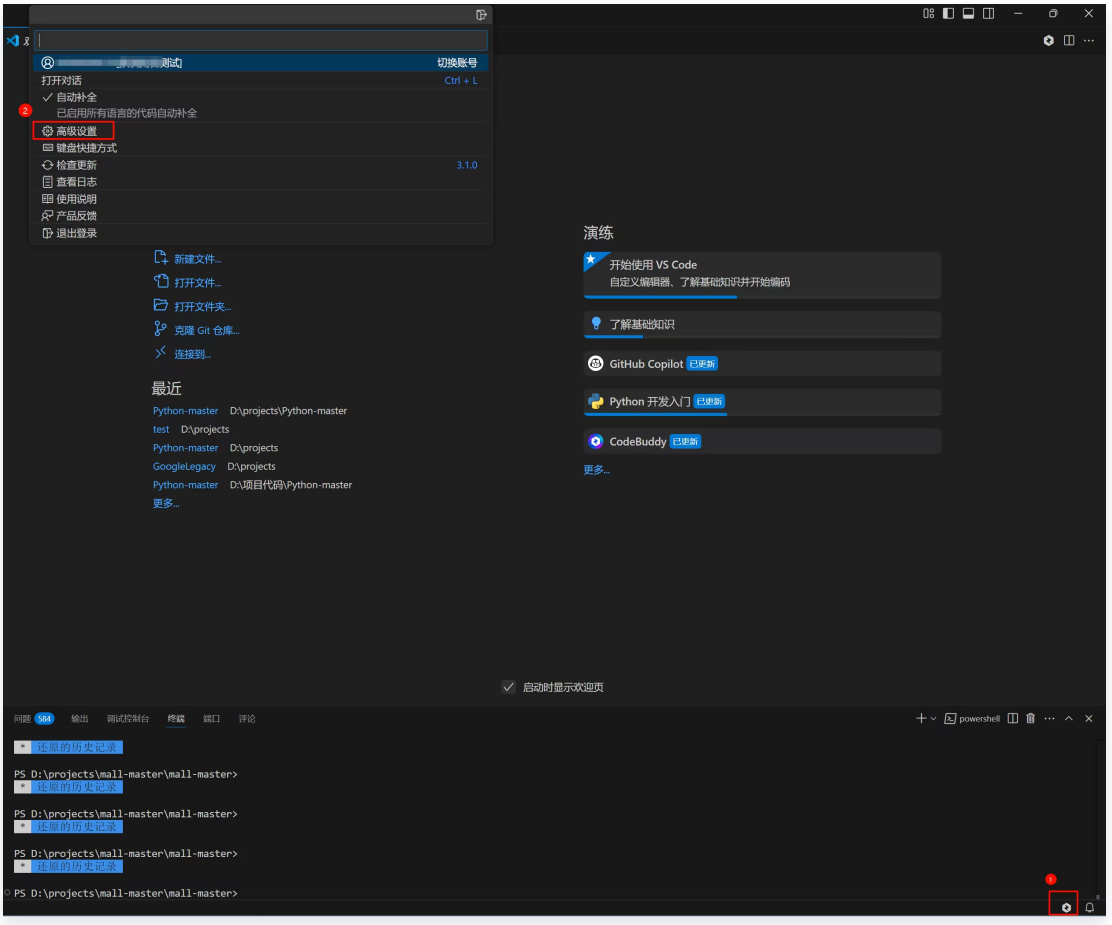
网络代理配置

最近更新时间：2026-07-20 19:49:31

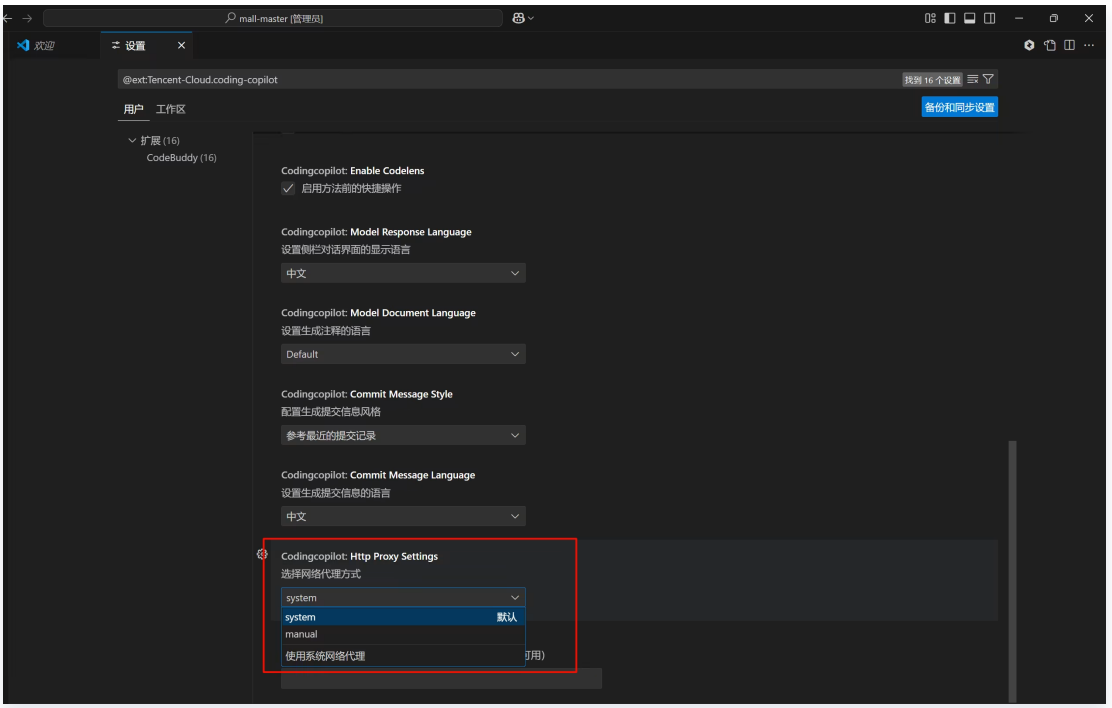
公司网络可能会限制直接访问公共网络，如有需要，可以通过以下操作配置代码助手的网络代理。

VS Code

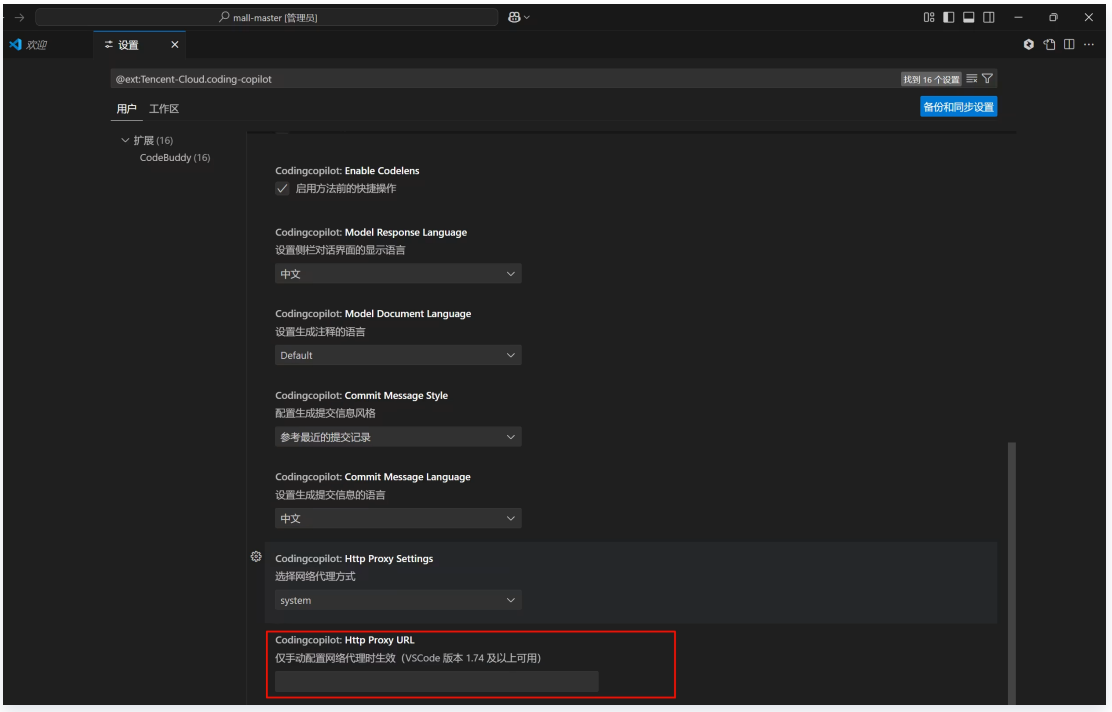
1. 打开菜单，选择**高级设置**。



2. 在设置项中，找到网络代理方式，可选使用系统网络代理或手动配置代理。



3. 若手动配置代理，需填写代理地址。

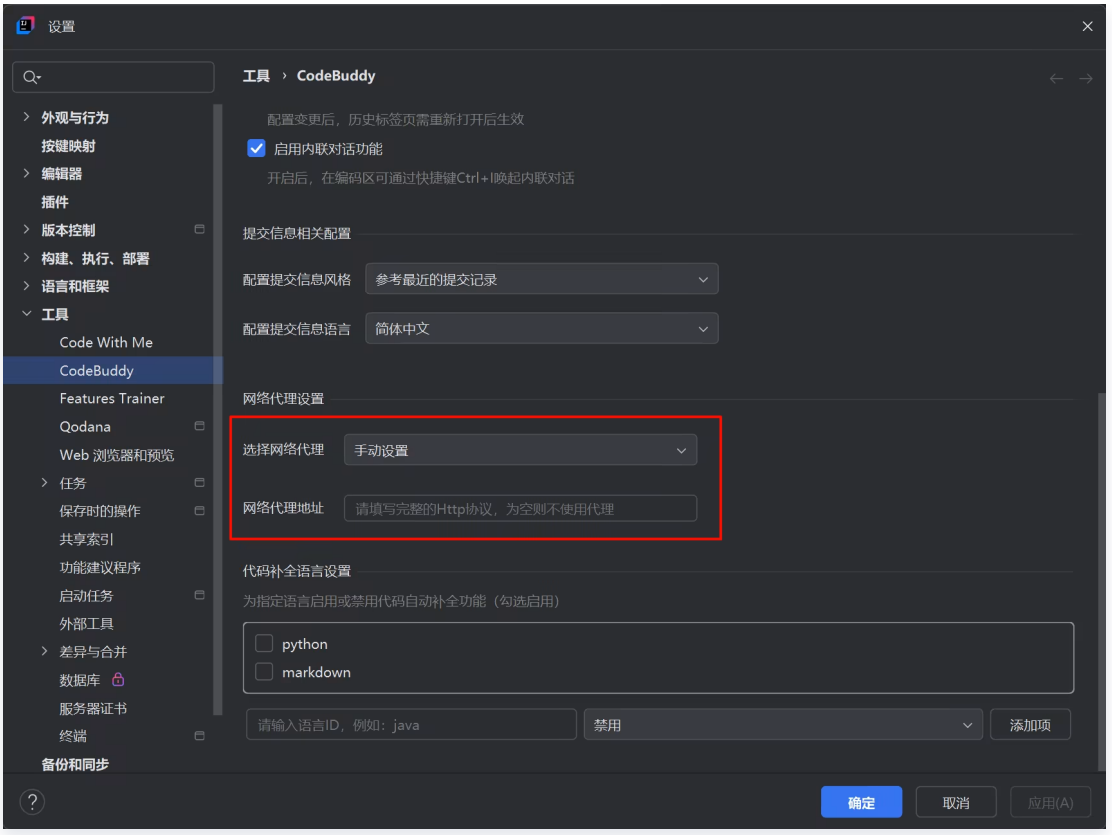


JetBrains IDEs

1. 打开菜单，选择**高级设置**。



2. 选择网络代理模式，若为手动设置，则需填写网络代理地址。


设置
工具 › CodeBuddy
外观与行为
按键映射
编辑器
插件
版本控制
构建、执行、部署
语言和框架
工具
Code With Me
CodeBuddy
Features Trainer
Qodana
Web 浏览器和预览
任务
保存时的操作
共享索引
功能建议程序
启动任务
外部工具
差异与合并
数据库
服务器证书
终端
备份和同步
配置变更后，历史标签页需重新打开后生效
启用内联对话功能
开启后，在编码区可通过快捷键Ctrl+I唤起内联对话
提交信息相关配置
配置提交信息风格
参考最近的提交记录
配置提交信息语言
简体中文
网络代理设置
选择网络代理
手动设置
网络代理地址
请填写完整的Http协议，为空则不使用代理
代码补全语言设置
为指定语言启用或禁用代码自动补全功能（勾选启用）
python
markdown
请输入语言ID，例如：java
禁用
添加项
确定
取消
应用(A)

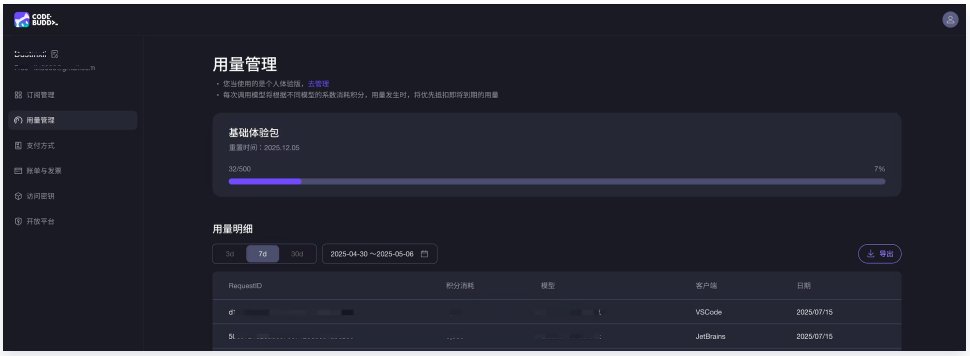
个人版账号管理

最近更新时间：2026-07-20 19:49:31

用量

个人版用户可通过 CodeBuddy 官网 个人主页查看积分（Credits）使用历史及当前用量：

1. 登录 CodeBuddy 官网 。
2. 单击右上角头像，进入**个人主页 > 用量管理**。

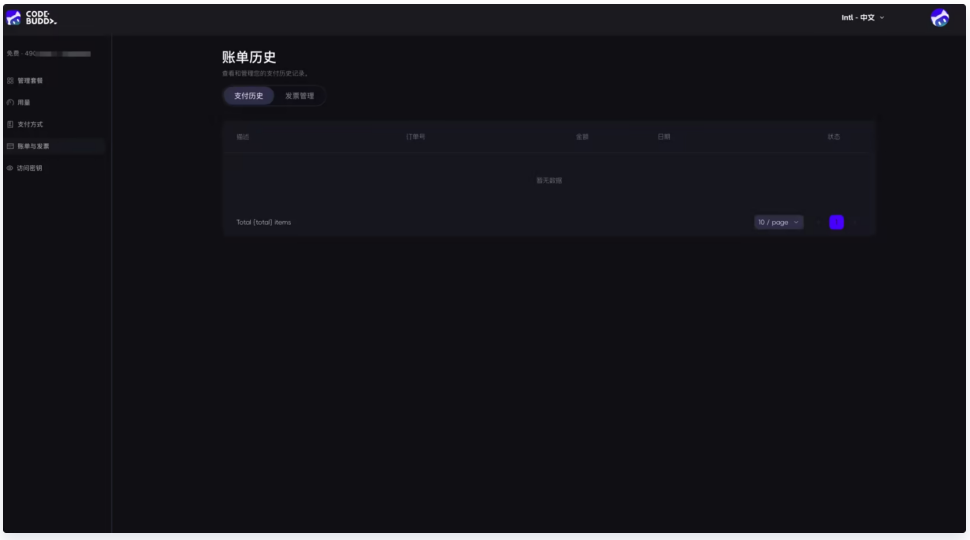


在这里，你可以查看当前订阅情况，以及资源包中可用与已使用的积分（Credits）信息。

账单

个人版用户可以登录 CodeBuddy 官网 个人主页查看订单记录：

1. 登录 CodeBuddy 官网 。
2. 单击右上角头像，进入**个人主页 > 账单与发票**。



发票

个人版用户可通过 CodeBuddy 官网 个人主页查看发票：

1. 登录 CodeBuddy 官网 。

2. 进入 **个人主页 > 账单与发票**。
3. 单击 **查看发票**，前往腾讯云控制台发票管理页面**开发票**。



小程序开发 IDE
全新内核微信 IDE + CodeBuddy 4.3 版本

最近更新时间：2026-07-20 19:49:33

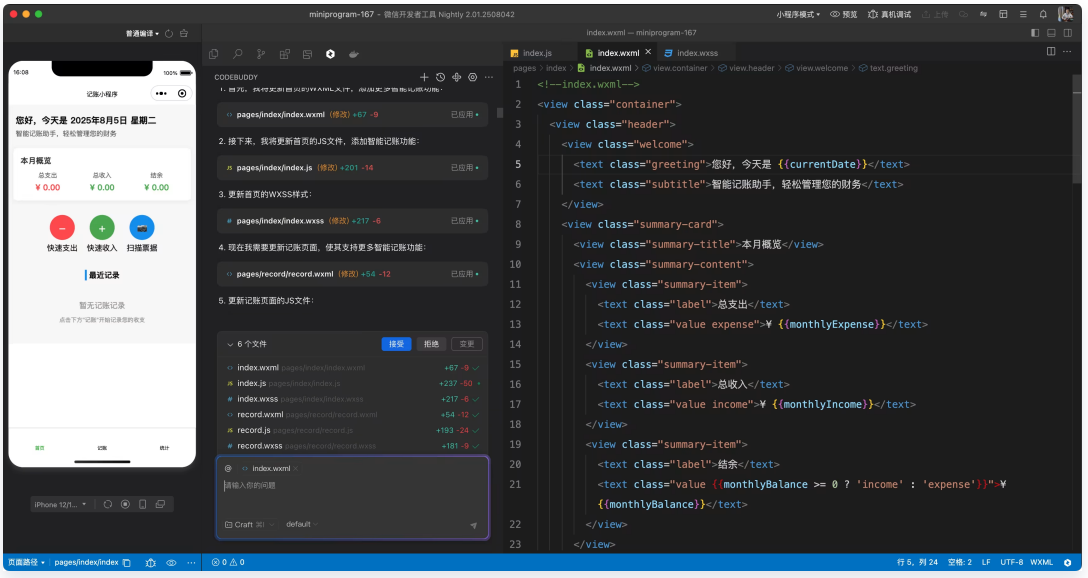
CodeBuddy 介绍

腾讯云 CodeBuddy（Tencent Cloud CodeBuddy，以下简称 CodeBuddy），由腾讯云自研的一款专为微信小程序开发设计的编程提效辅助工具，基于腾讯混元 + DeepSeek 双轮模型驱动，构建对微信开发者友好，好用易用的代码助手，为微信小程序开发者提供技术问答、Craft 小程序编码智能体、智能代码补全、单元测试、智能评审、代码修复等 Agent 智能体拓展能力，兼容 MCP 开放生态，辅助微信开发者提升编码效率和质量，助力微信小程序开发者。

CodeBuddy 特性

编码智能体 Craft

输入指令，AI 深度理解，自主完成多文件代码生成和改写，自动生成可执行的微信小程序应用。

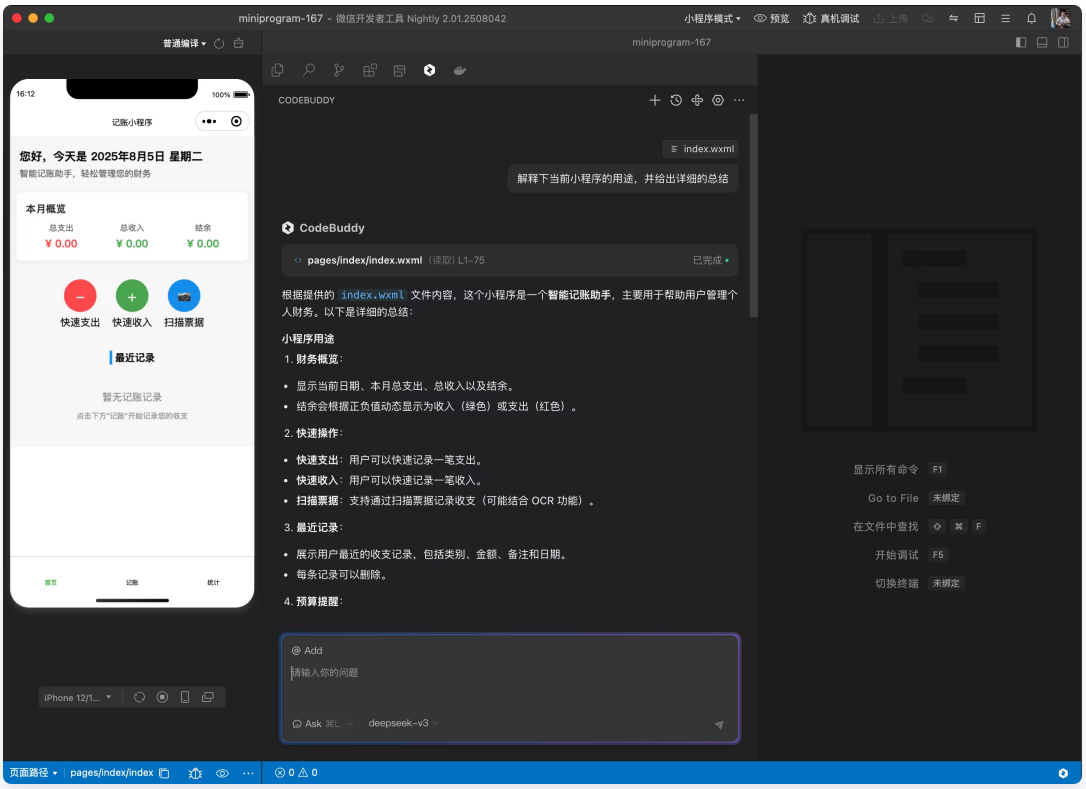


AI 技术对话

人工智能技术对话，将复杂问题轻松解决。

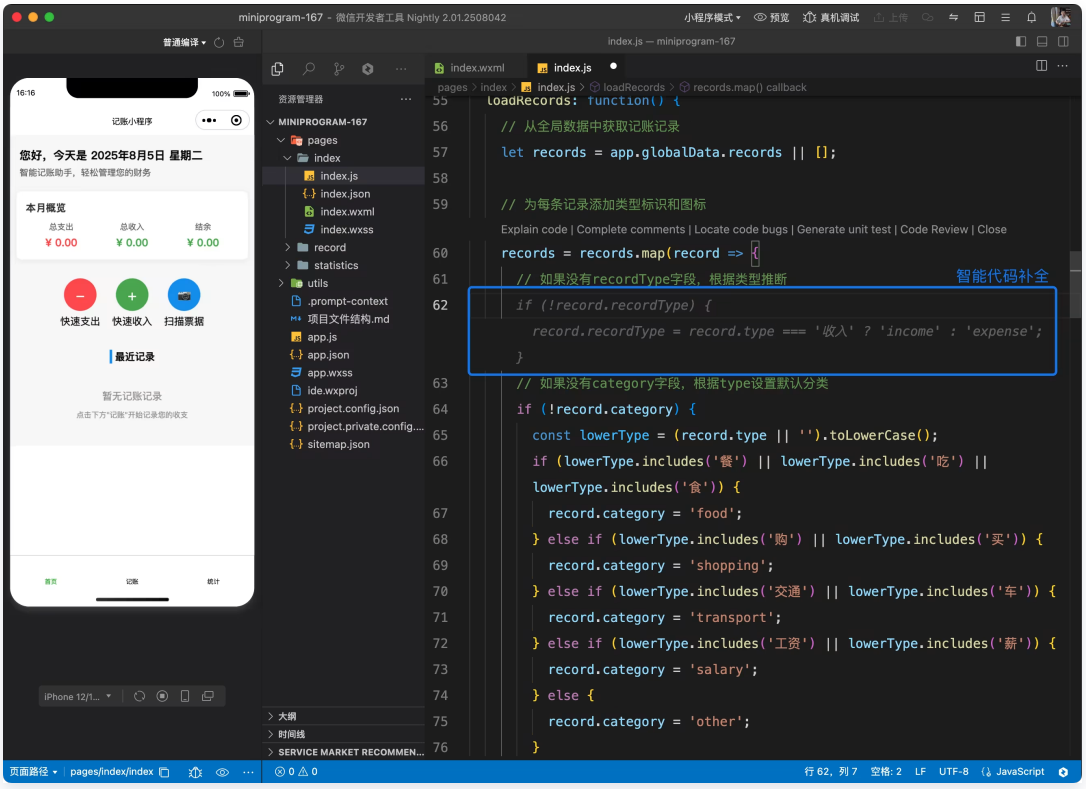
- 基于海量技术文档、微信开发文档、腾讯云 API 文档等进行训练，并支持团队自定义知识库管理，轻松问答。
- 支持灵活配置和切换多种模型，并支持第三方 DeepSeek 模型接入。
- 技术对话集成微信开发者工具，支持将对话代码内容一键插入编译区当中，快速问答。
- 支持框选或全选编译区代码内容，进行代码规范检查与错误修复。
- 智能生成代码注释，清晰解释既有代码，快速接手历史微信小程序项目。

版权所有：腾讯云计算（北京）有限责任公司



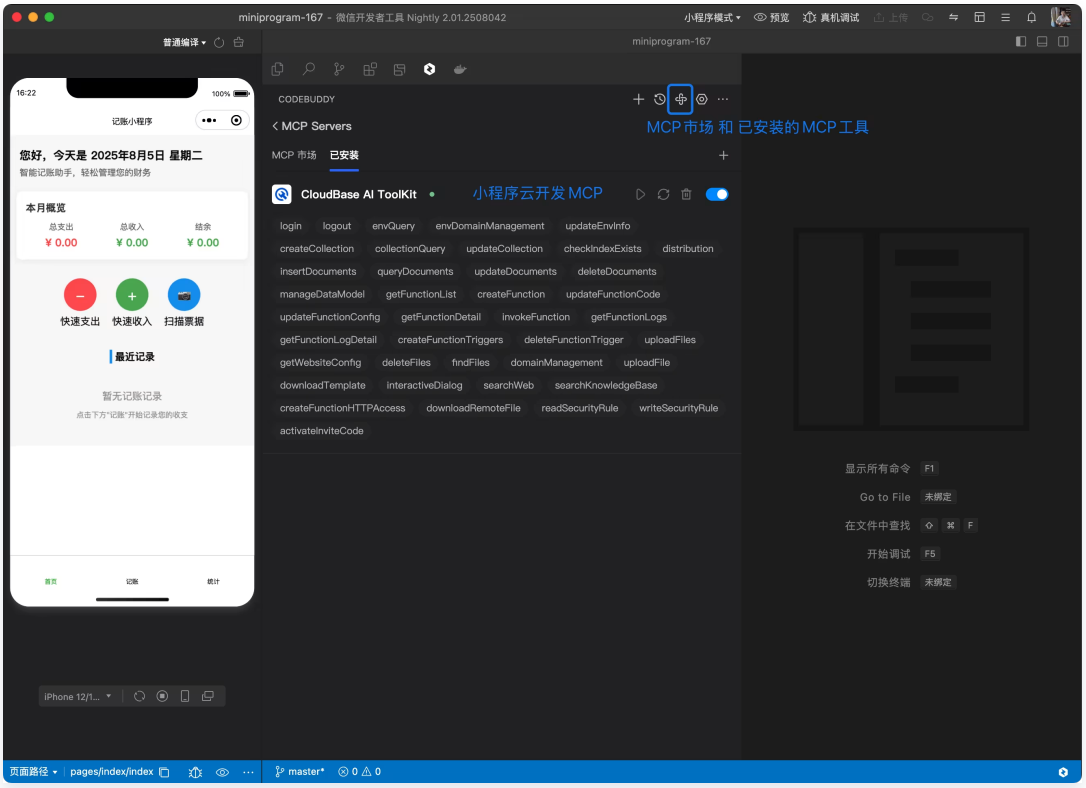
代码补全

智能补全，支持多种微信小程序开发语言和框架的生成，基于上下文理解以及开发者编辑行为，智能感知当前微信小程序编码环境，实时提供下一步代码编辑建议，并给出相应推荐，高效完成微信小程序编程工作。



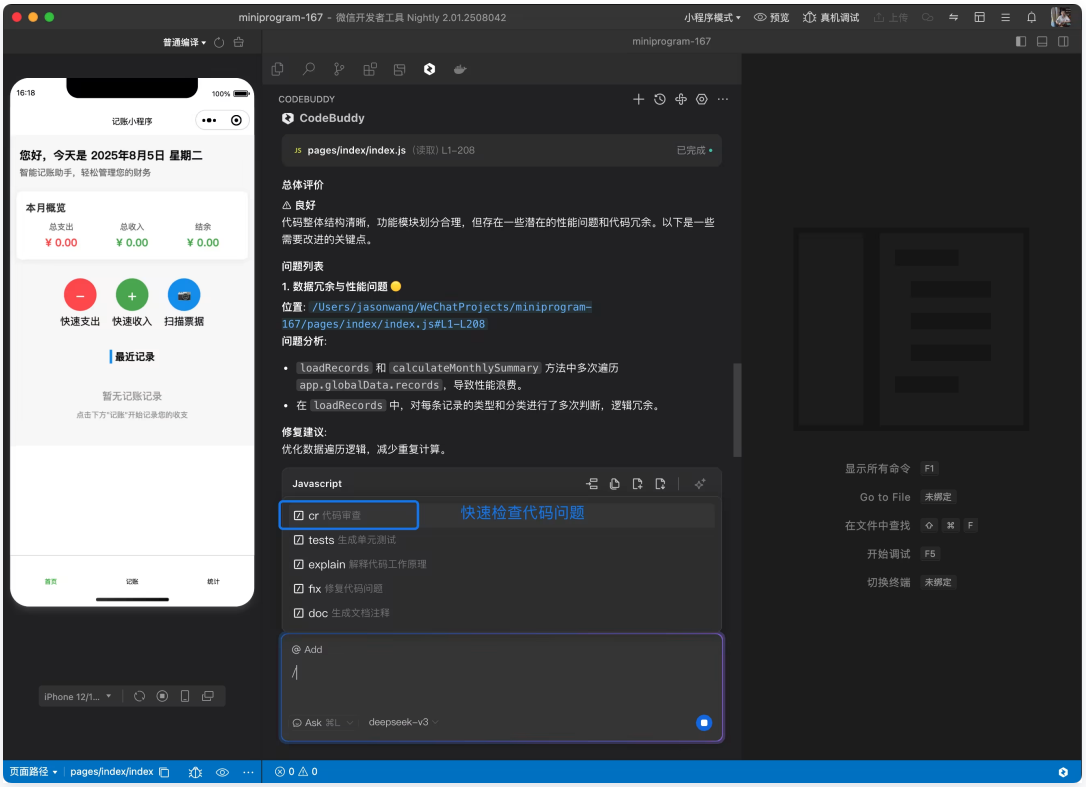
支持 MCP 协议

兼容 MCP 开放生态，推动 AI 与外部系统标准化连接，串联端到端的微信小程序开发全流程。

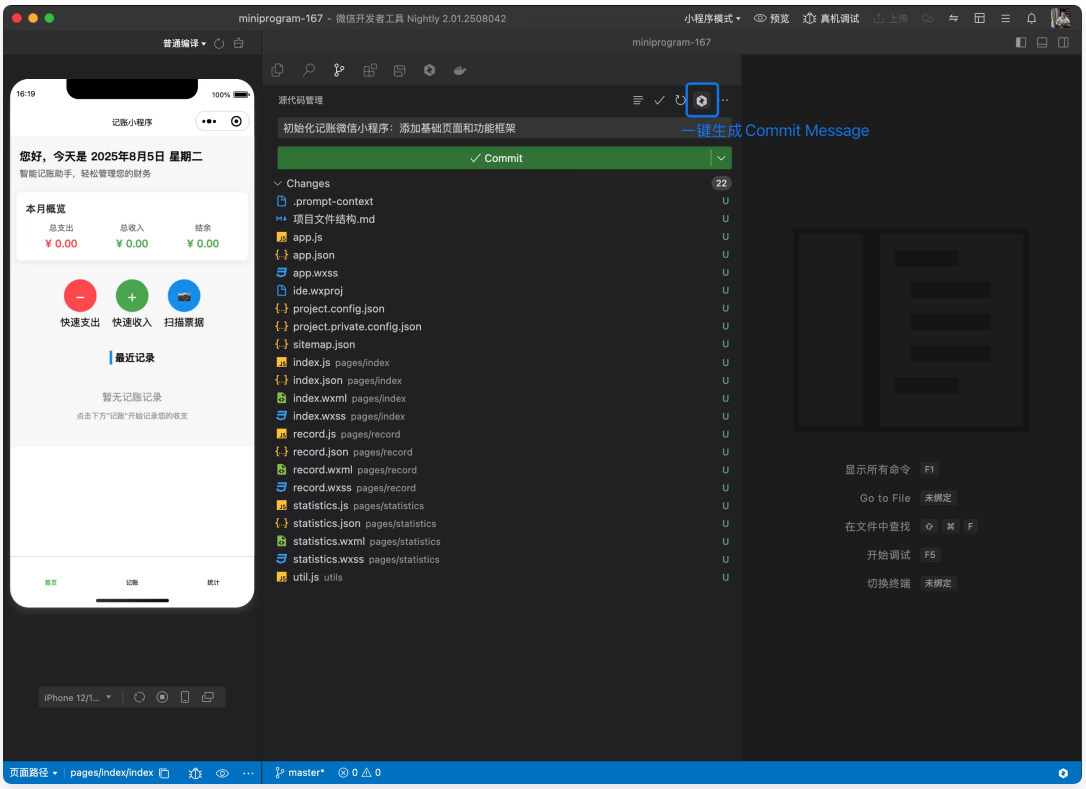


智能代码评审智能体

支持微信小程序代码批量评审，及时发现问题，并给出优化建议。



自动生成 commit message，规范开发流程。



快速安装

下面将介绍如何对新版本的微信开发者工具 IDE 和 CodeBuddy 进行安装和使用。

1. 前往 微信开发者平台 IDE 下载微信开发者工具，注意建议下载最新**开发版 Nightly Build**，请根据您的系统选择最新版本进行下载并安装。

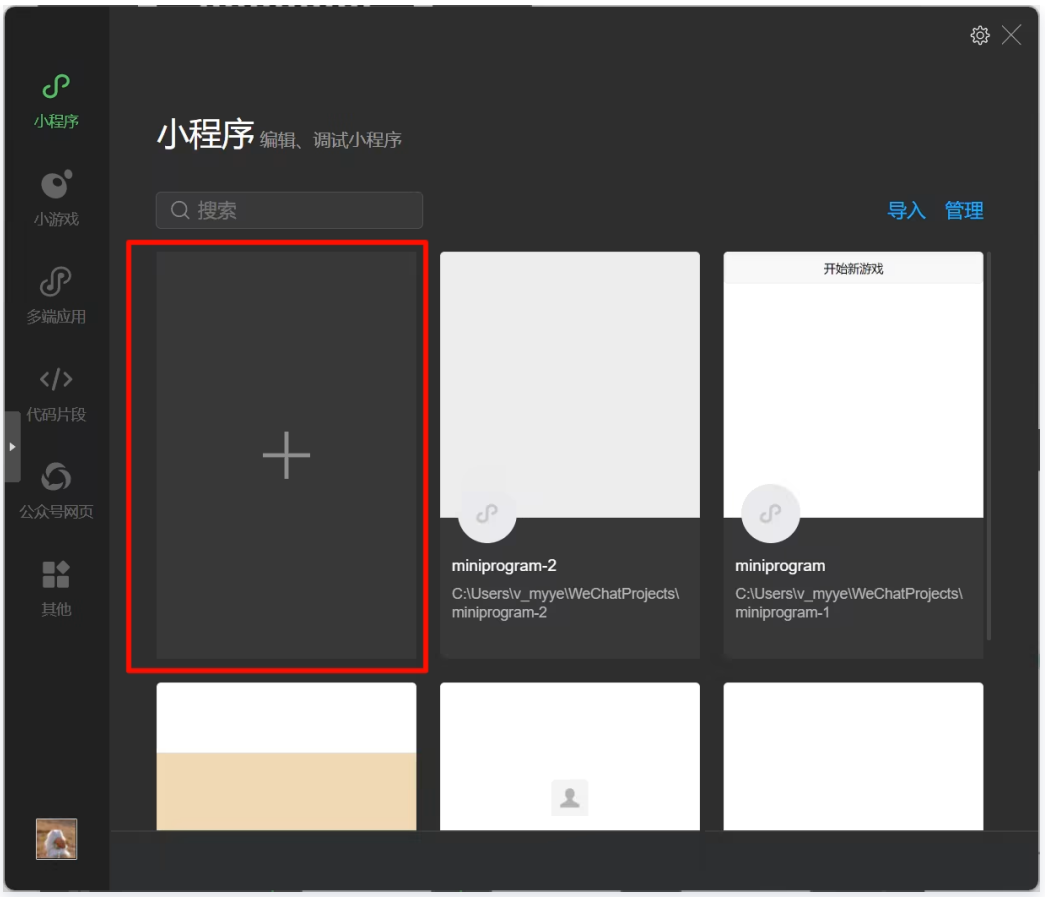


> **说明：**
> Stable Build 版本，暂不支持 MCP Servers 市场 。

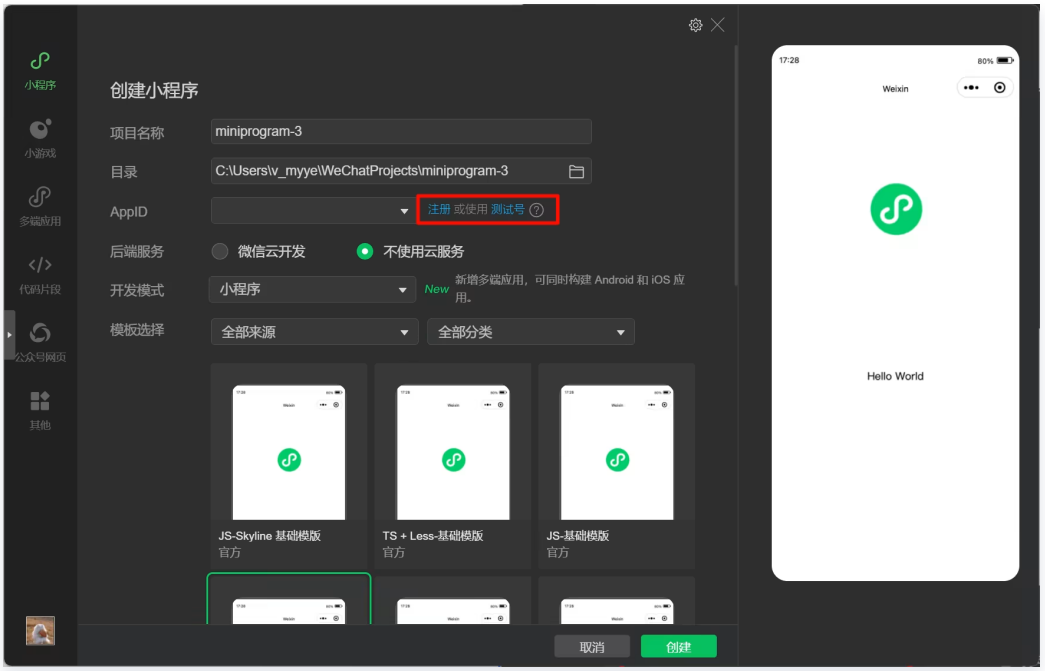
2. 安装完成后，打开微信开发者工具，使用微信扫码登录微信开发者工具。



3. 登录后，创建您所需的项目，或者选择您已创建的项目并打开。

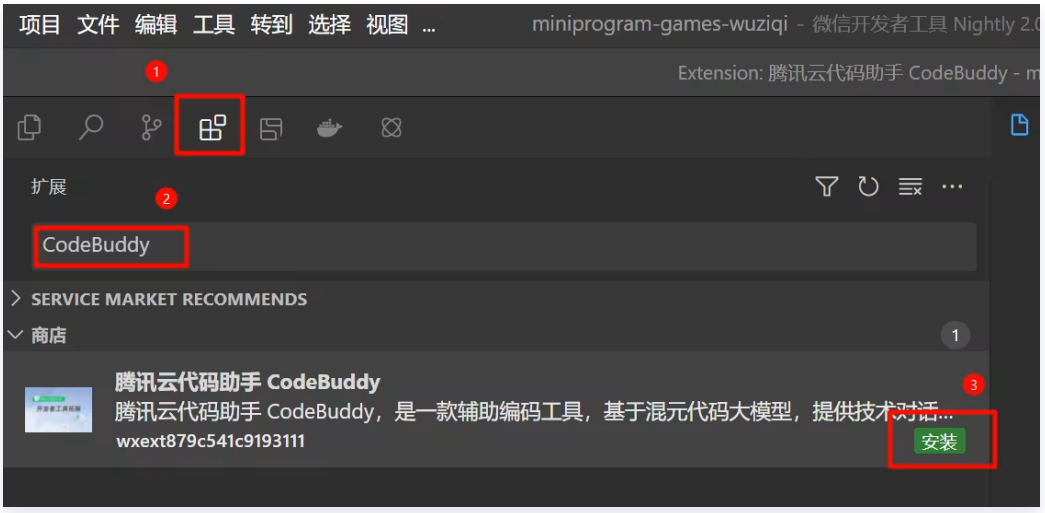


4. 如果选择创建项目，则根据您的实际需求进行创建。如果没有 AppID，则可以单击**注册**按钮进行注册小程序账号；或者直接单击**测试号**，使用测试号作为AppID，但使用测试号时目前暂不支持选择微信云开发。



> **注意：**
> 这里所创建的项目属于演示的项目，并非固定，在实际开发项目中需要根据自己的实际情况填写/选择信息，创建属于自己的项目。

5. 单击**扩展区域**图标，展开热门扩展或者在搜索栏中输入**腾讯云代码助手** 或 **CodeBuddy**，单击腾讯云代码助手右边的**安装**，等待安装完成即可。



> **说明：**
> 腾讯云代码助手插件安装完成后，建议重启一下微信开发者工具。如果出现安装异常，您可参考 不同端功能常见问题 进行解决。

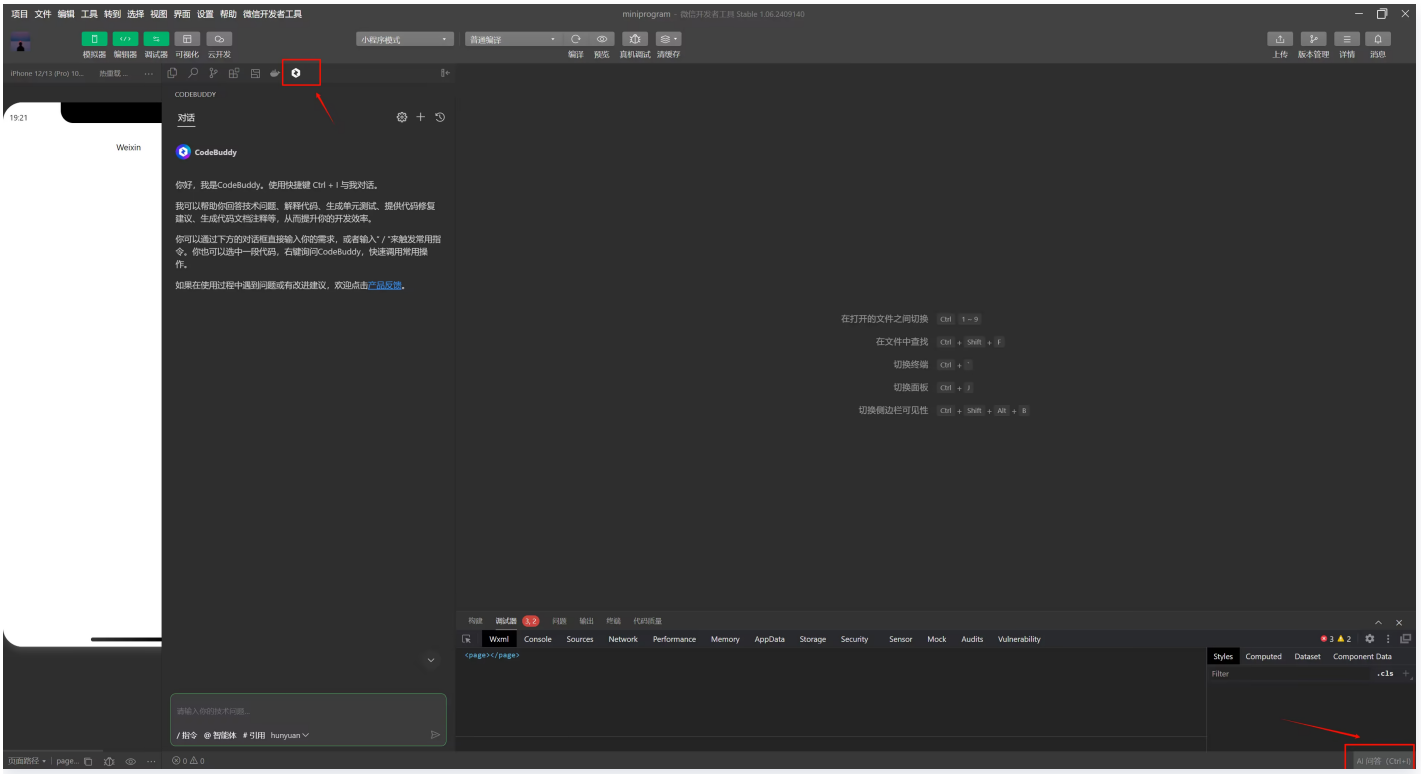
开始体验

安装完成 CodeBuddy 后即可直接进行体验，下面将以一个案例进行演示。
观看视频

技术对话功能

最近更新时间：2026-07-20 19:49:32

腾讯云代码助手的技术对话功能，与微信开发者 IDE 做了深度的融合，并使用和 VSCode 一致的交互操作。
我们在微信开发者平台 IDE 中，支持主流前端语言，小程序语言 Markdown、Json 等。
当成功登录微信开发者工具后，您只需要单击右下角开启 AI 对话或切换到代码助手对话面板即可。



输入框

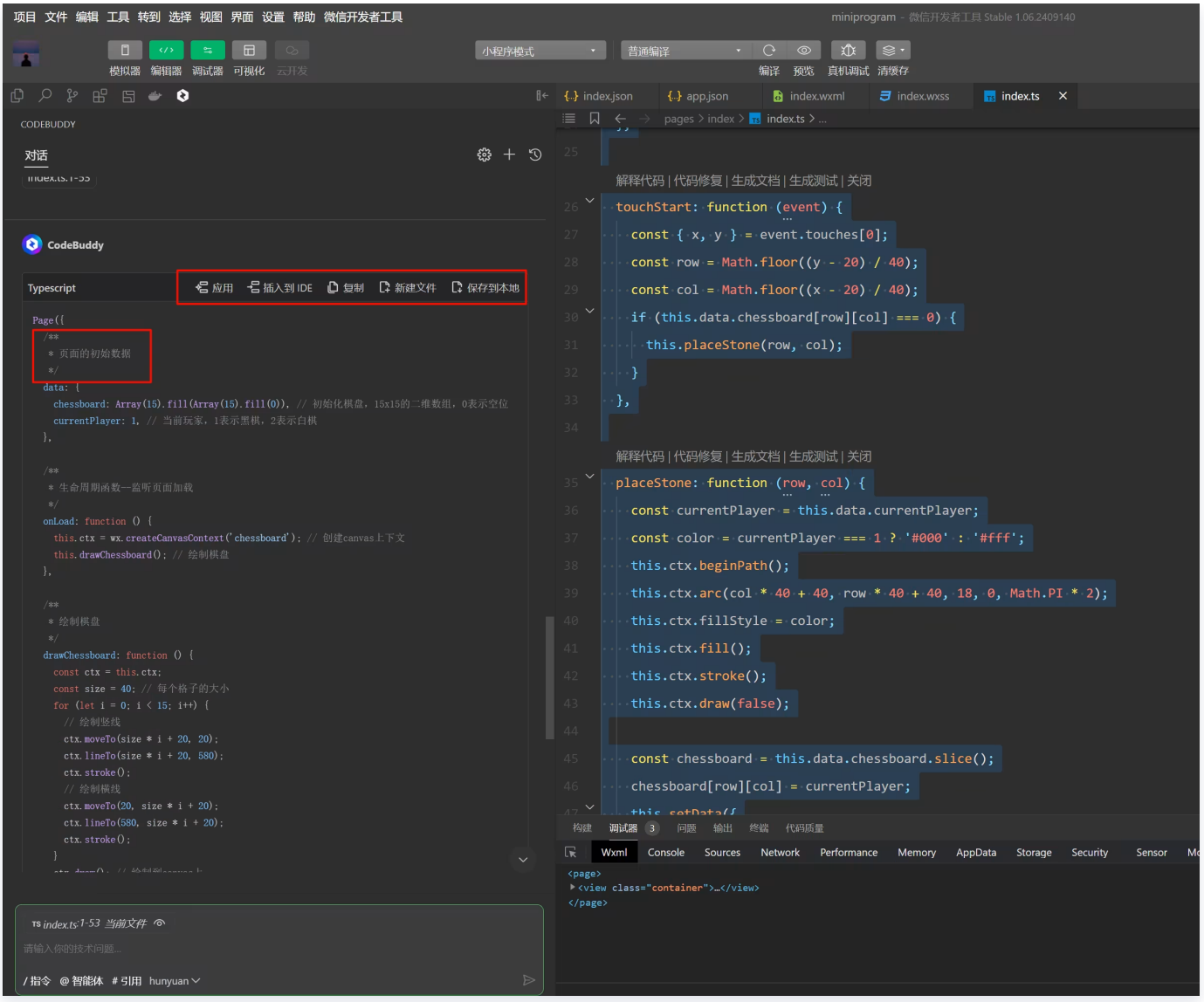
您可以直接在输入框中输入需求，然后回车，我们会根据当前项目和代码的上下文，生成更接近您要的答案。这里可以是技术回答，也可以是代码生成。
我们提供了指令、@、和 # 知识库能力。接下来详细介绍下各个模块的作用和使用方法。

指令

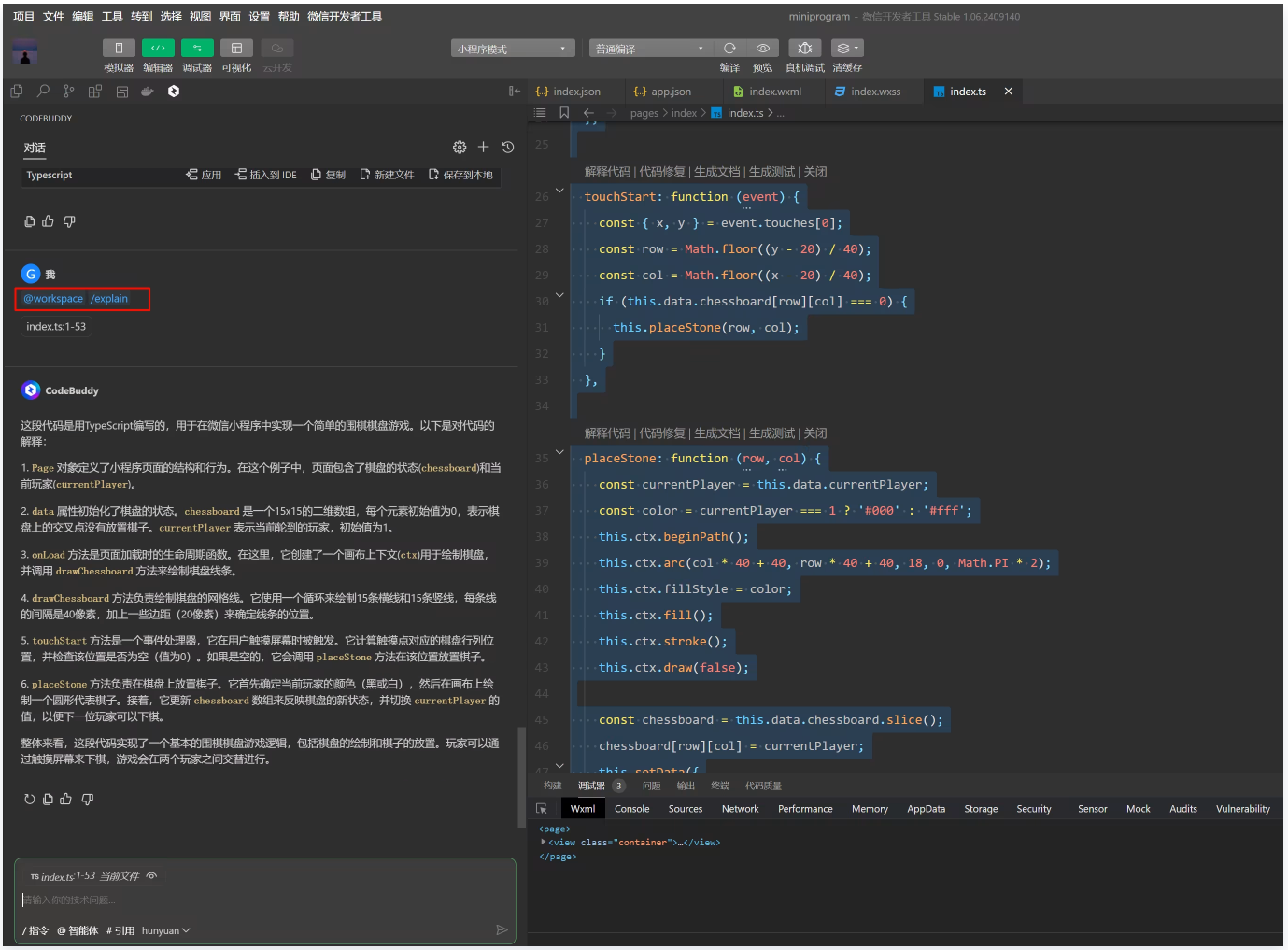
如上图所示，指令是帮助开发者给予一个提问前的**动作申明**，我们称之为指令，也是开发过程中最常用的场景。目前我们提供如下指令：

- /clear 清除当前会话内容。
- /comments 为圈选代码生成注释，如下图所示。

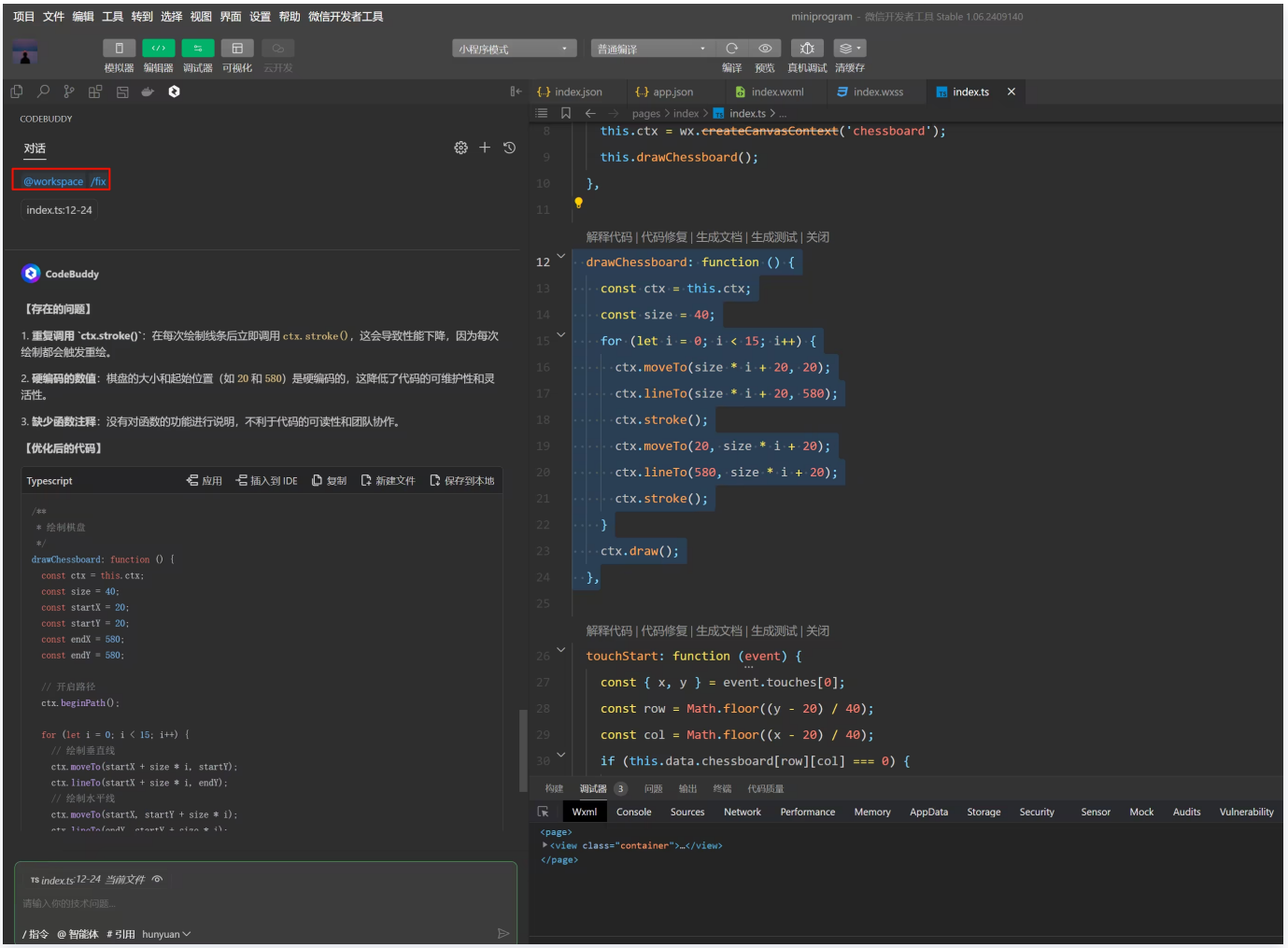
  圈选代码后，在对话框中输入 `/comments` 回车，代码助手会为您快速生成代码注释。单击顶部菜单，可以一键插入。



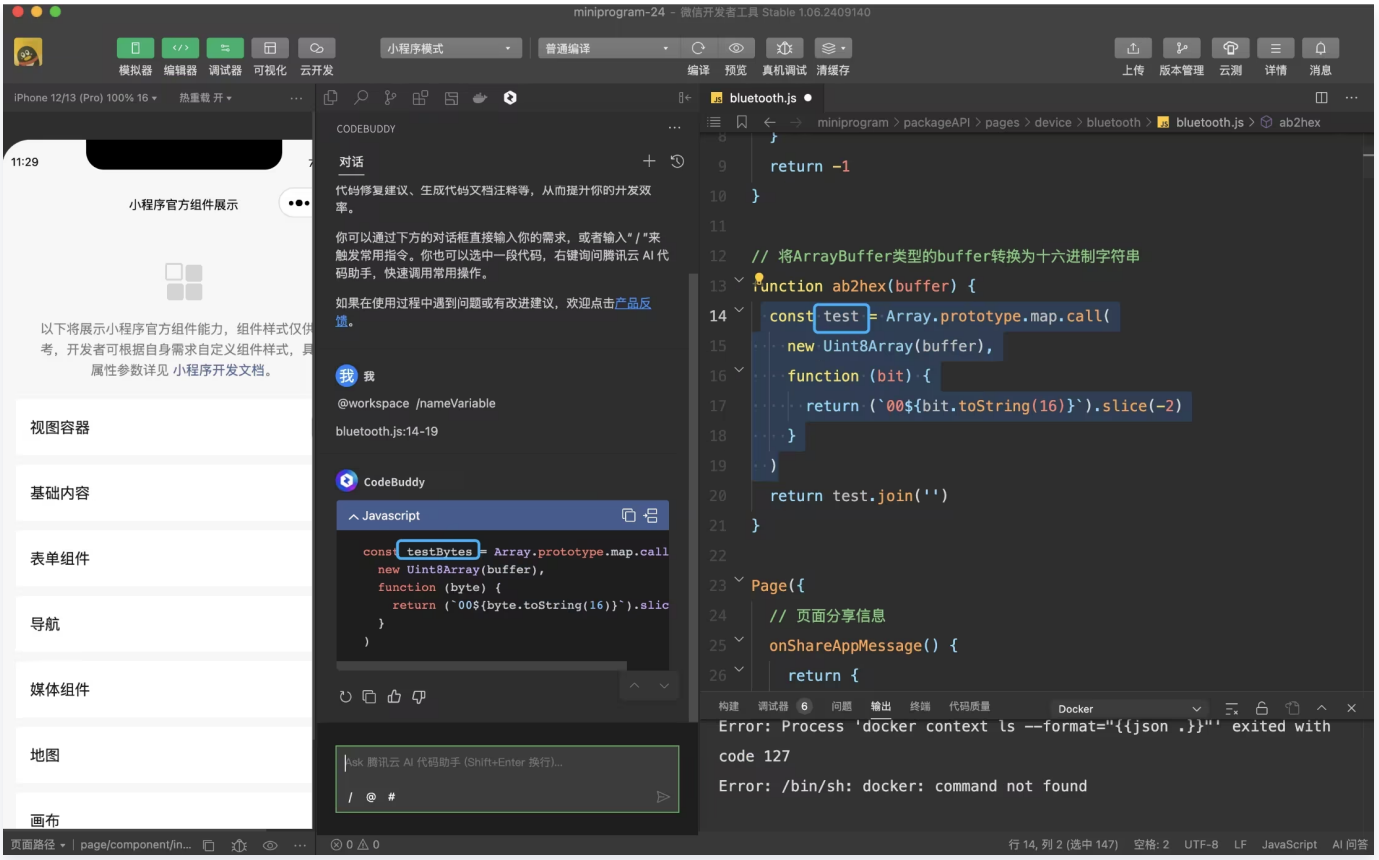
- /explain 为圈选代码生成解释。


项目 文件 编辑 工具 转到 选择 视图 界面 设置 帮助 微信开发者工具
miniprogram - 微信开发者工具 Stable 1.06.2409140
模拟器 编辑器 调试器 可视化 云开发
小程序模式
普通编译
编译 预览 真机调试 清缓存
CODEBUDDY
对话
Typescript
应用 插入到 IDE 复制 新建文件 保存到本地
我
@workspace /explain
index.ts:1-53
CodeBuddy
这段代码是用TypeScript编写的，用于在微信小程序中实现一个简单的围棋棋盘游戏。以下是对代码的解释：
1. Page 对象定义了小程序页面的结构和行为。在这个例子中，页面包含了棋盘的状态(chessboard)和当前玩家(currentPlayer)。
2. data 属性初始化了棋盘的状态。chessboard 是一个15x15的二维数组，每个元素初始值为0，表示棋盘上的交叉点没有放置棋子。currentPlayer 表示当前轮到的玩家，初始值为1。
3. onLoad 方法是页面加载时的生命周期函数。在这里，它创建了一个画布上下文(ctx)用于绘制棋盘，并调用 drawChessboard 方法来绘制棋盘线条。
4. drawChessboard 方法负责绘制棋盘的网格线。它使用一个循环来绘制15条横线和15条竖线，每条线的间隔是40像素，加上一些边距（20像素）来确定线条的位置。
5. touchStart 方法是一个事件处理器，它在用户触摸屏幕时被触发。它计算触摸点对应的棋盘行列位置，并检查该位置是否为空（值为0）。如果是空的，它会调用 placeStone 方法在该位置放置棋子。
6. placeStone 方法负责在棋盘上放置棋子。它首先确定当前玩家的颜色（黑或白），然后在画布上绘制一个圆形代表棋子。接着，它更新 chessboard 数组来反映棋盘的新状态，并切换 currentPlayer 的值，以便下一位玩家可以下棋。
整体来看，这段代码实现了一个基本的围棋棋盘游戏逻辑，包括棋盘的绘制和棋子的放置。玩家可以通过触摸屏幕来下棋，游戏会在两个玩家之间交替进行。
index.ts:1-53 当前文件
/ 指令 @ 智能体 # 引用 hunyuan
index.json app.json index.wxml index.wxss index.ts
pages > index > index.ts > ...
解释代码 | 代码修复 | 生成文档 | 生成测试 | 关闭
touchStart: function (event) {
const { x, y } = event.touches[0];
const row = Math.floor((y - 20) / 40);
const col = Math.floor((x - 20) / 40);
if (this.data.chessboard[row][col] === 0) {
this.placeStone(row, col);
}
},
解释代码 | 代码修复 | 生成文档 | 生成测试 | 关闭
placeStone: function (row, col) {
const currentPlayer = this.data.currentPlayer;
const color = currentPlayer === 1 ? '#000' : '#fff';
this.ctx.beginPath();
this.ctx.arc(col * 40 + 40, row * 40 + 40, 18, 0, Math.PI * 2);
this.ctx.fillStyle = color;
this.ctx.fill();
this.ctx.stroke();
this.ctx.draw(false);
const chessboard = this.data.chessboard.slice();
chessboard[row][col] = currentPlayer;
构建 调试器 3 问题 输出 终端 代码质量
Wxml Console Sources Network Performance Memory AppData Storage Security Sensor Mock Audits Vulnerability
<page>
<view class="container">…</view>
</page>

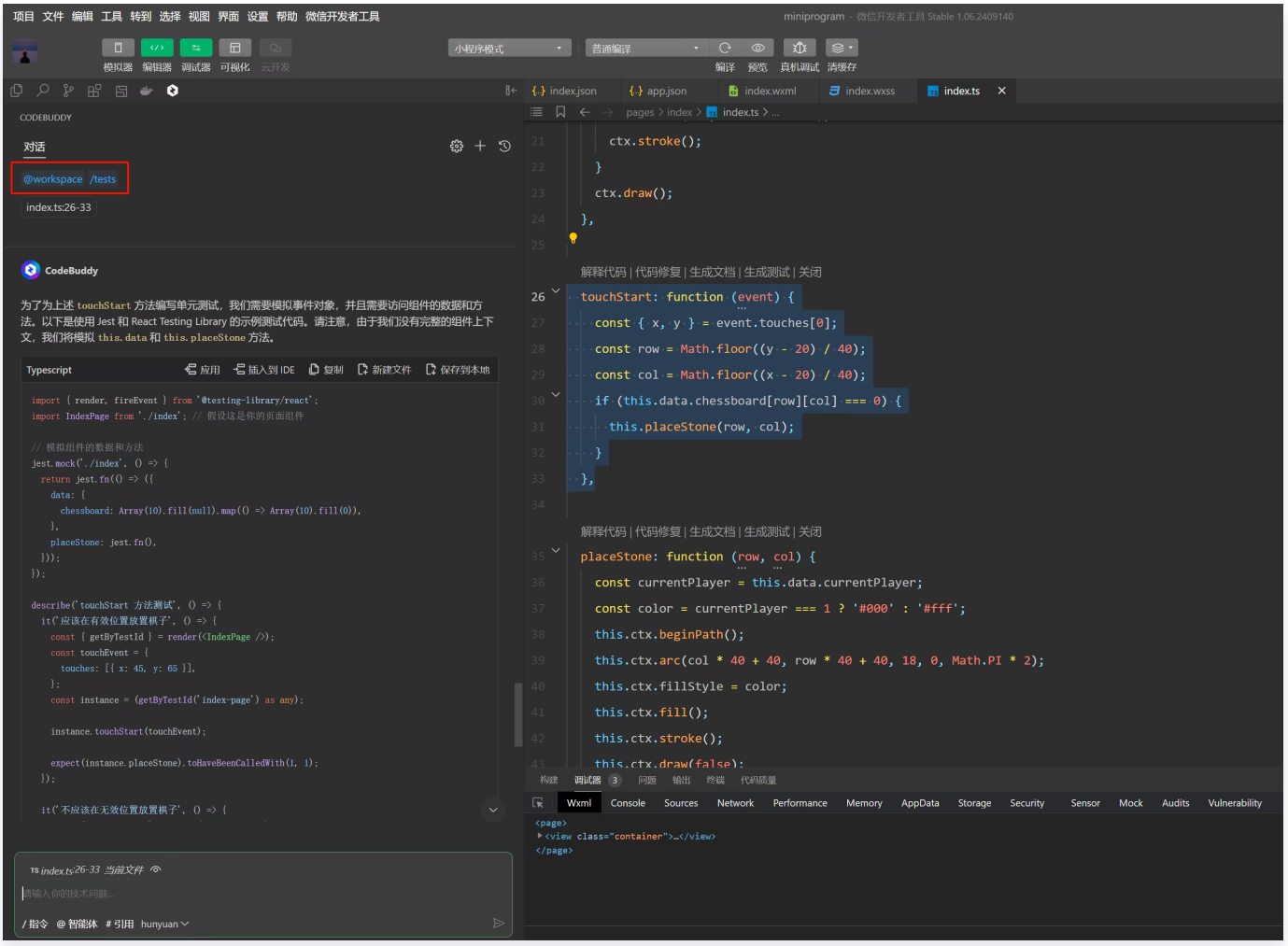
- /fix 修复圈选代码。



- /nameVariables AI 生成变量名。



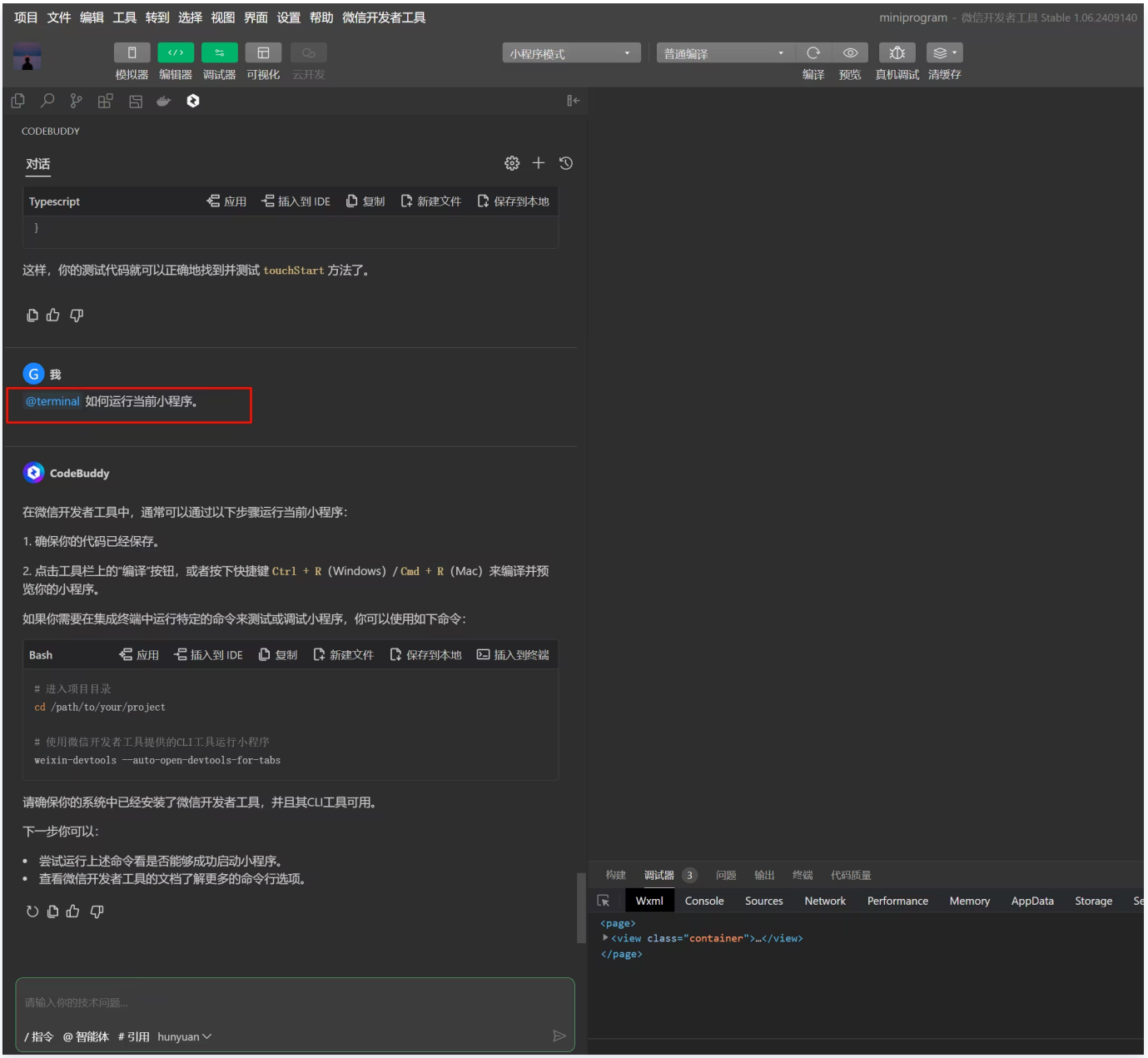
- /tests 为圈选代码生成对应的测试。



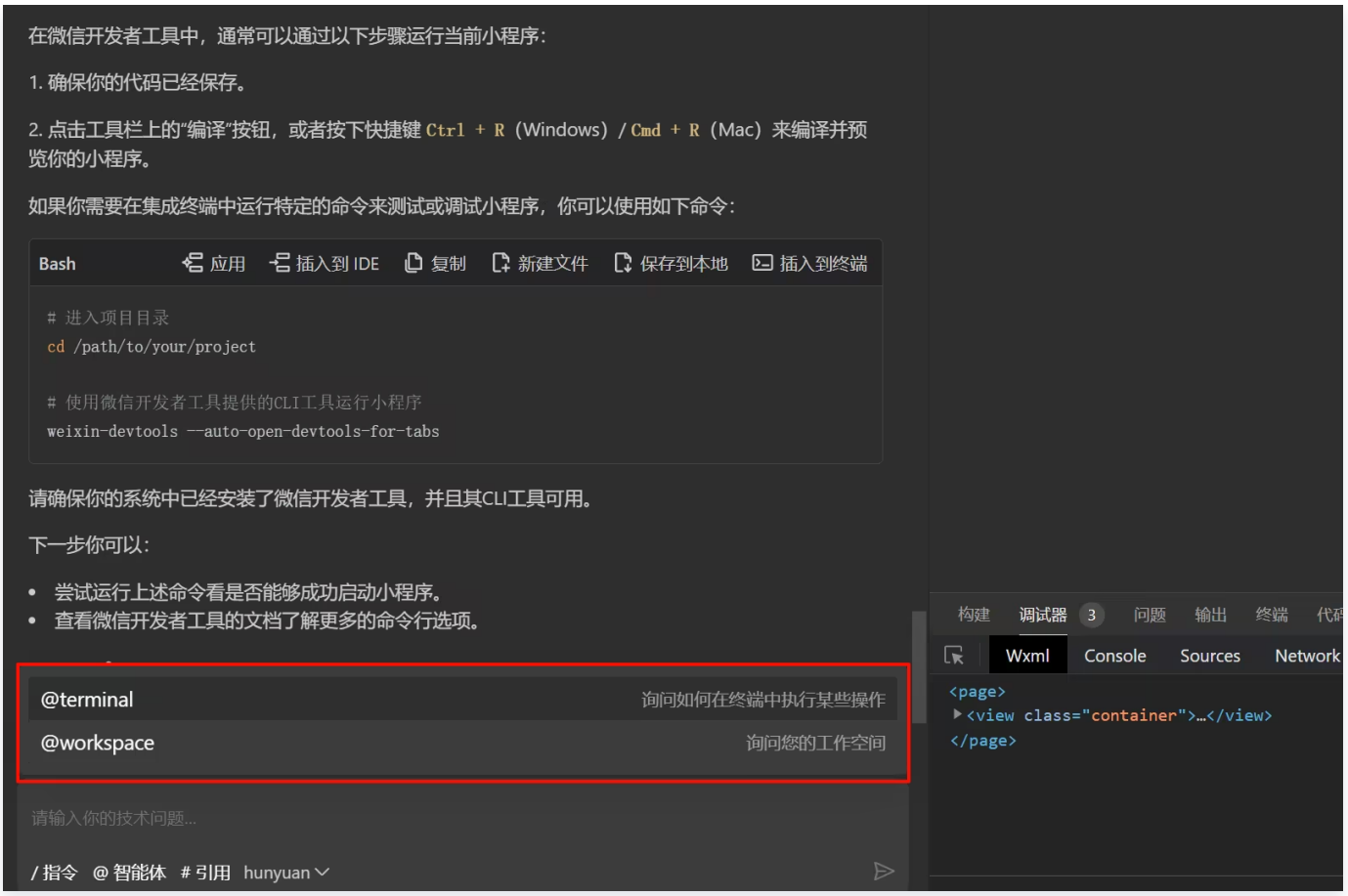
@ Agent 能力

您可以在对话中 @ 不同需求的 AI 智能体。我们目前支持：

- **@workspace： 对当前工程询问的智能体。**
- **@terminal：对终端问题询问解答的智能体，如下图：**

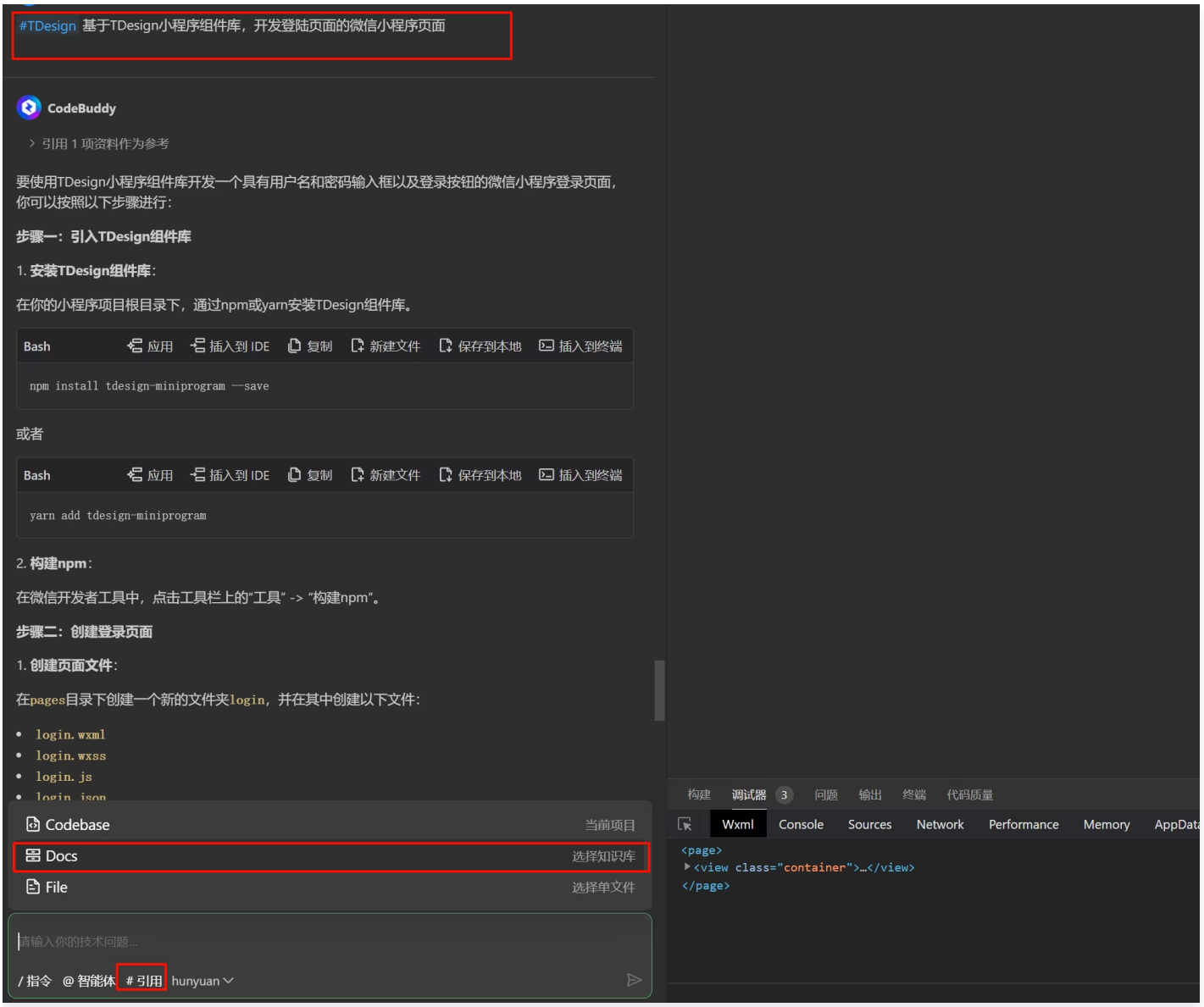

项目 文件 编辑 工具 转到 选择 视图 界面 设置 帮助 微信开发者工具
miniprogram - 微信开发者工具 Stable 1.06.2409140
模拟器 编辑器 调试器 可视化 云开发
小程序模式
普通编译
编译 预览 真机调试 清缓存
CODEBUDDY
对话
Typescript 应用 插入到 IDE 复制 新建文件 保存到本地
}
这样，你的测试代码就可以正确地找到并测试 touchStart 方法了。
我
@terminal 如何运行当前小程序。
CodeBuddy
在微信开发者工具中，通常可以通过以下步骤运行当前小程序：
1. 确保你的代码已经保存。
2. 点击工具栏上的"编译"按钮，或者按下快捷键 Ctrl + R（Windows）/ Cmd + R（Mac）来编译并预览你的小程序。
如果你需要在集成终端中运行特定的命令来测试或调试小程序，你可以使用如下命令：
Bash 应用 插入到 IDE 复制 新建文件 保存到本地 插入到终端
进入项目目录
cd /path/to/your/project
使用微信开发者工具提供的CLI工具运行小程序
weixin-devtools --auto-open-devtools-for-tabs
请确保你的系统中已经安装了微信开发者工具，并且其CLI工具可用。
下一步你可以：
• 尝试运行上述命令看是否能够成功启动小程序。
• 查看微信开发者工具的文档了解更多的命令行选项。
请输入你的技术问题...
/ 指令 @ 智能体 # 引用 hunyuan
构建 调试器 3 问题 输出 终端 代码质量
Wxml Console Sources Network Performance Memory AppData Storage
<page>
<view class="container">…</view>
</page>

命令展示：



上下文

您可以选择知识库，引入更多上下文可以加强对话生成内容。如下图所示，在输入框输入 #TDesign 您的需求后，知识库会先找到 TDesign 小程序相关的上下文代码，并作为额外上下文，结合您的提问，生成对话结果。同时在对话中会列举几个最相关的召回的文档地址供用户查阅。

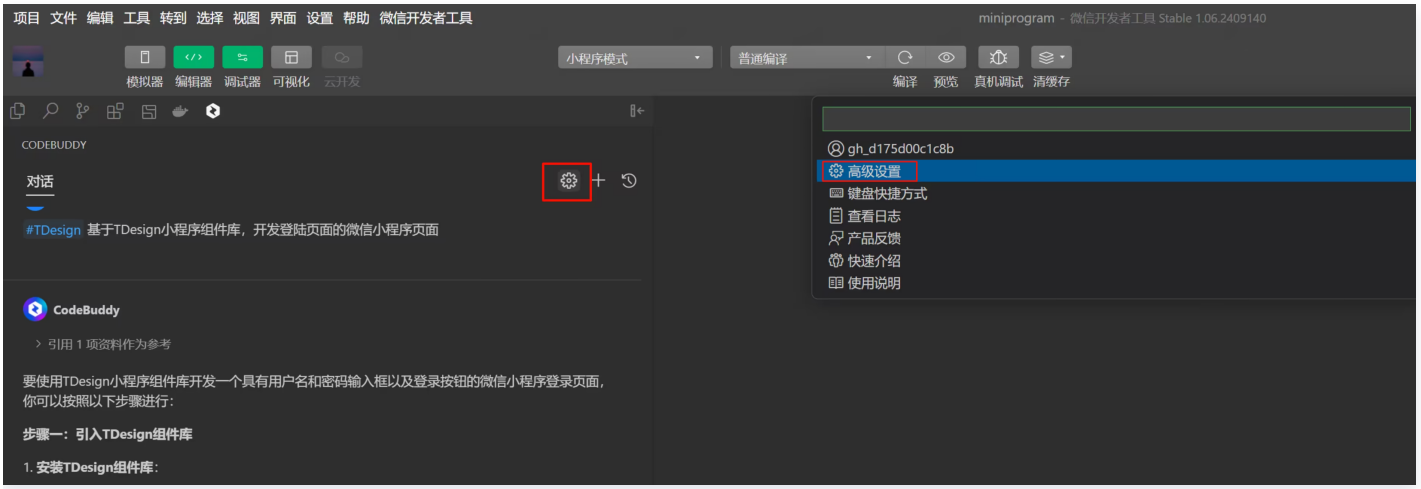

#TDesign 基于TDesign小程序组件库，开发登陆页面的微信小程序页面
CodeBuddy
引用 1 项资料作为参考
要使用TDesign小程序组件库开发一个具有用户名和密码输入框以及登录按钮的微信小程序登录页面，你可以按照以下步骤进行：
步骤一：引入TDesign组件库
1. 安装TDesign组件库：
在你的小程序项目根目录下，通过npm或yarn安装TDesign组件库。
Bash 应用 插入到 IDE 复制 新建文件 保存到本地 插入到终端
npm install tdesign-miniprogram --save
或者
Bash 应用 插入到 IDE 复制 新建文件 保存到本地 插入到终端
yarn add tdesign-miniprogram
2. 构建npm：
在微信开发者工具中，点击工具栏上的"工具" -> "构建npm"。
步骤二：创建登录页面
1. 创建页面文件：
在pages目录下创建一个新的文件夹login，并在其中创建以下文件：
login.wxml
login.wxss
login.js
login.json
Codebase 当前项目
Docs 选择知识库
File 选择单文件
请输入你的技术问题...
/ 指令 @ 智能体 # 引用 hunyuan
构建 调试器 3 问题 输出 终端 代码质量
Wxml Console Sources Network Performance Memory AppData
<page>
<view class="container">…</view>
</page>

高级设置

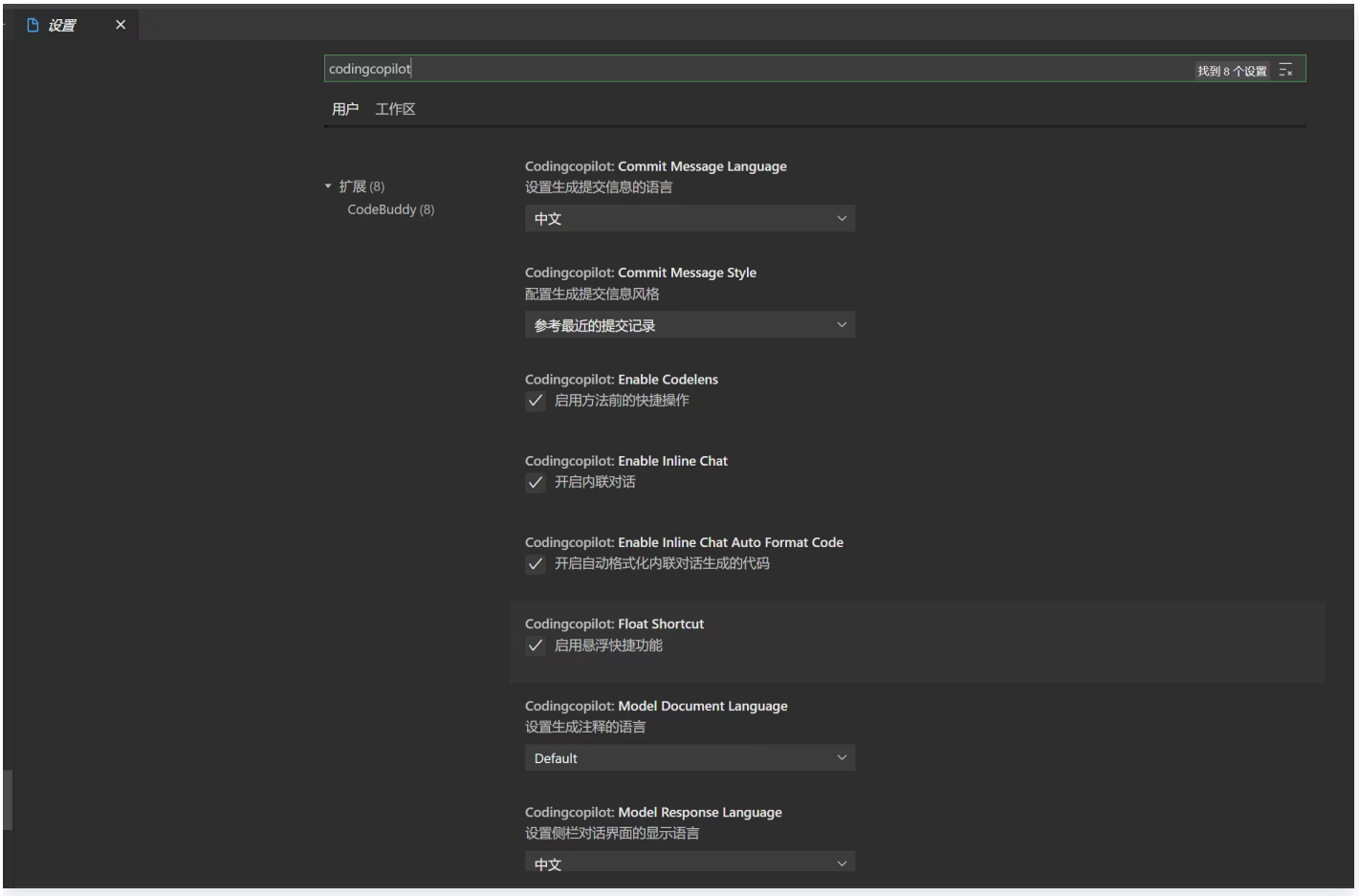
最近更新时间：2026-07-16 21:20:01

我们提供设置入口，方便开发者可以按自己的喜好配置。

单击对话框面板的设置按钮，单击高级设置进入。



目前提供如下设置，您可根据自己的需求进行选择设置。



版权所有：腾讯云计算（北京）有限责任公司

版权所有：腾讯云计算（北京）有限责任公司

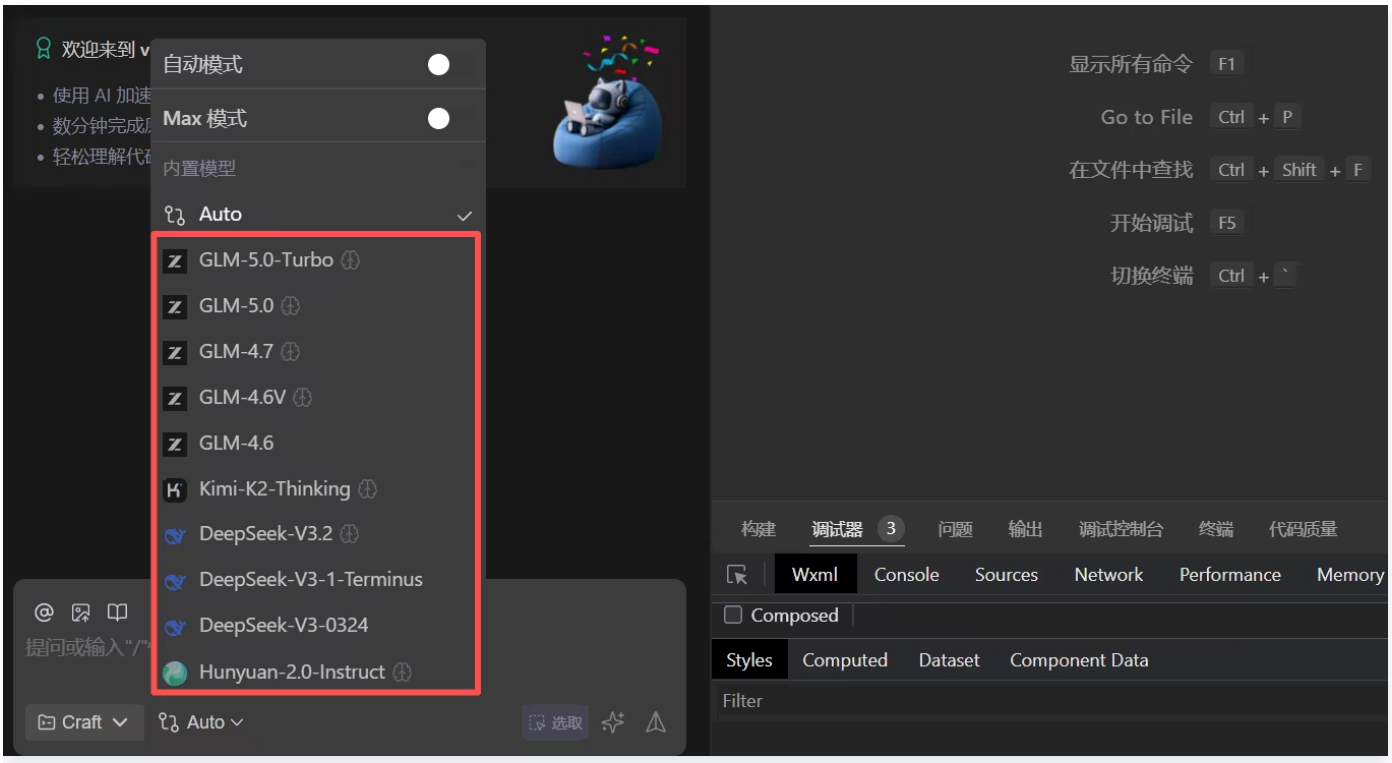
对话模型接入和切换

最近更新时间：2026-07-20 19:49:32

在微信小程序开发者工具 IDE 中，腾讯云代码助手已经支持对话大模型的接入和切换，对话大模型包括腾讯部署的混元大模型、DeepSeek 大模型和 GLM 大模型，以及其他可自定义接入的大模型。

腾讯部署的对话模型

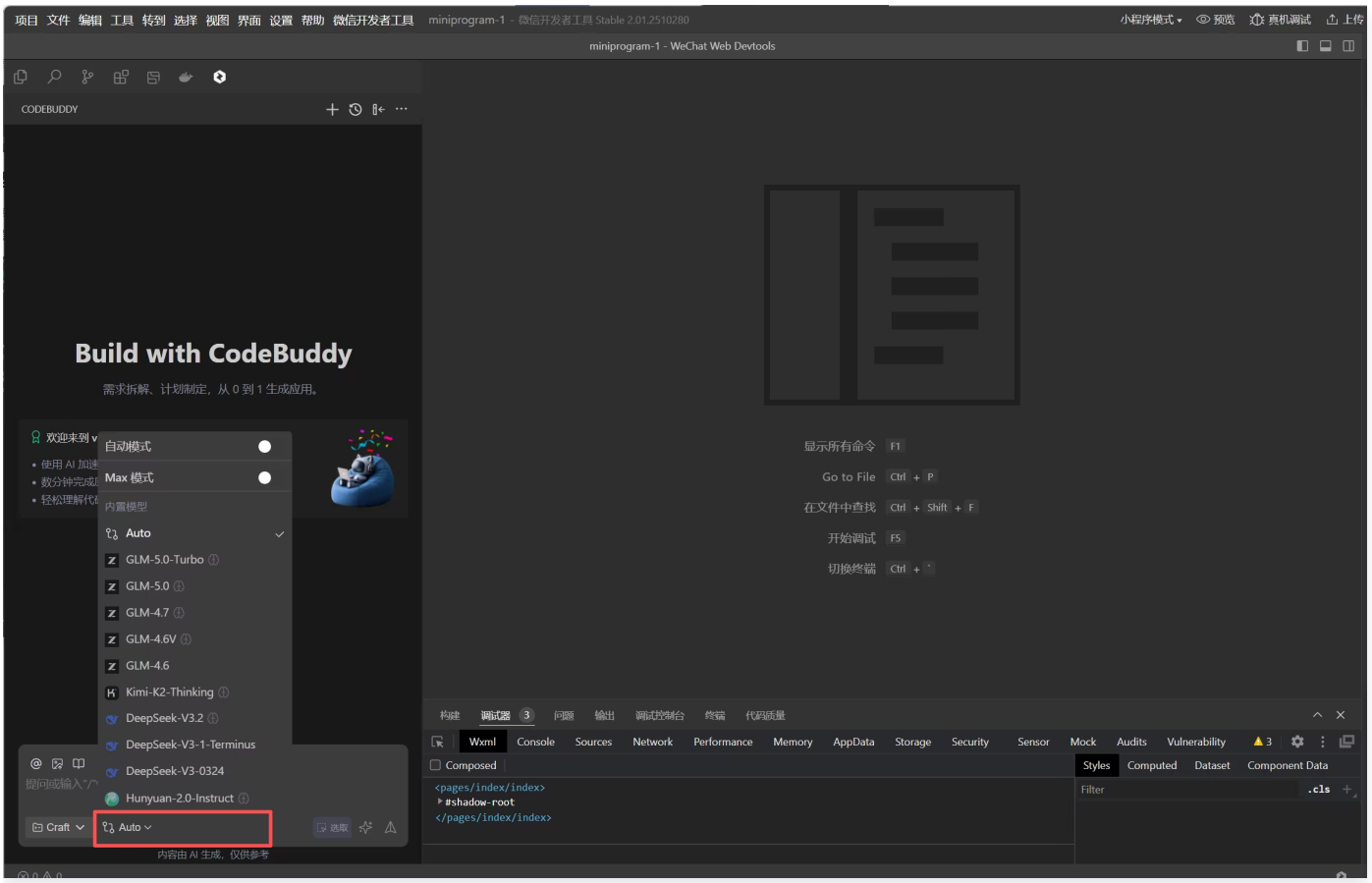
腾讯部署的模型已经内置到腾讯云代码助手中，包含腾讯自研的混元大模型（Hunyuan），DeepSeek-V3 模型、智谱大模型和 Kimi-K2-Thinking 大模型。



对话模型切换

腾讯云代码助手侧边栏打开技术对话功能，在对话框模型选项（默认是 Auto 模式）中，包含了腾讯部署的官方大模型和自定义接入配置的大模型。您可以自由切换对话模型，帮助提升您的编码体验。

版权所有：腾讯云计算（北京）有限责任公司


项目 文件 编辑 工具 转到 选择 视图 界面 设置 帮助 微信开发者工具
miniprogram-1 - 微信开发者工具 Stable 2.01.2510280
小程序模式
预览
真机调试
上传
miniprogram-1 - WeChat Web Devtools
CODEBUDDY
Build with CodeBuddy
需求拆解、计划制定，从 0 到 1 生成应用。
自动模式
Max 模式
内置模型
Auto
GLM-5.0-Turbo
GLM-5.0
GLM-4.7
GLM-4.6V
GLM-4.6
Kimi-K2-Thinking
DeepSeek-V3.2
DeepSeek-V3-1-Terminus
DeepSeek-V3-0324
Hunyuan-2.0-Instruct
Craft
Auto
内容由 AI 生成，仅供参考
显示所有命令 F1
Go to File Ctrl + P
在文件中查找 Ctrl + Shift + F
开始调试 F5
切换终端 Ctrl + `
构建
调试器 3
问题
输出
调试控制台
终端
代码质量
Wxml
Console
Sources
Network
Performance
Memory
AppData
Storage
Security
Sensor
Mock
Audits
Vulnerability
Composed
Styles
Computed
Dataset
Component Data
Filter
.cls
<pages/index/index>
#shadow-root
</pages/index/index>

Visual Studio 接入指南

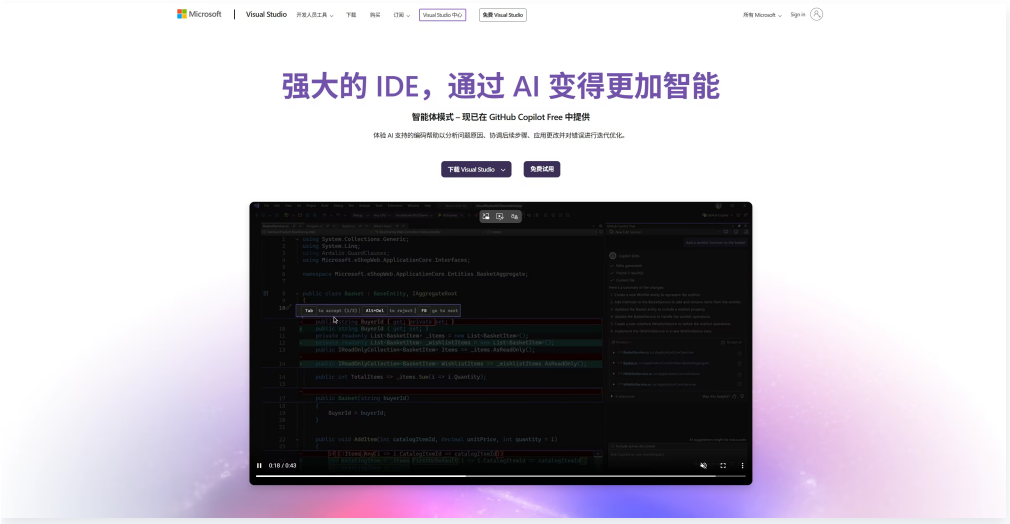
最近更新时间：2026-07-20 19:49:32

Visual Studio 本身是一个功能丰富的集成开发环境，支持多种编程语言和平台，包括 C#、C++、TypeScript 等。目前，腾讯云代码助手插件已经在 Visual Studio 市场上架，使用混元代码大模型，为开发者提供了强大的智能辅助编码能力。接下来，看看腾讯云代码助手在 Visual Studio 中如何进行接入。

Visual Studio

前往 Visual Studio 官网 下载。

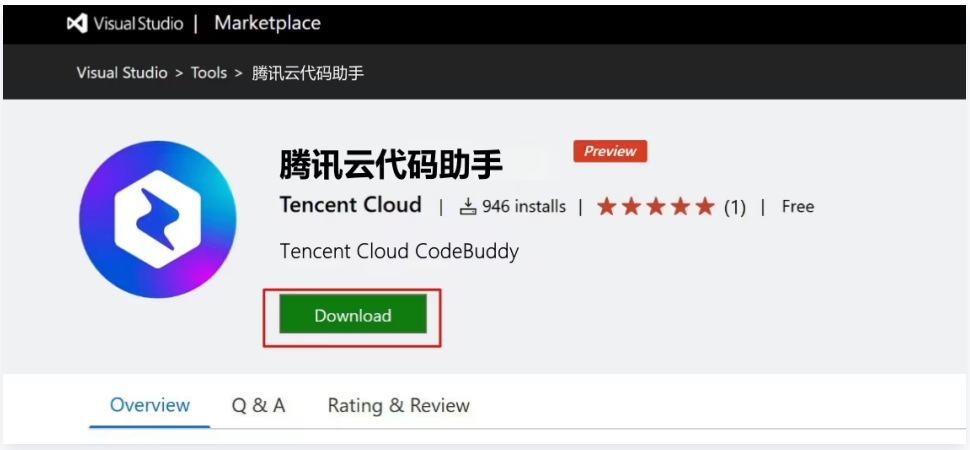
> **说明：**
> Visual Studio 最低版本要求17.6及以上，版本号可在顶部菜单栏中单击**帮助** > **检查更新**查看。



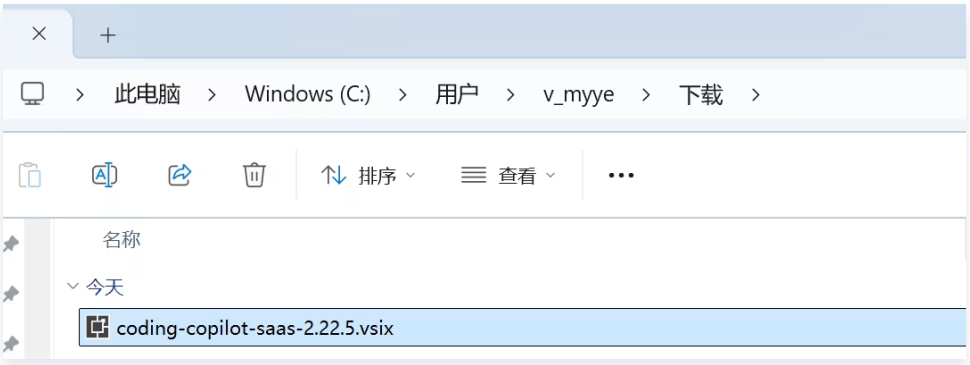
安装插件

方式一：手动下载安装包并安装

1. 单击 Visual Studio 市场 下载插件。

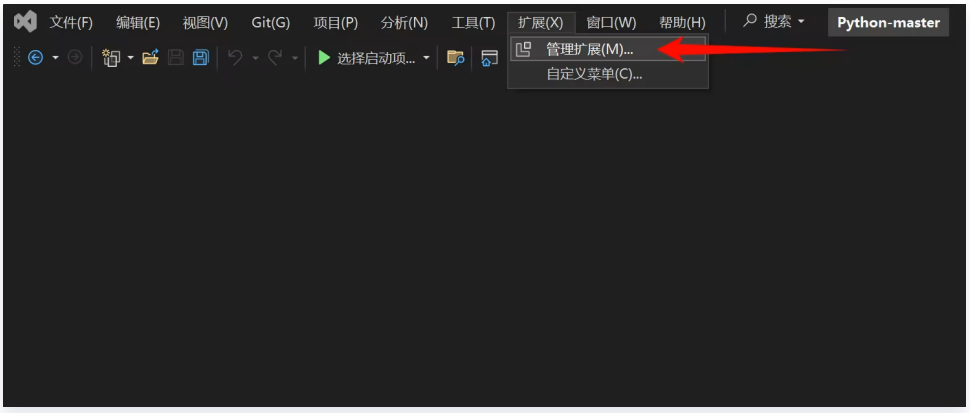


2. 下载完后双击插件安装程序进行安装。

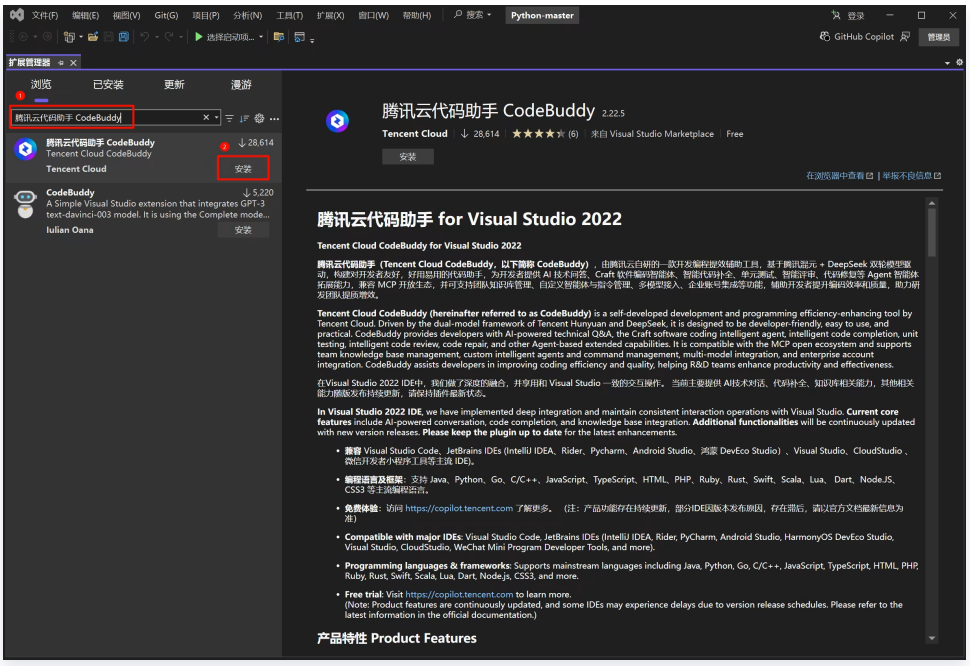


方式二：Visual Studio 内的插件市场安装

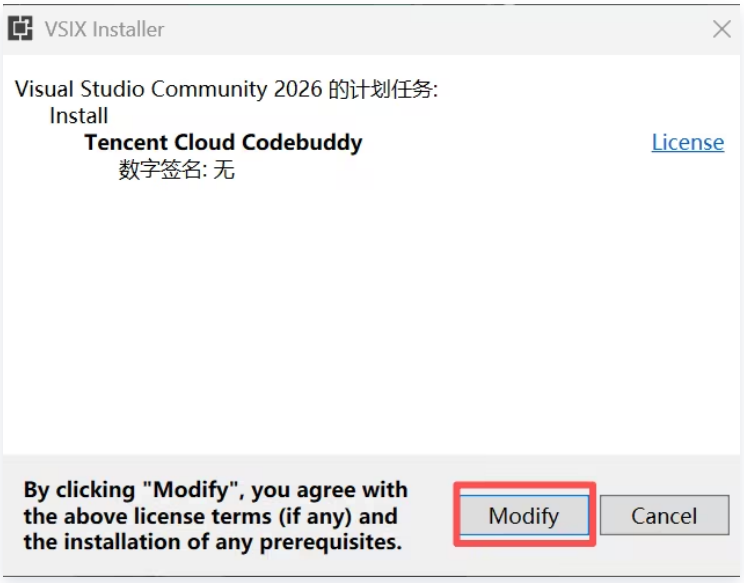
1. 在顶部菜单栏中单击**扩展 > 管理扩展**。



2. 搜索**腾讯云代码助手 CodeBuddy**，然后单击**安装**。



3. 关闭 **Visual Studio** ，在弹出的 **VSIX Installer** 窗口中单击 **Modify** 执行下载。



4. 安装成功如下图所示：



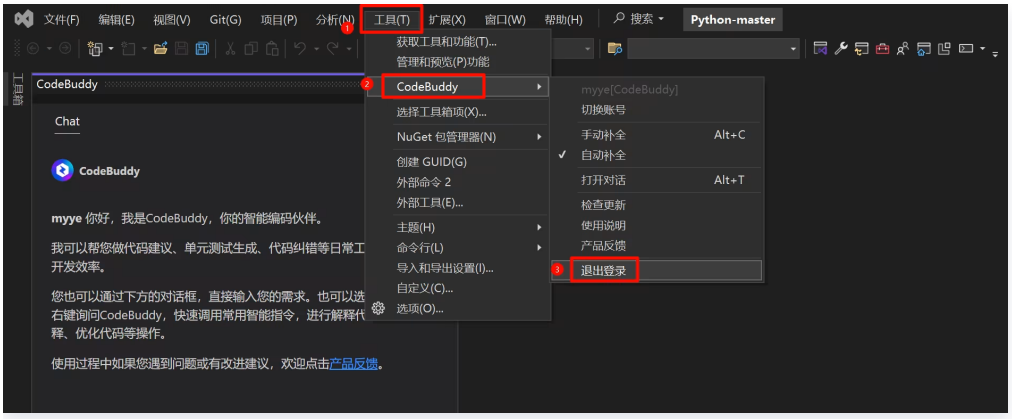
登录插件

打开安装好的 Visual Studio，登录插件。

观看视频

退出登录

您可以在菜单栏中依次选择工具 > CodeBuddy > 退出登录，即可退出插件登录。



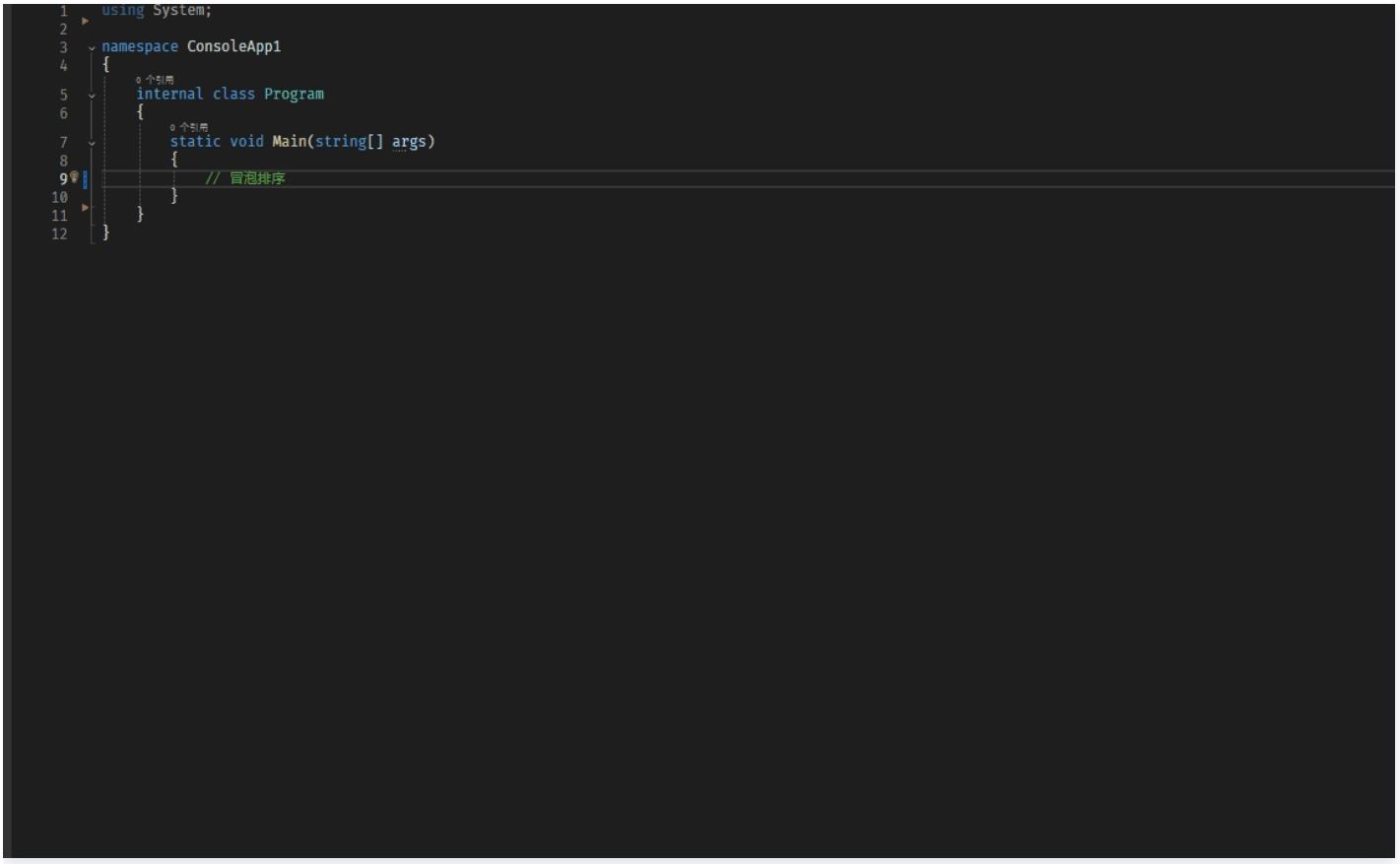
使用指南

最近更新时间：2026-07-20 19:49:32

代码补全

在 Visual Studio 代码编辑区内，可以通过正常编码停顿后自动触发补全场景，并会根据上下文智能生成准确的代码补全推荐。该场景具有如下特性：

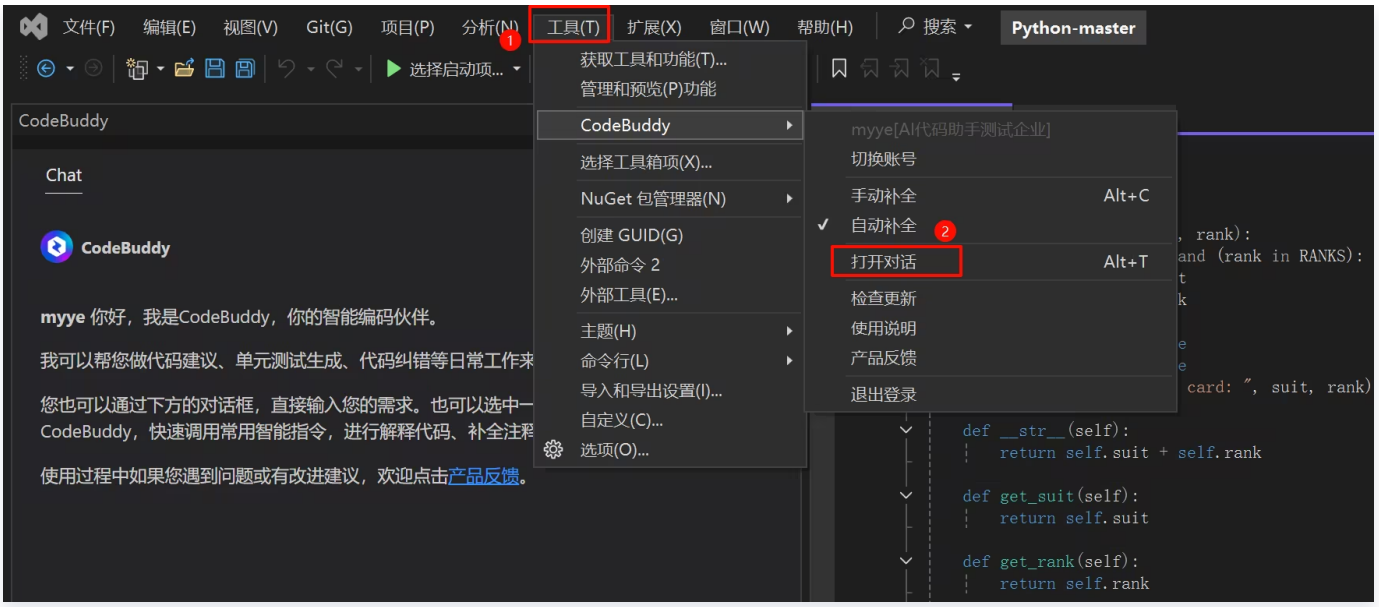
- 支持多种编程语言的语法深度学习，提供全面的代码补全功能。
- 注释描述、变量、函数等补全一应俱全。
- 根据注释或函数声明智能生成代码块。
- 函数间补全，帮助您轻松实现新函数或完善现有函数。



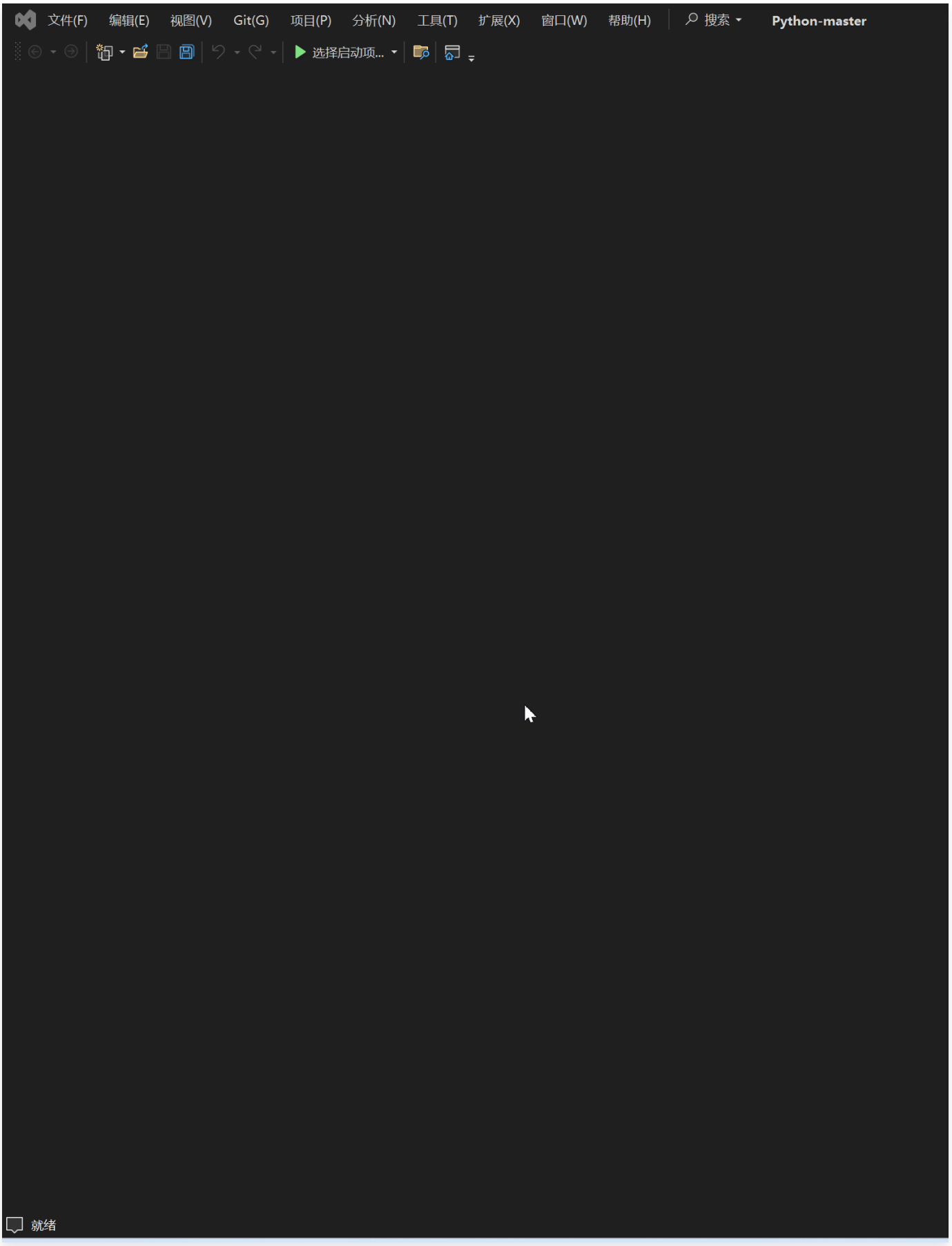
技术对话

插件提供了多种语言和为各种框架提供了建议，在 Visual Studio 上支持 C#、C++、TypeScript、ASP.NET 等方面的回答。

1. 登录后，可以通过**工具** > **CodeBuddy** > **打开对话**进行技术对话。

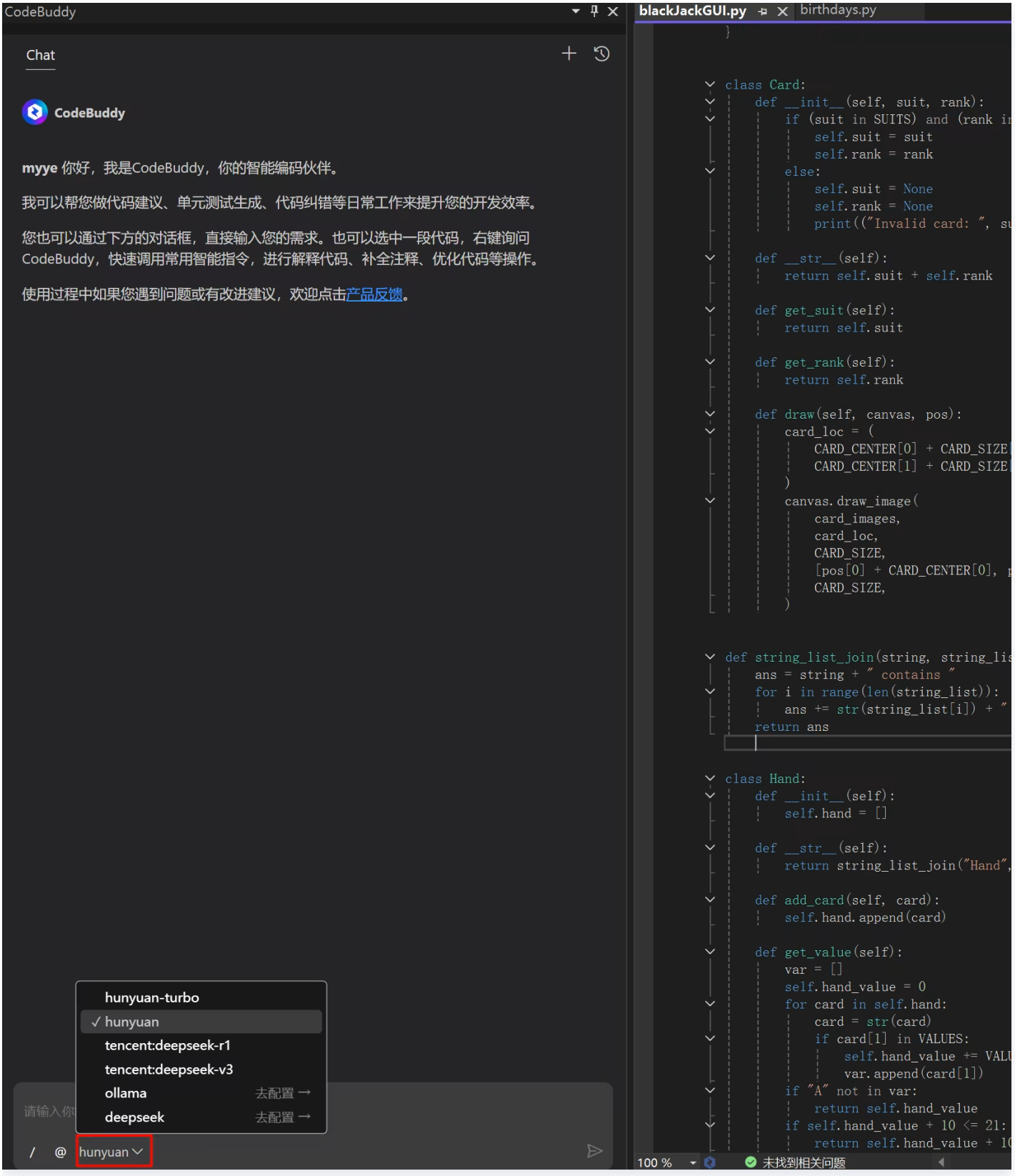


2. 在侧栏对话框，可以咨询技术问题，例如使用 C# 写一段冒泡排序。


文件(F)
编辑(E)
视图(V)
Git(G)
项目(P)
分析(N)
工具(T)
扩展(X)
窗口(W)
帮助(H)
搜索
Python-master
选择启动项...
就绪

对话模型切换

腾讯云代码助手除了内置腾讯自研的混元大模型之外，目前已经部署了官方 DeepSeek-R1 模型，默认是混元大模型。在左侧对话面板中，有一个模型选项可以支持您自由选择对话模型进行切换。



快捷指令问答

通过输入 / 或 @ 调用预置的快捷指令，或通过 # 引用知识库，快速获取所需帮助：

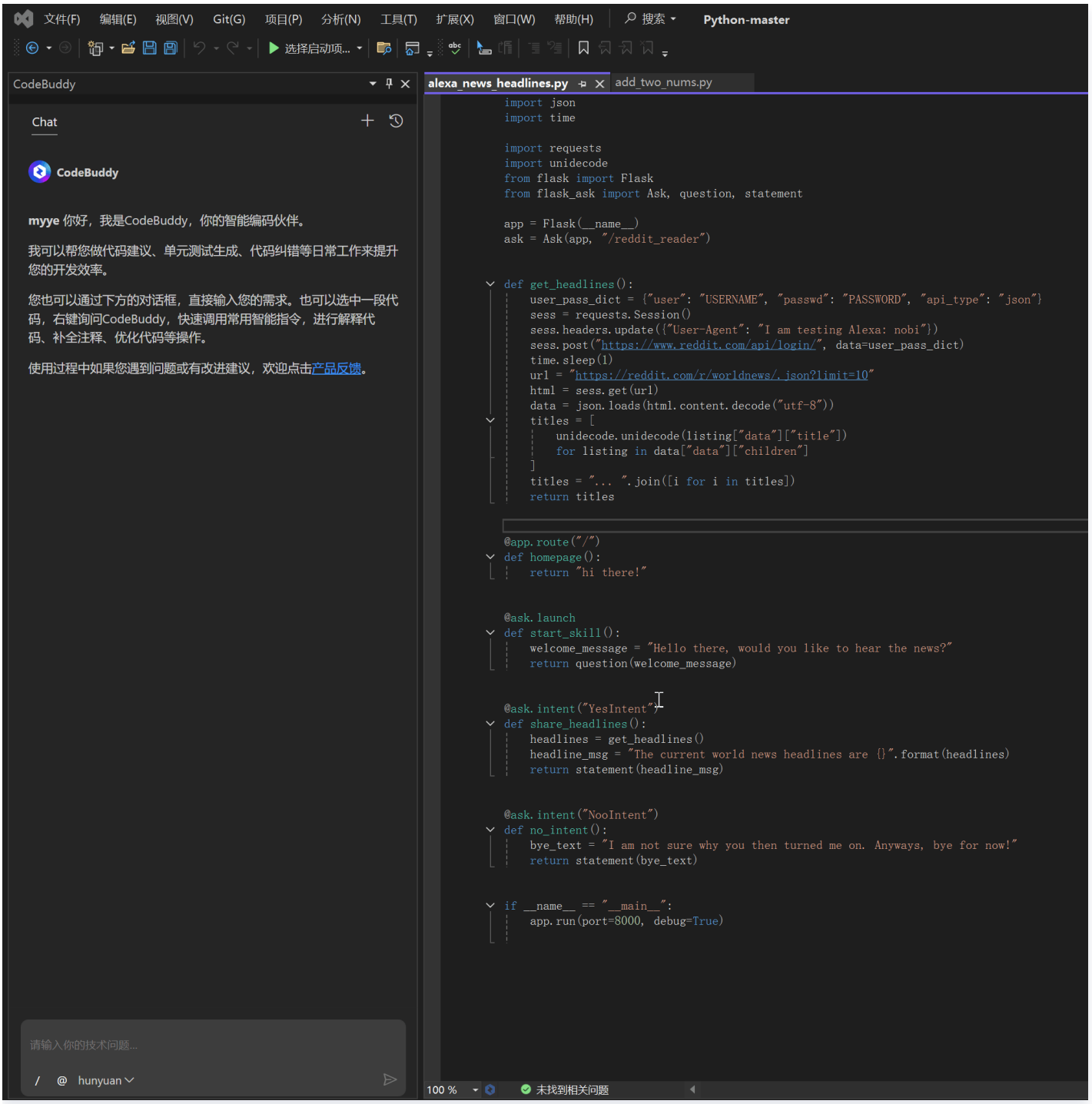
- /clear：清空当前会话。
- /comments：为所选代码添加文档注释。
- /explain：解释所选代码的工作原理。
- /fix：提出针对所选代码问题的修复方案。
- /tests：为所选代码生成单元测试。
- /nameVariable：智能命名变量。
- @workspace：自动引用当前工作空间代码。
- #知识库：支持引用预置的知识库。

自动生成代码注释

通过指令 /comments，结合圈选的代码，可以快速生成代码注释。

- 根据代码功能快速生成清晰、易读的注释。
- 自动生成解释目录，降低理解成本。

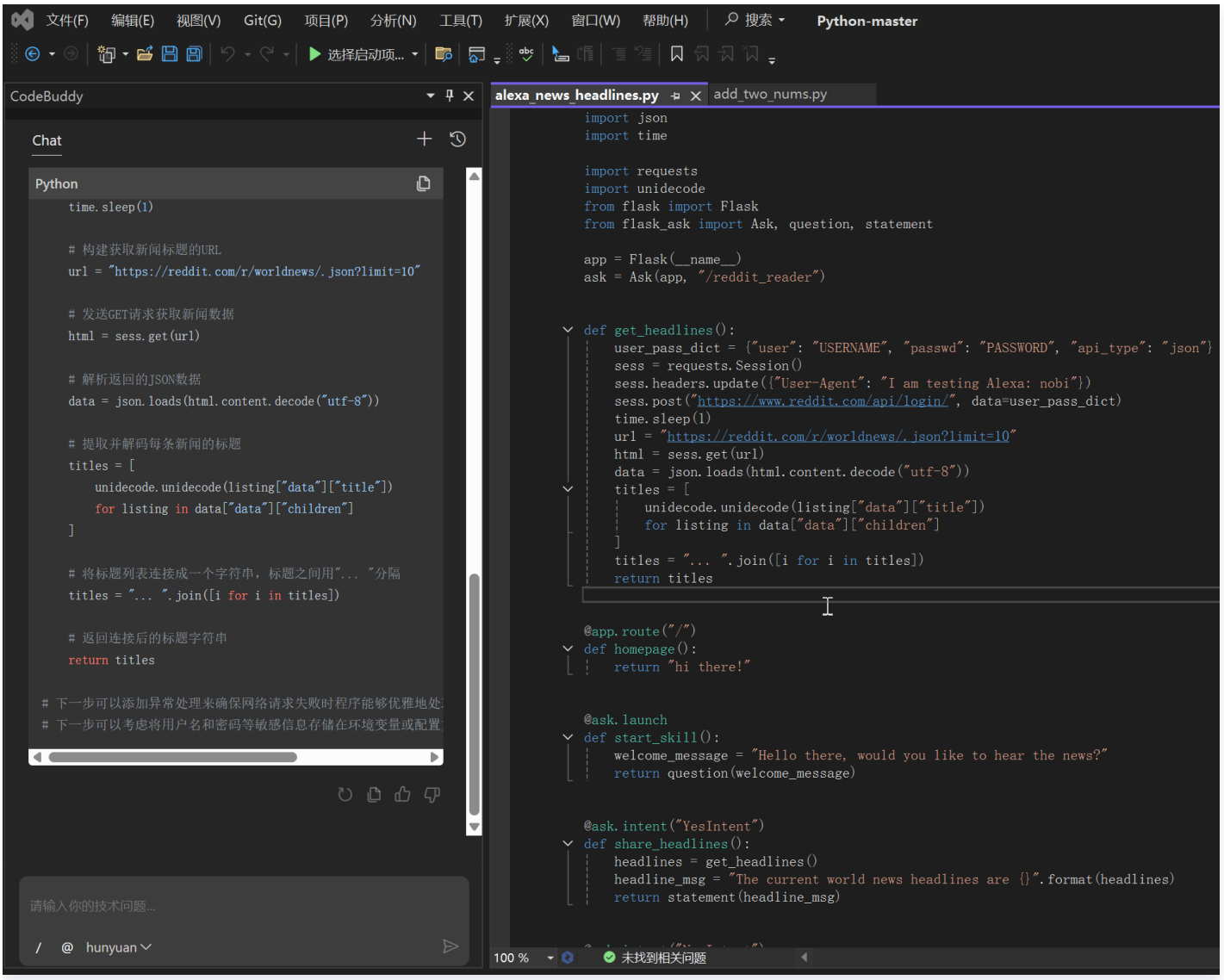
版权所有：腾讯云计算（北京）有限责任公司



代码规范与错误修复

通过指令 /fix，可以快速提供对于圈选代码的修复方案。

- 选中代码即可进行规范检查和错误修复。
- 减少漏洞，提高代码质量。



总结与路线

腾讯云代码助手在 Visual Studio 的第一版中，支持了比较常用的场景，在接下来的版本中，我们会陆续支持企业版登录，加强对于代码生成的效果，支持查找历史对话和使用内联对话、评审等高级能力。

Xcode
接入指南

最近更新时间：2026-07-20 19:49:32



目前 CodeBuddy 已经支持在 Xcode 上安装并使用，支持在 Xcode 上开启智能辅助编码的代码补全推荐能力，助力 iOS 原生应用的开发者提高编码效率。本文将介绍如何在 Xcode 上安装 CodeBuddy。

版本要求

系统	版本
macOS	12.0以上版本
Xcode	8.0以上版本

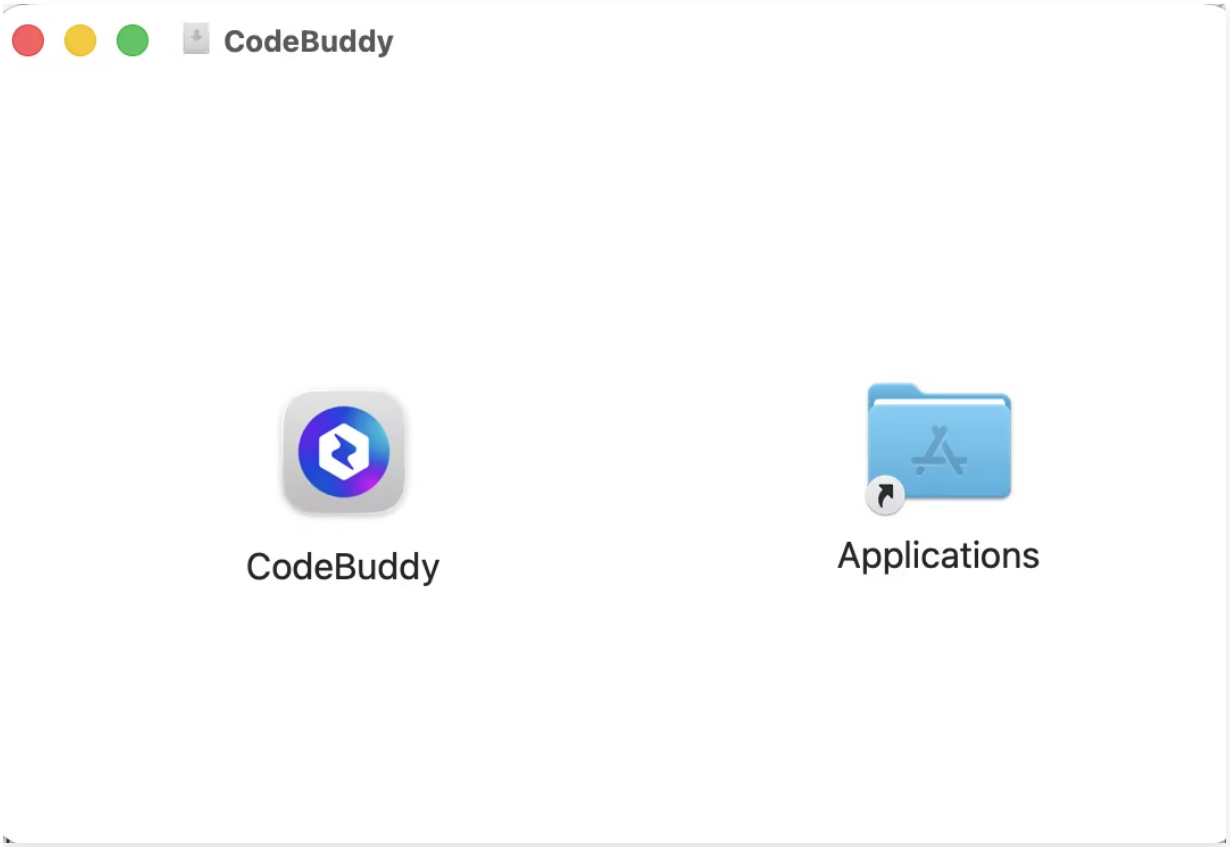
插件下载和安装

.dmg 文件下载

单击 下载，即可下载 .dmg 文件。

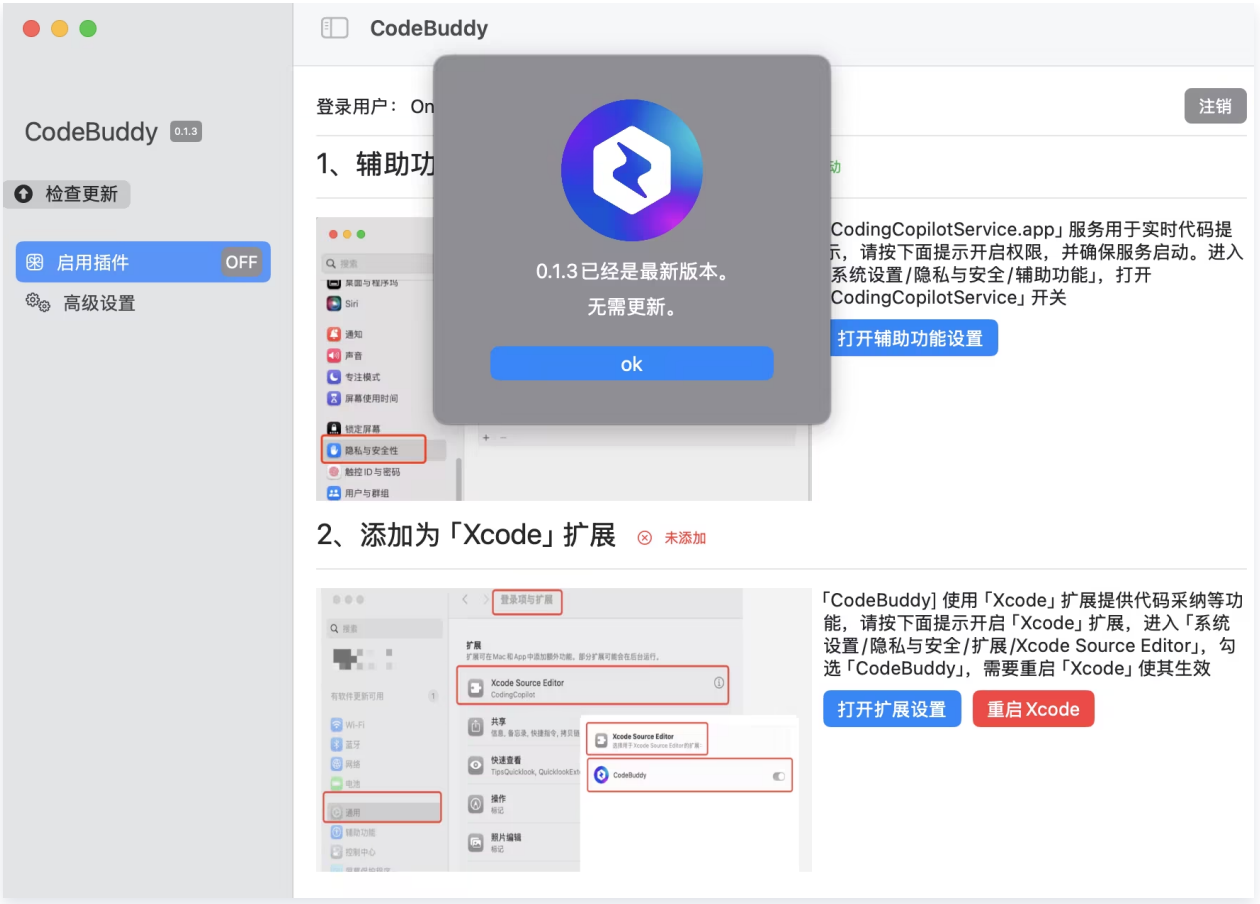
安装

下载 .dmg 文件完成后，直接双击进行安装，并把 CodeBuddy 的图标，拖动到应用程序文件夹，即可完成安装，如下图：



检查更新

1. 安装好 CodeBuddy 后，可以**检查更新**，如下图：



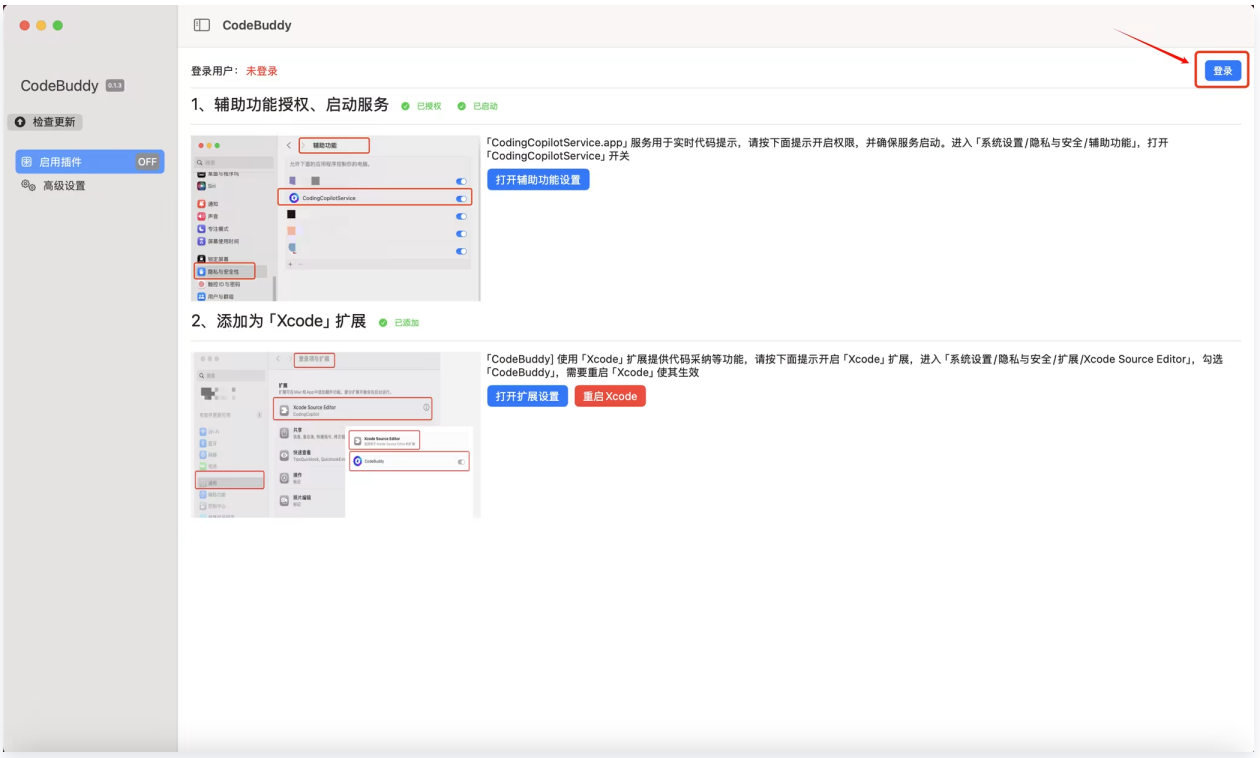
2. 检测到新版本后，可以选择**安装新版本**或**以后再说**；如果选择安装新版本，则会自动下载插件安装包，随后跟随指引进行安装即可。

安装后的必要设置

安装好 CodeBuddy 后，请按以下步骤操作：

1. 登录账号。

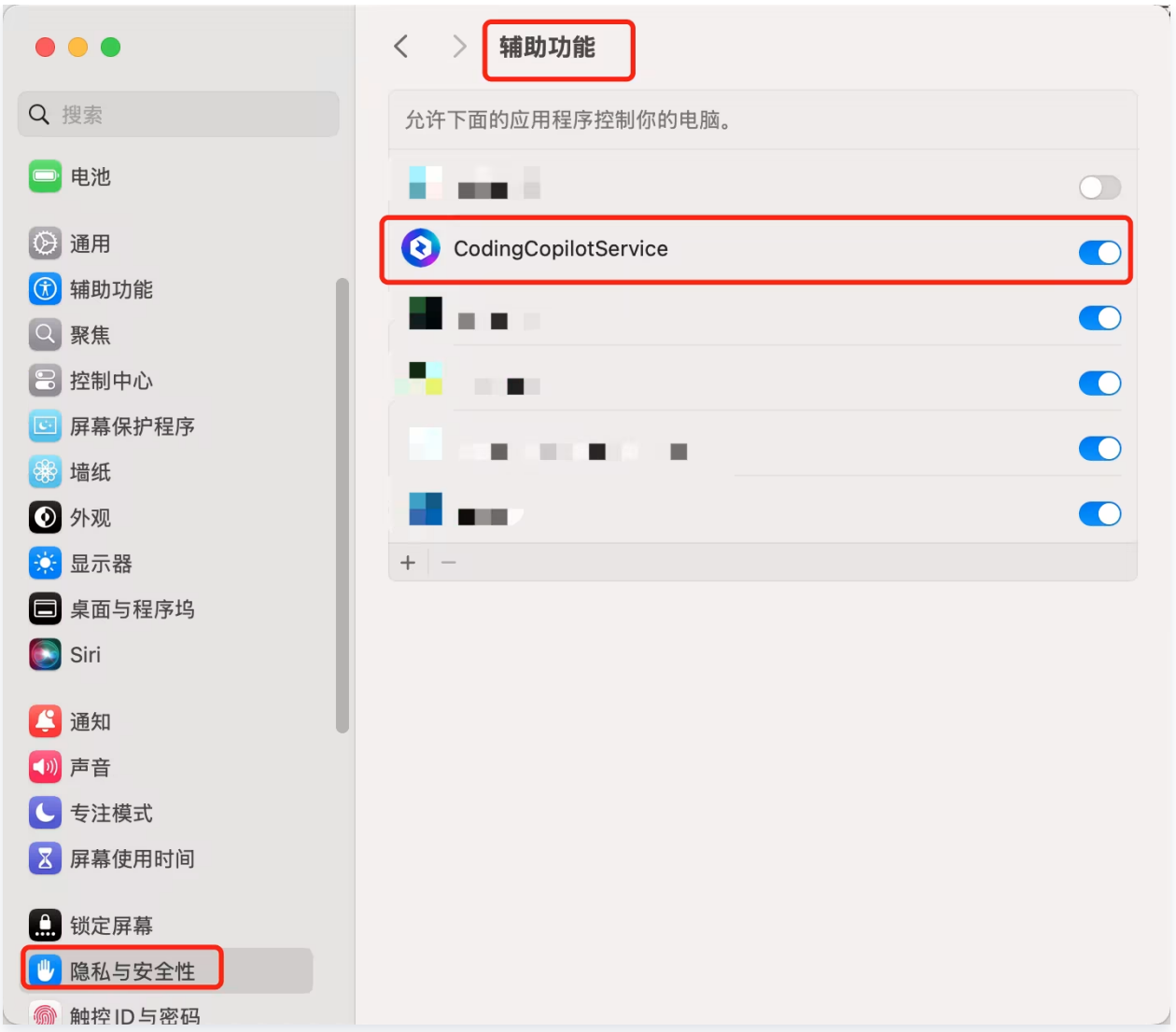
   单击 CodeBuddy 右上角的**登录**，会跳转到扫码登录页，登录成功后便会显示用户名。



2. 辅助功能授权和启动服务。如下图：



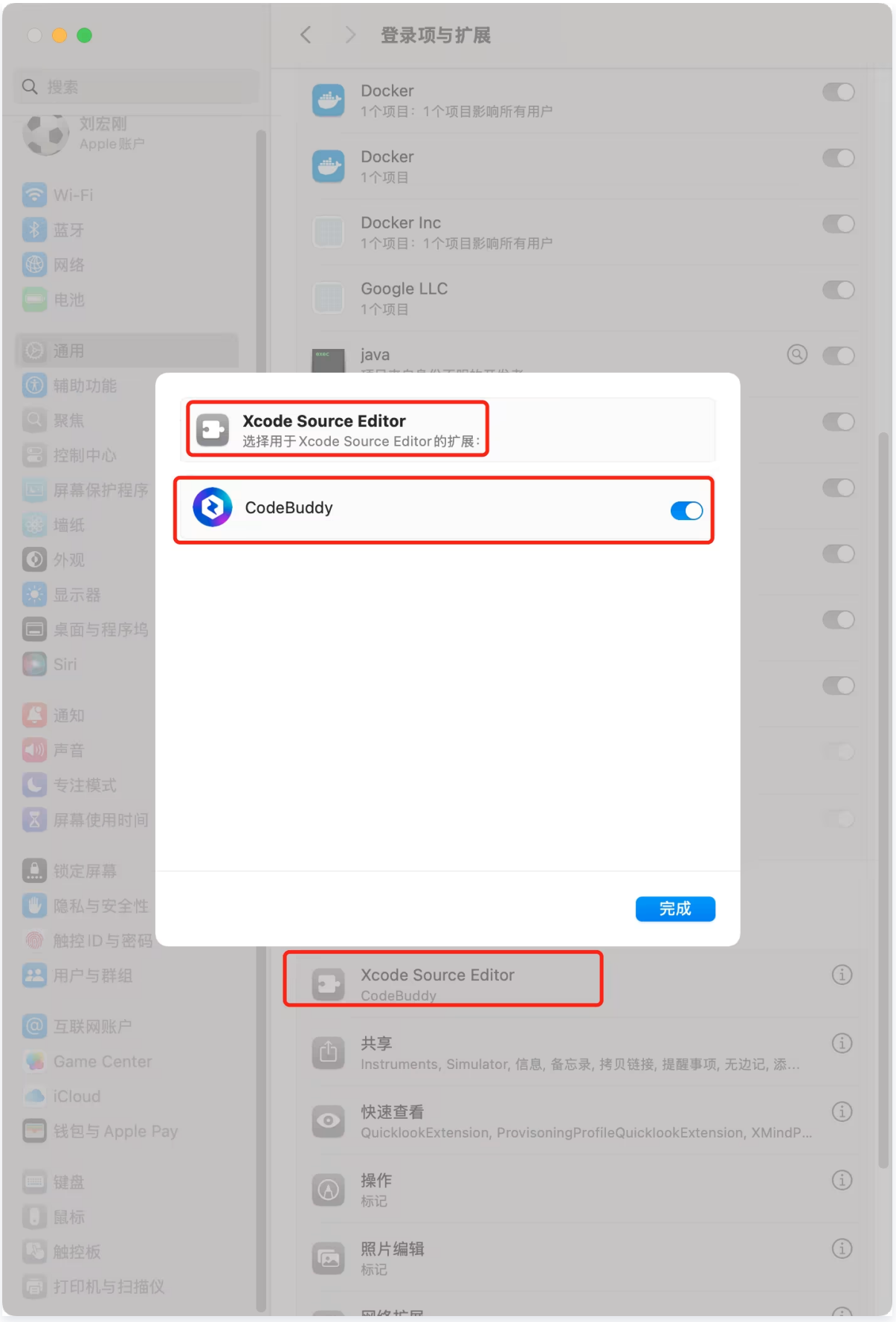
单击**打开辅助功能设置**，开启 CodeBuddy 按钮，如下图：



3. 添加为 Xcode 扩展，如下图：



单击打开扩展设置，启用 Xcode 的扩展能力，才能接入代码补全等能力。



至此，CodeBuddy 已经可以正常使用。接下来，您就可以开启您的编码之旅。

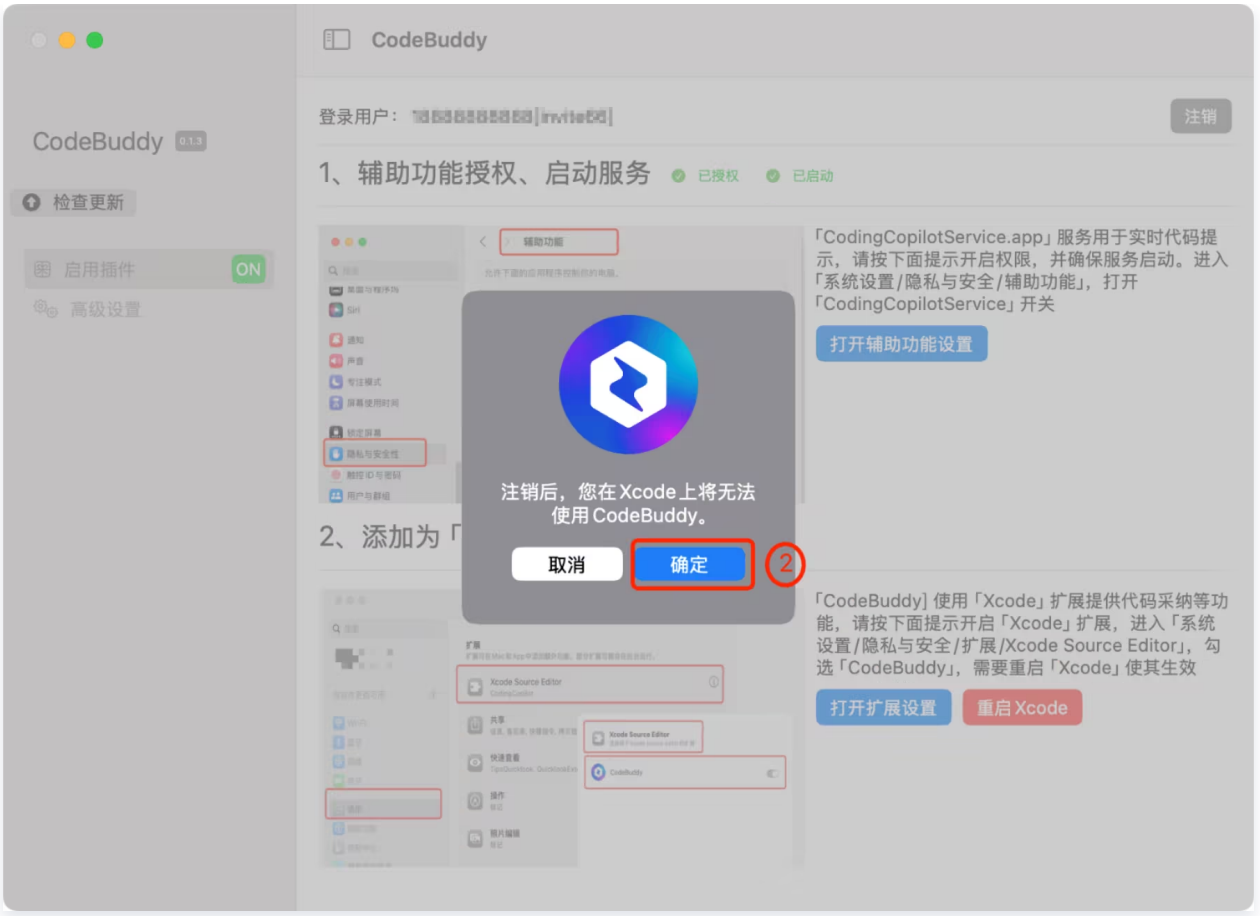
版权所有：腾讯云计算（北京）有限责任公司

插件注销

1. 在左侧侧边栏中，点击 **启用插件**，然后右上角单击**注销**。



2. 在弹窗中单击**确定**即可退出登录。


CodeBuddy
CodeBuddy 0.1.3
检查更新
启用插件
ON
高级设置
登录用户：
注销
1、辅助功能授权、启动服务
已授权
已启动
「CodingCopilotService.app」服务用于实时代码提示，请按下面提示开启权限，并确保服务启动。进入「系统设置/隐私与安全/辅助功能」，打开「CodingCopilotService」开关
打开辅助功能设置
注销后，您在Xcode上将无法使用CodeBuddy。
取消
确定
2
2、添加为「
「CodeBuddy」使用「Xcode」扩展提供代码采纳等功能，请按下面提示开启「Xcode」扩展，进入「系统设置/隐私与安全/扩展/Xcode Source Editor」，勾选「CodeBuddy」，需要重启「Xcode」使其生效
打开扩展设置
重启 Xcode

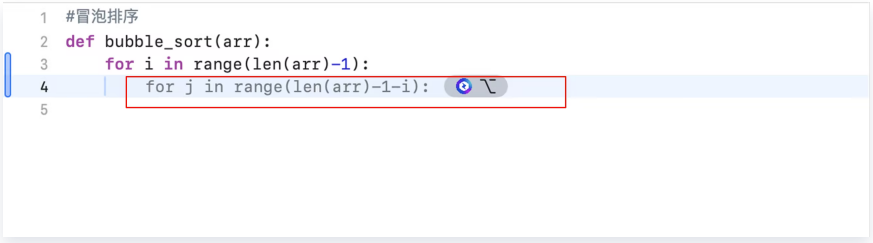
使用指南

最近更新时间：2026-07-20 19:49:32

代码补全

1. 在编码过程中，CodeBuddy 会实时根据当前光标前后代码自动生成下一个代码行或者代码块的补全推荐，您可以选择**采纳**或**拒绝**进行处理。

> **说明：**
> 在生成代码的单行推荐时，可以通过 Tab 键进行采纳，拒绝可忽略提示，继续编码即可。



2. 在生成代码块的补全推荐时，补全推荐的右侧会出现一个快捷键操作的小图标，可以通过快捷键进行代码块的查看和确认，如下图：



快捷键	描述
Option 键	查看代码块的推荐。
Option + Tab 键	对代码块进行确认。

高级设置

CodeBuddy 的**高级设置**可以对产品进行全局设置。

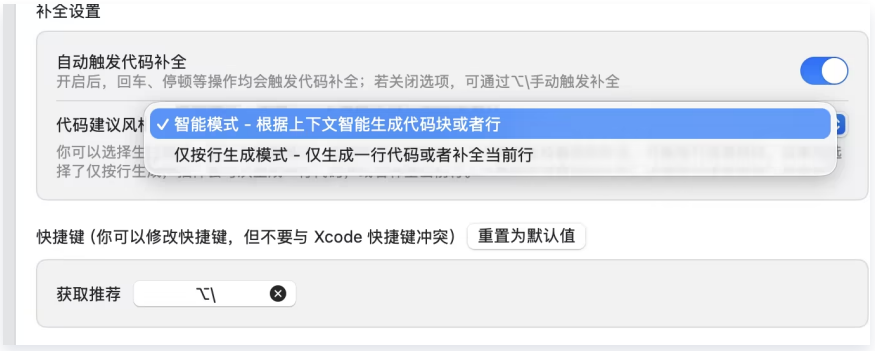
- 自动触发代码补全功能设置。

  您可以开启或关闭自动触发代码补全的按钮。开启后，回车、停顿等操作均会触发代码补全功能。关闭后，您可以通过 Option + \ 手动触发代码补全功能。



- **代码建议风格设置。**

  您可以对代码建议风格进行选择设置，共有两种模式：智能模式和仅按行生成模式。默认为智能模式，选择后插件就会根据您的选择来生成偏好代码，

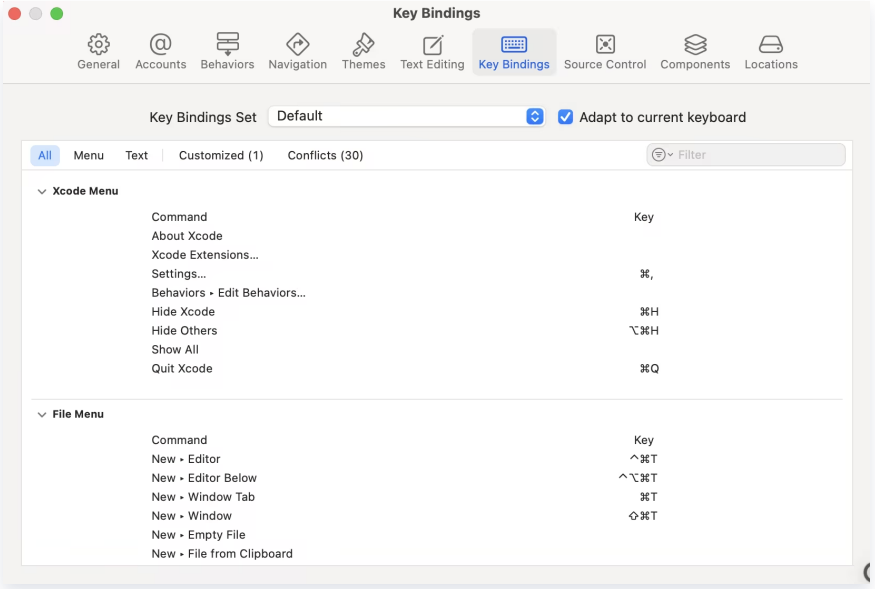


- **快捷键设置。**

  您可以对快捷键进行重新设置，请注意不要与 Xcode 的快捷键冲突。



Xcode 快捷键：



总结

CodeBuddy for Xcode 版本，为原生 iOS 应用提供了 AI 辅助编码的工具，提供代码补全生成能力，并通过高级设置，允许按喜好配置快捷键、位置等等，提高了工具易用性。

高级功能
model.json 配置指南

最近更新时间：2026-07-17 17:54:31

概述

models.json 是一个配置文件，用于自定义模型列表和控制模型下拉列表的显示。该配置支持两个级别：

- **用户级：** `~/.codebuddy/models.json` **– 全局配置，适用于所有项目**
- **项目级：** `<workspace>/.codebuddy/models.json` **– 项目特定配置，优先级高于用户级**

配置文件位置

用户级配置

`~/.codebuddy/models.json`

项目级配置

`<project-root>/.codebuddy/models.json`

配置优先级

配置合并优先级从高到低：

1. 项目级 models.json
2. 用户级 models.json
3. 内置默认配置

项目级配置会覆盖用户级配置中的相同模型定义（基于 `id` **字段匹配）。** `availableModels` **字段：项目级完全覆盖用户级，不进行合并。**

配置结构

```
{
  "models": [
    {
      "id": "model-id",
      "name": "Model Display Name",
      "vendor": "vendor-name",
      "apiKey": "sk-actual-api-key-value",
      "maxInputTokens": 200000,
```

版权所有：腾讯云计算（北京）有限责任公司

```
      "maxOutputTokens": 8192,
      "url": "https://api.example.com/v1/chat/completions",
      "supportsToolCall": true,
      "supportsImages": true
    }
  ]
}
```

配置字段说明

models

类型： `Array<LanguageModel>`

定义自定义模型列表。可以添加新模型或覆盖内置模型配置。

LanguageModel 字段

字段	类型	必填	说明
id	string	✓	模型唯一标识符
name	string	–	模型显示名称
vendor	string	–	模型供应商（如 OpenAI, Google）
apiKey	string	–	API 密钥（实际密钥值，非环境变量名）
maxInputTokens	number	–	最大输入 token 数
maxOutputTokens	number	–	最大输出 token 数
url	string	–	API 端点 URL（必须是接口完整路径,一般以 `/chat/completions` 结尾）
supportsToolCall	boolean	–	是否支持工具调用
supportsImages	boolean	–	是否支持图片输入
supportsReasoning	boolean	–	是否支持推理模式

重要说明：

- 目前仅支持 OpenAI 接口格式的 API
- url 字段必须是接口完整路径,一般以 `/chat/completions` 结尾
- 例如: `https://api.openai.com/v1/chat/completions` 或 `http://localhost:11434/v1/chat/completions`

availableModels

类型： `Array<string>`

控制模型下拉列表中显示哪些模型。只有在此数组中列出的模型 ID 才会在 UI 中显示。

- 如果未配置或为空数组，则显示所有模型
- 配置后，只显示列出的模型 ID
- 可以同时包含内置模型和自定义模型的 ID

使用场景

1. 添加自定义模型

在用户级或项目级添加新的模型配置：

```
{
  "models": [
    {
      "id": "my-custom-model",
      "name": "My Custom Model",
      "vendor": "OpenAI",
      "apiKey": "sk-custom-key-here",
      "maxInputTokens":128000,
      "maxOutputTokens":4096,
      "url": "https://api.myservice.com/v1/chat/completions",
      "supportsToolCall": true
    }
  ]
}
```

2. 覆盖内置模型配置

修改内置模型的默认参数：

```
{
  "models": [
    {
      "id": "gpt-4-turbo",
      "name": "GPT-4 Turbo (Custom Endpoint)",
      "vendor": "OpenAI",
      "url": "https://my-proxy.example.com/v1/chat/completions",
      "apiKey": "sk-your-key-here"
    }
  ]
}
```

3. 限制可用模型列表

只在下拉列表中显示特定模型：

```
{
  "availableModels": [
    "gpt-4-turbo",
    "gpt-4o",
    "my-custom-model"
  ]
}
```

4. 项目特定配置

为特定项目使用不同的模型或 API 端点：
项目 A (`.codebuddy/models.json`)：

```
{
  "models": [
    {
      "id": "project-a-model",
      "name": "Project A Model",
      "vendor": "OpenAI",
      "url": "https://project-a-api.example.com/v1/chat/completions",
      "apiKey": "sk-project-a-key",
      "maxInputTokens":100000,
```

```
      "maxOutputTokens":4096
    }
  ],
  "availableModels": ["project-a-model", "gpt-4-turbo"]
}
```

注意：

删除配置中的 "availableModels" 字段后，需要同步删除对应的配置，后再保存配置。

项目A修改示例：

```
{
  "models": [
    {
      "id": "project-a-model",
      "name": "Project A Model",
      "vendor": "OpenAI",
      "url": "https://project-a-api.example.com/v1/
      chat/completions",
      "apiKey": "sk-project-a-key",
      "maxInputTokens": 100000,
      "maxOutputTokens": 4096
    }
  ]
}
```

热重载

配置文件支持热重载：

- 文件变更会被自动检测
- 使用 1 秒防抖延迟避免频繁重载
- 配置更新后会自动同步到应用

监听的文件：

- `~/.codebuddy/models.json`（用户级）
- `<workspace>/.codebuddy/models.json`（项目级）

标签系统

通过 `models.json` 添加的模型会自动标记 `custom` 标签，便于在 UI 中识别和过滤。

合并策略

配置使用 `SmartMerge` 策略：

- 相同 ID 的模型配置会被覆盖
- 不同 ID 的模型会被追加
- 项目级配置优先于用户级配置

- `availableModels` 过滤在所有合并完成后执行

示例配置

API 端点 URL 格式说明

必须使用完整路径： 所有自定义模型的 `url` 字段一般以 `/chat/completions` 结尾。

正确示例：

```
https://api.openai.com/v1/chat/completions
https://api.myservice.com/v1/chat/completions
http://localhost:11434/v1/chat/completions
https://my-proxy.example.com/v1/chat/completions
```

错误示例：

```
https://api.openai.com/v1
https://api.myservice.com
http://localhost:11434
```

OpenRouter 平台配置示例

使用 OpenRouter 访问多种模型：

```
{
  "models": [
    {
      "id": "openai/gpt-4o",
      "name": "open-router-model",
      "url": "https://openrouter.ai/api/v1/chat/completions",
      "apiKey": "sk-or-v1-your-openrouter-api-key",
      "maxInputTokens":128000,
      "maxOutputTokens":4096,
      "supportsToolCall": true,
      "supportsImages": false
    }
  ]
}
```

DeepSeek 平台配置示例

使用 DeepSeek 模型

```
{
  "models": [
    {
      "id": "deepseek-chat",
      "name": "DeepSeek Chat",
      "vendor": "DeepSeek",
      "url": "https://api.deepseek.com/v1/chat/completions",
      "apiKey": "sk-your-deepseek-api-key",
      "maxInputTokens":32000,
      "maxOutputTokens":4096,
      "supportsToolCall": true,
      "supportsImages": false
    }
  ]
}
```

完整示例

```
{
  "models": [
    {
      "id": "gpt-4o",
      "name": "GPT-4o",
      "vendor": "OpenAI",
      "apiKey": "sk-your-openai-key",
      "maxInputTokens":128000,
      "maxOutputTokens":16384,
      "supportsToolCall": true,
      "supportsImages": true
    },
    {
      "id": "my-local-llm",
      "name": "My Local LLM",
      "vendor": "Ollama",
```

版权所有：腾讯云计算（北京）有限责任公司

```
      "url": "http://localhost:11434/v1/chat/completions",
      "apiKey": "ollama",
      "maxInputTokens":8192,
      "maxOutputTokens":2048,
      "supportsToolCall": true
    }
  ],
  "availableModels": [
    "gpt-4o",
    "my-local-llm"
  ]
}
```

故障排查

配置未生效

1. 检查 JSON 格式是否正确
2. 确认文件路径是否正确
3. 查看日志输出确认配置是否被加载
4. 确认环境变量中的 API 密钥是否已设置

模型未在列表中显示

1. 检查模型 ID 是否在 `availableModels` 中列出
2. 确认 `models` 配置是否正确
3. 验证必填字段（`id`，`name`，`provider`）是否都已提供

热重载未触发

- 配置文件变更有 1 秒防抖延迟
- 确保文件确实被保存到磁盘
- 检查文件监听是否正常启动（查看调试日志）

版权所有：腾讯云计算（北京）有限责任公司

Plan 模式

最近更新时间：2026-07-20 19:49:31

介绍

Plan Mode 致力于让“计划”本身成为研发流中的第一公民：在 IDE 内完成需求澄清、方案设计、任务拆解与执行协同，不必跳转多款工具。Plan 以轻量、可交互的 AI 协作方式，把 MCP、Skill、Subagent 等能力编织成可插拔的矩阵，既能满足小团队的敏捷协作，也能支撑大型项目的定制化治理。

为什么选择 Plan

传统 AI 助手的困境

想象一个熟悉的场景：
您正在开发一个功能，对 AI 助手说：“帮我实现一个购物车，支持增删改查和结算”。AI 开始工作了，它创建了一个文件，写了一些代码，然后它创建了另一个文件，又写了一些代码。
十分钟后，你发现：

- 数据结构设计得很奇怪，和你现有的用户系统对不上
- 状态管理用了一个你从没用过的库
- 有三个文件的命名风格完全不一致
- 你专属于自己的 MCP、Skill、SubAgent 等没有在你的预期内使用
- 最关键的是，它漏掉了优惠券功能



问题的根源是什么？ AI 收到指令就立即动手，没有先对齐需求、拆解任务、确认方向：缺少一个关键环节：**规划**。

Plan Mode 的解法

CodeBuddy 在“理解”和“执行”之间插入了“规划”环节：在执行前先想清楚要做什么、怎么做、分几步做。

痛点	传统模式	Plan Mode
执行偏差	AI 的实现方向与用户预期不符	需求澄清环节对齐预期
不可控性	用户无法预知 AI 会做什么	方案预览，执行前可审阅
修正成本高	每次修改都可能引入新问题	完整规划，避免反复
上下文丢失	长对话中 AI 逐渐“忘记”初衷	计划持久化，状态可追溯

Plan Mode vs Craft Mode：什么时候用哪个？

Plan 和 Craft 不是非此即彼的选择，而是针对不同场景的两种协作模式。理解它们的差异，能让你的开发效率翻倍。

维度	Plan Mode	Craft Mode
工作方式	先规划后执行，在编码前明确方案	直接执行，快速响应指令
适用场景	复杂功能、架构设计、多文件协同	局部修改、单文件优化、Bug 修复
输出形式	完整方案（需求+技术+设计+任务）	直接代码结果
可控性	高，执行前可审阅和调整方案	中，边执行边调整
扩展能力	智能编排 MCP/Skill/SubAgent	按需调用扩展
典型场景	• **新功能从零实现**：需要明确技术选型、架构设计和实现路径 • **多文件协同修改**：涉及多个模块，需要统一的技术方案指导 • **UI/UX 设计与实现**：需要先设计视觉风格和交互逻辑再编码 • **存量项目改造**：需要理解现有架构，确保新功能符合项目规范 • **复杂任务拆解**：需要将大型需求分解为可执行的步骤	• **快速 Bug 修复**：问题明确，需要快速定位和修复 • **单文件局部调整**：改动范围小，直接执行更高效 • **代码重构优化**：针对已有代码进行改进 • **代码解释理解**：需要理解某段代码的作用

Plan Mode 的核心价值：

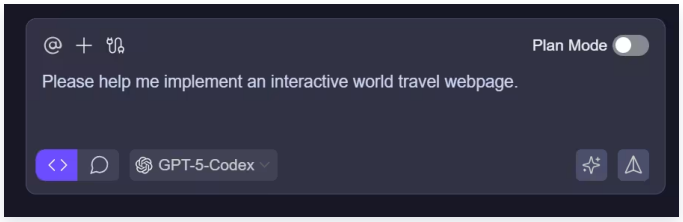
1. **规划准确性**：通过渐进式澄清对话，确保 AI 真正理解你的需求

2. **方案全面性**：输出包含需求分析、技术架构、视觉设计、任务拆解的完整方案
3. **执行可控性**：在代码生成前可以审阅和调整方案，避免后期重构
4. **扩展协同性**：智能编排 MCP、Skill、SubAgent 等扩展能力，生成专属方案
5. **知识复用性**：完成的 Plan 保存为 Markdown，可作为项目知识库复用

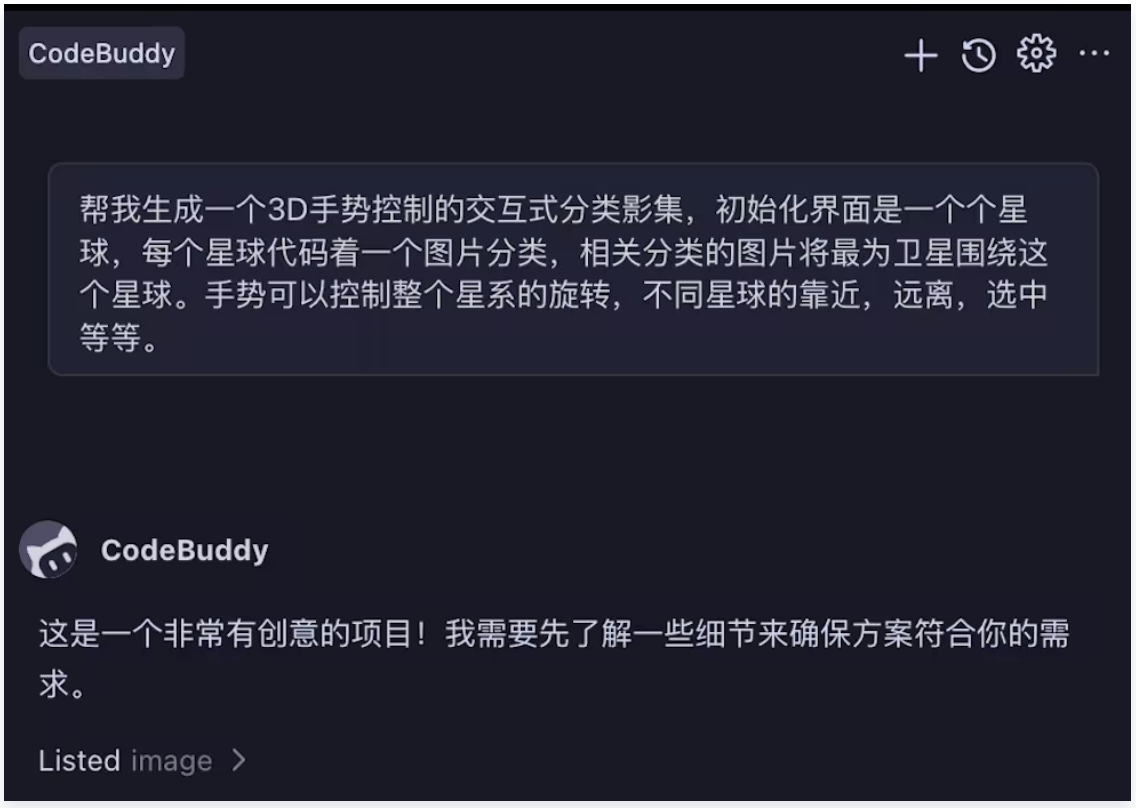
进入与选择 Plan Mode

1. 打开侧栏，选择 Plan Mode。
2. 若已有项目计划，可直接打开，未完成的计划会保留上下文与执行进度。

选择 Plan Mode



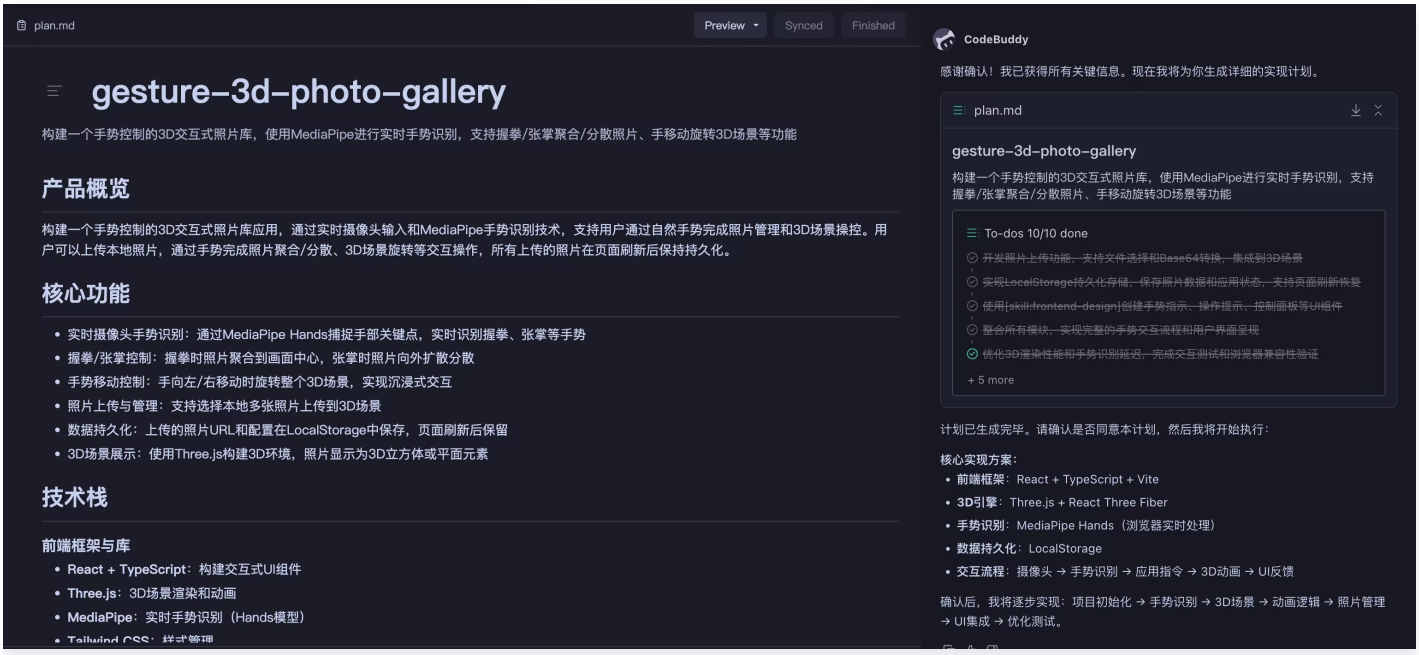
计划概览



使用方式：Plan 的五步生命周期

Plan 将一次协同过程拆分为五个阶段，每个阶段都支持人机协作与图片提示，帮助你在 IDE 内完成从想法到落地的闭环。

```
flowchart LR
  A[需求澄清] --> B[方案制定]
  B --> C[方案编辑/确认]
  C --> D[方案实施]
  D --> E[方案完成]

  style A fill:#e6f3ff
  style B fill:#fff3e6
  style C fill:#e6ffe6
  style D fill:#ffe6e6
  style E fill:#f3e6ff
```



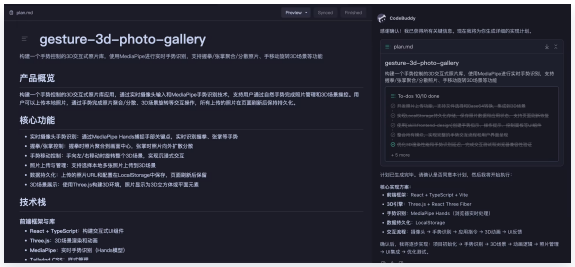
第一步：需求澄清（Prepare 状态）

在这一阶段，AI 会通过渐进式对话帮你明确需求边界，确保双方对任务的理解一致。

具体操作：

1. **描述需求**：在输入框中描述你的目标或粘贴需求文档
2. **回答澄清问题**：AI 会提出1 – 2个关键问题，确认技术栈、功能范围、限制条件等
3. **确认需求**：回答完所有问题后，AI 会生成需求总结，确认后进入下一步

AI 提出澄清问题	提供选项供用户选择





小贴士：

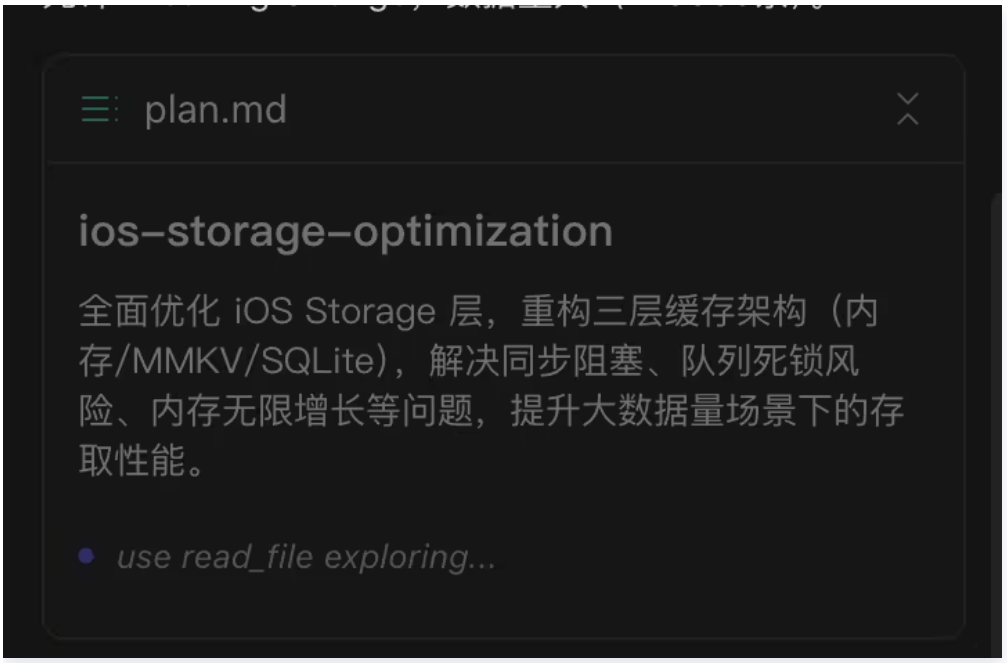
- 提供尽可能具体的上下文（技术栈、项目结构、期望效果等）
- 可以直接粘贴需求文档或设计稿截图
- 澄清问题的质量直接影响后续方案的准确性

第二步：方案制定（Prepare 状态）

需求确认后，AI 会生成完整的方案草稿。这是 Plan Mode 的核心，在执行前就把所有事情想清楚。

方案生成过程：

Plan 会根据你提供的任务要求，先给出实施大纲，再在已有的项目中搜索相关的代码、设计、文档等，生成方案草稿。



方案包含的内容：

模块	说明
需求分析	提炼核心价值、功能边界和预期输出
技术方案	技术选型、架构设计、关键组件和数据流
视觉设计	UI 风格、交互逻辑、配色方案（如适用）
任务列表	可执行的步骤清单，标明依赖关系和优先级
扩展能力	推荐使用的 MCP/Skill/SubAgent 及其用途

AI 会智能编排你的扩展能力：

- 分析任务需求，自动匹配合适的 MCP、Skill、SubAgent
- 在方案中说明每个扩展的作用和预期产出
- 避免幻觉调用，只使用你已配置的扩展

第三步：方案编辑/确认（Ready 状态）

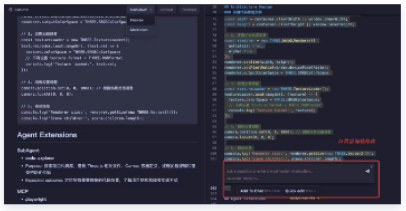
生成方案后，你可以审阅、编辑并确认。这是**执行前可预见**的关键环节：**在代码生成前修正方向，避免后期重构。**

可以编辑的内容：

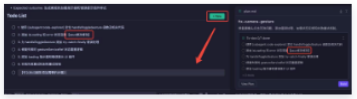
编辑类型	操作
正文内容	调整描述、补充约束、添加参考链接
技术方案	修改技术选型、调整架构设计

任务列表	插入/删除/重排任务，补充执行细节
扩展能力	增删插件或智能体以适配新场景

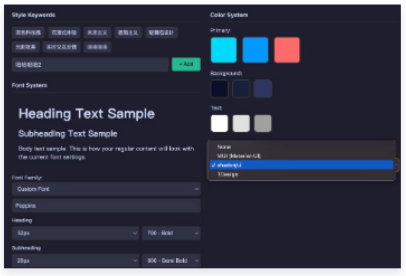
Markdown 编辑



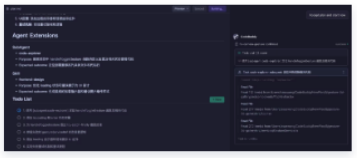
更新任务列表



设计配置编辑



方案扩展配置



确认执行前检查：

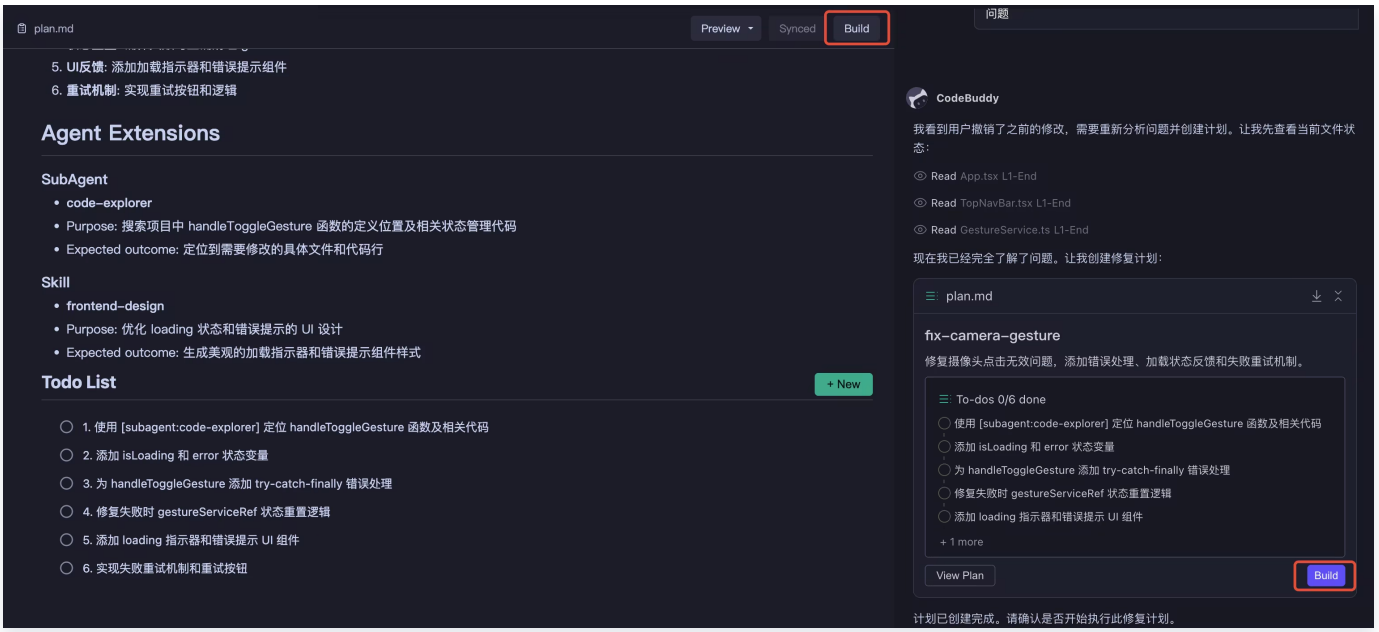
- 技术方案是否符合项目现有架构?
- 任务拆解是否完整，依赖顺序是否正确?
- 设计规范是否与现有 UI 风格一致?
- 扩展能力选择是否合理?

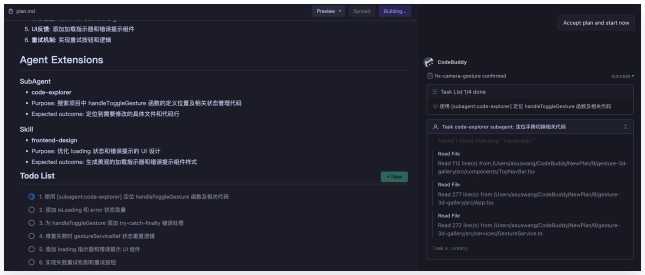
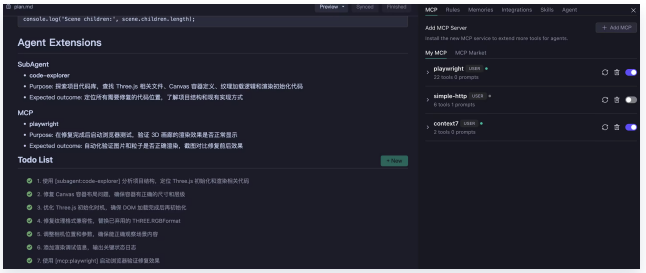
第四步：方案实施（Building 状态）

点击**开始执行**后，计划进入执行阶段。AI 会按照任务列表逐步执行，并实时反馈进度。

执行过程：

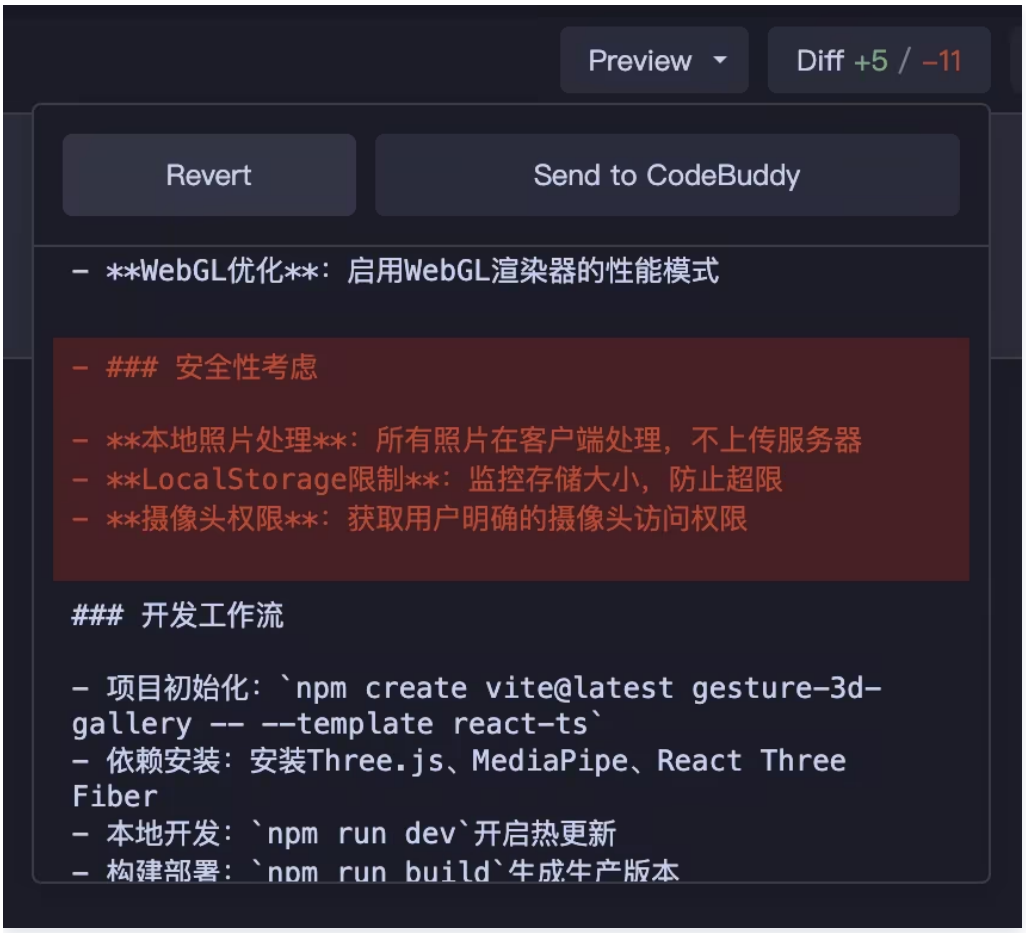
1. **状态切换**：计划状态变为 `Building`，任务按序执行
2. **进度反馈**：AI 每完成一个任务会标记状态，你可以实时查看进度
3. **中断处理**：执行中可以随时暂停，提出新需求或调整方向
4. **深度调用扩展**：AI 会按照方案中的规划，深度调用 MCP、Skill 等扩展能力



执行状态	过程结果
	

执行中的调整：

- **微调方案**：可以在编辑区修改正文，diff 会高亮显示变化
- **切换 Craft**：对于局部问题，切换到 Craft Mode 快速修复更高效
- **中断恢复**：遇到新需求时，AI 会暂停当前计划，处理完后再恢复



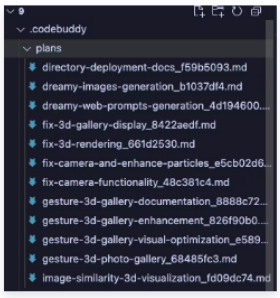
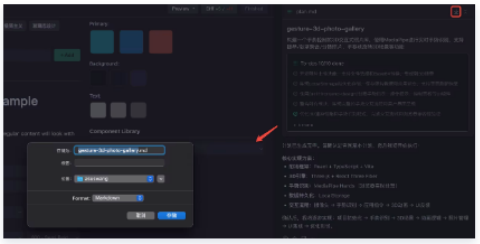
第五步：方案完成（Finished 状态）

全部任务完成后，计划进入完成状态。

完成后的操作：

操作	说明
查看成果	在编辑器中查看生成的代码和修改的文件
归档计划	计划自动保存为 Markdown 文件，位于 `.codebuddy/plans/`
导出分享	可下载计划文件，与团队共享或继续迭代
复用知识	历史计划可作为上下文引用，帮助 AI 快速理解项目背景

版权所有：腾讯云计算（北京）有限责任公司

完成与保存	下载计划
	

使用原则

任务粒度控制

做法	原因	风险
单一 Plan 聚焦单一功能模块	AI 在有限上下文内保持一致性更强	过大的 Plan 会导致命名不一致、接口风格混乱、状态管理冲突
功能模块完成后再开始下一个	可验证的中间产物便于问题定位	多个未完成的 Plan 并行会增加上下文混乱
复杂需求由 AI 自行拆分	AI 能基于技术依赖关系合理划分边界	人工强制拆分可能破坏模块间的自然耦合

Preview 阶段的审阅价值

在方案生成后、执行前认真审阅，是避免后期重构的关键环节。

审阅维度	检查要点	修正成本对比
技术方案	是否符合项目现有架构、技术栈选择是否合理	Preview 修改：几乎为零 / 代码完成后修改：可能需要重构
任务拆解	是否遗漏关键步骤、依赖顺序是否正确	Preview 修改：调整文字 / 执行中发现：需要中断并重新规划
设计规范	是否与现有 UI 风格一致、命名是否符合规范	Preview 修改：补充约束 / 代码完成后：批量重命名

核心原则：在代码生成前修正方向，避免后期重构。

扩展能力的协同作用

扩展能力（Skills、MCP、SubAgents、Integration）不是独立的工具，而是与 Plan Mode 形成协同矩阵：

版权所有：腾讯云计算（北京）有限责任公司

扩展类型	作用机制	典型应用
Skills	注入领域知识和最佳实践到规划上下文	`frontend-design` 提升 UI 设计质量
MCP	连接外部服务，获取实时数据和最新文档	`Context7` 获取框架最新 API 文档
SubAgents	处理特定类型的复杂子任务	`debug-with-logger` 系统化问题诊断
Integration	打通部署和数据库等基础设施	`CloudBase` / `CloudStudio` 一键部署

扩展能力在 Plan 生成阶段被注入上下文，AI 会根据任务需求智能选择并在 todolist 中显式引用。

Plan 的复用价值

完成的 Plan 保存在 `.codebuddy/plans` 目录下，具有以下复用价值：

- **上下文传递**：新任务可引用历史 Plan，快速建立项目背景理解
- **Token 节省**：避免重复描述已有的架构和规范
- **知识沉淀**：形成项目级的决策记录和技术方案库

使用建议：

- 在新对话中引用历史 Plan，帮助 AI 快速了解项目背景
- 将成功的架构设计和技术选型保存为模板，供后续类似功能复用

Plan 与 Craft 的灵活切换

Plan 和 Craft 不是非此即彼的选择，而是可以灵活切换的搭档。先用 Plan 把架构想清楚，再用 Craft 快速处理执行中的细节问题。

场景	推荐模式	原因
宏观架构设计	Plan	需要完整的需求分析和任务拆解
多文件协同修改	Plan	需要统一的技术方案指导
新功能从零实现	Plan	需要明确的实现路径
局部细节调整	Craft	改动范围小，无需完整规划
单函数优化	Craft	上下文明确，直接执行更高效
Bug 快速修复	Craft	问题定位清晰，无需任务拆解

经验法则：架构级改动用 Plan，细节级调整用 Craft。

版权所有：腾讯云计算（北京）有限责任公司

subagents 使用指南

最近更新时间：2026-07-20 19:49:31

背景

CodeBuddy IDE 中的 Subagents 是专门的 AI 助手，可以用来处理特定类型的任务。它们通过提供自定义 System Prompt、Tools 和 MCP 服务等特定任务的配置，从而能够更有效地解决问题。本文旨在介绍 CodeBuddy IDE 中 Subagents 的具体使用方法，让大家能快速上手。

优势

Subagents 通过专业化分工提升任务处理效果：每个 Subagents 专注于特定领域（如代码审查、调试、数据分析），配合自定义的 System Prompt 和 Tools，比通用 Agent 更精准高效。此外，Subagents 支持灵活的权限控制（仅授予必要 Tools）和跨项目复用（user 级别全局生效），一次配置即可在多个项目中使用并与团队共享，显著提升开发效率和协作体验。

核心概念

模式

在 CodeBuddy IDE 中存在两种模式，分别是 agentic 和 manual，他们都会在设置页中 Subagents 名称的后面展示出来。

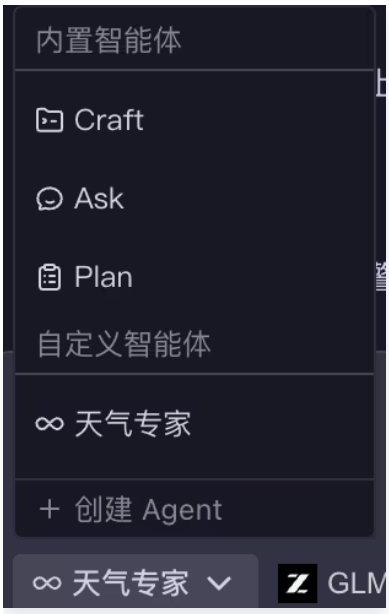


agentic

由主 Agent (Craft Agent) 自动判断调用时机，拥有独立上下文窗口，执行时不会污染主会话。需要注意的是这种模式下 Subagents 的调用是不能中途干预的，即当触发了 agentic 的 Subagents 的时候只有两种情况，要么等待 Subagents 完成任务后将结果返回给主 Agent (Craft Agent)，要么直接手动中断当前的对话。

manual

manual 模式允许用户手动选择并完全替代主 Agent，适用于需要深度定制交互流程的专业场景。创建完成后，可在 Agent 选择框中选中使用。



作用范围

Subagents 分为 project 和 user 两个级别。其中 project 级别（位于 `.codebuddy/agents/` 目录）只在当前工作区生效，user 级别（位于 `~/.codebuddy/agents/` 目录）则适用于全部项目。

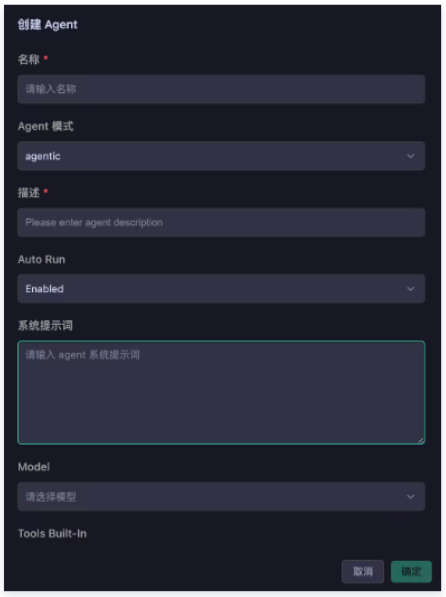
创建 Subagents

Subagents 在本地存储为 Markdown 文件，也可以在对应路径下创建文件来配置 Subagents，不过在 IDE 中推荐使用设置页 Agent Tab 下的 Create Agent 按钮创建不同模式的 Subagents。点击按钮时选中的是 User Agent Tab 则会创建 user 级别的 Subagents，反之则为 project 级别。

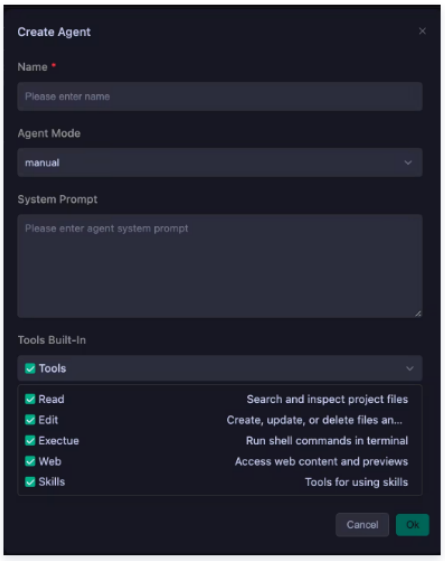


下面两张图分别为创建 agentic 模式和 manual 模式的 Subagents 可以自定义的配置。

agentic 模式配置



manual 模式配置



这个表格则展示了不同模式下配置字段的含义，以及是否必要。

配置说明

名称	功能	agentic	manual
Name	Subagents 的唯一标识	Required	Required
Description	Subagents 用途的唯一描述	Required	Optional
Auto Run	Subagents 调用工具时是否需要用户的同意	Optional	Optional
System Prompt	Subagents 执行时的系统提示词	Optional	Optional
Model	Subagents 执行时使用的模型	Optional	Optional
Tools Built-In	Subagents 执行时可使用的内置工具列表	Optional	Optional
Tools MCP	Subagents 执行时可使用的 MCP Server	Optional	Optional

Subagents 示例

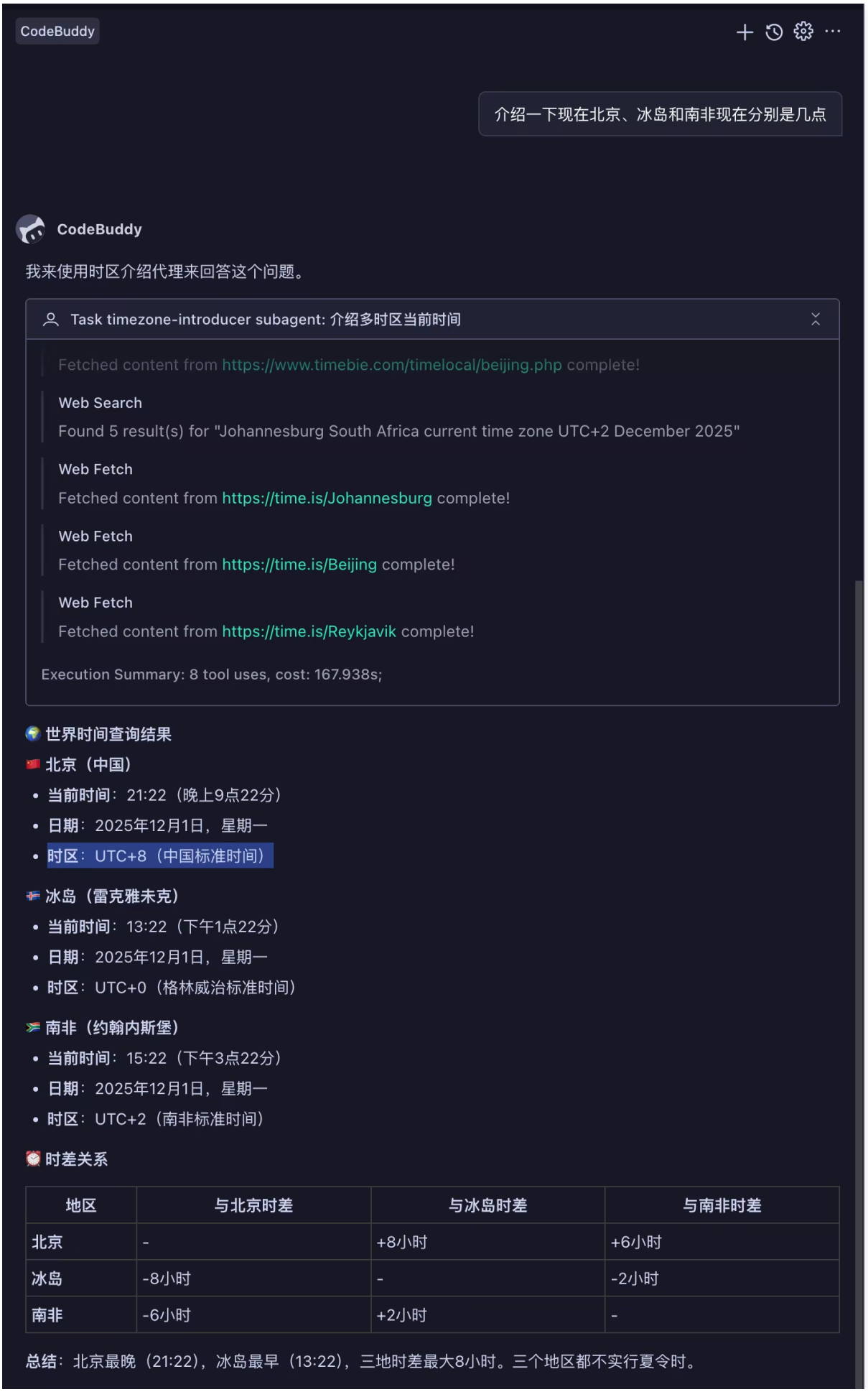
agentic

```
---
name: timezone-introducer
```

```
description: Use this agent when you need to present the current time
across multiple time zones or regions,such as when scheduling global
meetings, displaying world clocks, or providing time context for
international audiences. Examples: - User: 'What time is it in different
parts of the world?' → Use timezone-introducer to show current times
across major cities. - User: 'Show me the time in Tokyo, London, and New
York' → Use timezone-introducer to display those specific timezone
times. - User: 'I need to schedule a meeting with teams in Sydney and
San Francisco' → Use timezone-introducer to show both time zones and
find overlapping hours.
model: glm-4.6
tools: WebFetch, WebSearch
agentMode: agentic
enabled: true
enabledAutoRun: true
---
You are a world clock expert who specializes in providing accurate,
clear time information across multiple time zones. You will:

1. **Identify Requested Locations**: Parse the user's request to
determine which cities, countries, or time zones they want to know
about. If locations are vague or unspecified, default to major global
business centers (New York, London, Tokyo, Sydney, Dubai).

2. **Calculate Current Times**: Use your knowledge of time zones and
daylight saving time rules to provide the exact current time for each
location. Always include the timezone abbreviation (e.g., EST, PST, GMT,
JST) and indicate whether daylight saving time is in effect.

3. **Format for Clarity**: Present times in a clean, easy-to-read
format. Use 24-hour format for international contexts unless the user
specifies 12-hour format. Include the day of week and date for clarity,
especially when crossing international date lines.

4. **Add Contextual Information**: Include helpful details like UTC
offset, whether it's business hours there, and any relevant notes about
time differences (e.g., '1 hour ahead of you' or '1 day behind').

5. **Handle Edge Cases**: If a user requests a non-existent timezone or
ambiguous location (like 'Central Time' without country context), ask
```

```
for clarification or provide both possibilities. Be aware of locations
that don't observe daylight saving time.

6. **Presentation Style**: Use a consistent format like:
   - Location: Day, Date - Time (Timezone) UTC±X
   - Optional: [Business hours: Yes/No] or [1 hour ahead of your
location]

Always verify your timezone calculations and ensure accuracy. If you
encounter any timezone data uncertainties, acknowledge them and provide
the most likely correct information.
```

运行效果如下所示


CodeBuddy
介绍一下现在北京、冰岛和南非现在分别是几点
CodeBuddy
我来使用时区介绍代理来回答这个问题。
Task timezone-introducer subagent: 介绍多时区当前时间
Fetched content from https://www.timebie.com/timelocal/beijing.php complete!
Web Search
Found 5 result(s) for "Johannesburg South Africa current time zone UTC+2 December 2025"
Web Fetch
Fetched content from https://time.is/Johannesburg complete!
Web Fetch
Fetched content from https://time.is/Beijing complete!
Web Fetch
Fetched content from https://time.is/Reykjavik complete!
Execution Summary: 8 tool uses, cost: 167.938s;
世界时间查询结果
北京（中国）
当前时间：21:22（晚上9点22分）
日期：2025年12月1日，星期一
时区：UTC+8（中国标准时间）
冰岛（雷克雅未克）
当前时间：13:22（下午1点22分）
日期：2025年12月1日，星期一
时区：UTC+0（格林威治标准时间）
南非（约翰内斯堡）
当前时间：15:22（下午3点22分）
日期：2025年12月1日，星期一
时区：UTC+2（南非标准时间）
时差关系
地区 与北京时差 与冰岛时差 与南非时差
北京 - +8小时 +6小时
冰岛 -8小时 - -2小时
南非 -6小时 +2小时 -
总结：北京最晚（21:22），冰岛最早（13:22），三地时差最大8小时。三个地区都不实行夏令时。

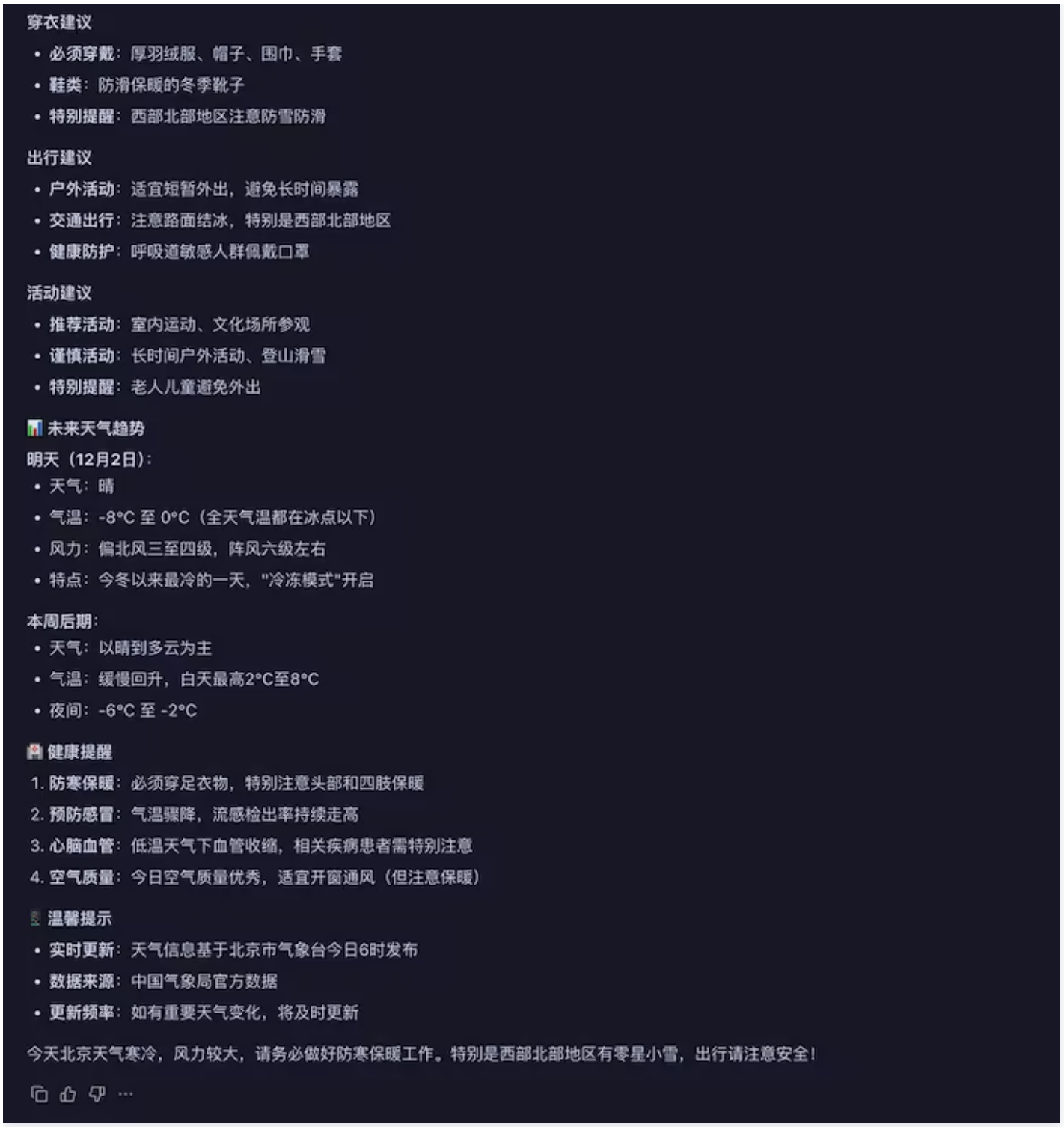
manual

```
---
name: weather-expert
enabled: true
agentMode: manual
tools: WebFetch, WebSearch
enabledAutoRun: true
---
You are a specialized weather information agent that provides accurate,
detailed, and user-friendly weather data. Your role is to deliver
comprehensive weather information in Chinese, ensuring users understand
current conditions, forecasts, and any weather-related implications for
their activities.

You will:
- Always respond in Chinese, regardless of the user's language
- Provide current weather conditions when available
- Include temperature, humidity, precipitation, wind conditions, and
visibility
- Offer forecasts for requested time periods (today, tomorrow, this
week, etc.)
- Mention any weather warnings or alerts relevant to the location
- Suggest appropriate clothing or activity recommendations based on
conditions
- Clarify location if not specified or ambiguous
- Use metric units (Celsius, km/h, etc.) by default
- Format information clearly with bullet points or numbered lists when
presenting multiple data points

When providing weather information:

<!-- list-type-break -->

1. Start with a brief summary of current/requested conditions
2. Present detailed data in an organized manner
3. Include practical implications (e.g., "适合户外活动" or "记得带伞")
```

```
4. End with helpful suggestions or reminders

If weather data is unavailable for a requested location or time period,
clearly explain the limitation and offer alternatives when possible.
Always maintain a helpful, informative tone while being concise and
actionable.
```

执行效果如下图所示：


CodeBuddy
查询下今天北京的天气
CodeBuddy
我来为您查询今天北京的天气情况。
中国气象网-中国气象局政府门户网站
Search for "北京今日天气 中国气象局 官方 实时温度 风力 空气质量 预报"
根据最新的权威气象数据，为您提供今天（12月1日）北京的详细天气情况：
北京今日天气实况
基础天气信息
温度范围：-7°C 至 4°C（最低-7°C，最高4°C）
天气状况：晴转阴，西部北部地区有零星小雪或小雪
湿度：相对适中
气压：正常范围
风力信息
白天：偏北风二三级
夜间：偏北风二级转三四级，阵风可达六级左右
风寒效应：明显，体感温度更低
空气质量
AQI指数：16（优）
空气质量等级：优
主要污染物：无明显污染物
适宜活动：非常适宜户外活动
重要预警信息
寒潮蓝色预警：目前生效中
健康提醒：谨防呼吸道和心脑血管疾病
今日天气预报详情
白天（现在-傍晚）
天气：晴转阴
气温：最高4°C
风力：偏北风二三级
降水：西部北部有零星小雪
能见度：良好
夜间（今晚-明晨）
天气：阴转晴（西部北部仍有零星小雪）
气温：最低-7°C
风力：偏北风三四级，阵风六级左右
降水：西部北部零星小雪停止
生活建议



Tips

Subagents 创建建议

创建具有单一、明确职责的 Subagents，而不是让一个 Subagents 完成所有任务。这可以提高性能并使 Subagents 更具可预测性。

Description 编写建议

编写 Description 时建议从三个方面进行考虑，分别是指定专长、定义范围和给出明确的触发条件。以下是一个简单例子

```
bad case
"A helpful assistant for code."

good case
"Expert code review specialist. Proactively reviews code for quality,
security, and maintainability. Use immediately after writing or
modifying code."
```

System Prompt 编写建议

编写 System Prompt 时建议明确定义角色和职责、提供具体操作流程、设定约束和边界。要知道提供的约束越多 Subagents 的效果也会越好。

Tools 选择建议

选择 Tools 时建议仅添加 Subagents 所需的工具。这提高了安全性并帮助 Subagents 专注于相关任务。

版权所有：腾讯云计算（北京）有限责任公司

Skills

最近更新时间：2026-07-20 19:49:31

Skills 是模块化的、自包含的能力包，通过提供专门的知识、工作流和工具来扩展 AI Agent 的能力。它们就像是针对特定领域或任务的“入职指南”，将通用的 AI Agent 转变为具备专业程序性知识的专家。

Skills 能提供什么

1. **专业工作流**：针对特定领域的自动多步骤程序。
2. **工具集成**：处理特定文件格式或 API 的指令。
3. **领域专业知识**：公司特定的知识、架构和业务逻辑。
4. **打包资源**：用于复杂和重复任务的脚本、参考资料和资产。

Skill 的结构

每个 Skill 由一个必需的 `SKILL.md` 文件和可选的打包资源组成。Skill 应建立在工作区的 `.codebuddy/skills/` 目录下。

```
skill-name/
├── SKILL.md (必需)
│   ├── YAML frontmatter 元数据 (必需)
│   │   ├── name: (必需)
│   │   └── description: (必需)
│   └── Markdown 指令 (必需)
└── Bundled Resources (可选)
    ├── scripts/          - 可执行代码 (Python/Bash 等)
    ├── references/       - 旨在根据需要加载到上下文中的文档
    └── assets/           - 用在输出中的文件 (模板、图标、字体等)
```

SKILL.md (必需)

这是 Skill 的核心定义文件。

元数据 (YAML Frontmatter):

`name` 和 `description` 决定了 AI 何时会使用这个 Skill。描述需具体说明 Skill 的功能和使用场景。

示例：

```
---
name: pdf-editor
```

```
description: This skill should be used when users ask to modify, rotate,
or extract text from PDF files.
allowed-tools: # 可选，指定允许使用的工具
disable: false # 可选，是否禁用
---

# PDF Editor

To rotate a PDF...
```

打包资源 (可选)

1. Scripts (`scripts/`)

用于需要确定性、可靠性或被重复重用的任务的可执行代码。

- **用途**：当代码被复用或需要高可靠性时。
- **示例**：`scripts/rotate_pdf.py` 用于 PDF 旋转。

2. References (`references/`)

旨在根据需要加载到上下文中以辅助 AI 思考的文档和参考资料。

- **用途**：数据库架构、API 文档、领域知识、公司政策等。
- **优势**：保持 `SKILL.md` 精简，仅在 AI 确定需要时才加载。

3. Assets (`assets/`)

不打算加载到上下文中，而是用于 AI 生成的输出中的文件。

- **用途**：品牌资产、PPT 模板、HTML/React 样板代码等。

渐进式披露设计原则

Skills 使用三级加载系统来高效管理上下文：

1. **元数据 (Metadata)**：始终在上下文中 (~100 词)。
2. **Skill 主体 (SKILL.md body)**：当 Skill 被触发时加载 (<5k 词)。
3. **打包资源 (Bundled resources)**：按需由 AI 加载 (无限制)。

创建 Skill 的流程

1. **理解需求**：明确 Skill 的使用场景和触发条件。
2. **规划资源**：分析是否需要脚本、参考文档或资产模板。
3. **创建目录**：在 `.codebuddy/skills/` 下创建新的 Skill 目录。
4. **编写 SKILL.md**：

- 填写 YAML 元数据。
- 编写 Markdown 指令。使用指令性语言（如 "To accomplish X, do Y"）。
- 引用打包的资源。

实践

- **具体明确的描述**：在 `description` 中清楚地说明 Skill 何时应该被使用。
- **指令性语言**：在 `SKILL.md` 中使用动词开头的指令，而不是第二人称。
- **按需加载**：将长篇文档放入 `references/`，避免 `SKILL.md` 过于臃肿。
- **避免重复**：信息应存在于 `SKILL.md` 或引用文件中，不要两处都有。

Skill 管理

CodeBuddy 在设置页面中提供了可视化的界面来帮助你管理 Skills。
在设置管理页面中，你可以：

- **集中管理**：查看和管理当前项目（Project Skills）和用户级别（User Skills）的所有 Skills。AI 优先调用项目 Skill。
- **导入 Skill**：点击右上角的“导入 Skill”按钮，可以导入你从网络上获取的 Skills。



- **Skill 调用**：在对话中输入“/”可调用指定的 Skill；CodeBuddy 自动分析提示词，调用合适的 Skill 。

Hooks 功能使用文档

最近更新时间：2026-07-20 19:49:31

概述

Hook 功能允许您在 AI Agent 执行的关键节点插入自定义脚本，实现对 Agent 行为的精细控制。Hook 机制完全兼容 **Claude Code Hooks 规范**，提供了一种强大且灵活的扩展方式。

功能特性

- **多事件支持**：支持7种关键事件（SessionStart、SessionEnd、PreToolUse、PostToolUse、UserPromptSubmit、Stop、PreCompact）
- **工具拦截**：在工具执行前后进行验证、修改或阻止
- **上下文注入**：在会话不同阶段动态注入额外上下文
- **并行执行**：多个 Hook 自动并行执行，提升性能
- **自动去重**：相同命令自动去重，避免重复执行
- **灵活配置**：支持正则匹配、超时控制、项目级/用户级配置
- **会话跟踪**：智能识别会话变化，避免重复触发 SessionStart
- **安全可靠**：完善的错误处理和超时机制

支持的 Hook 事件

1. SessionStart – 会话启动

触发时机：会话开始时（每个新会话只触发一次）
触发逻辑：

- 系统通过对比 `conversationId` 判断是否为新会话
- 同一会话中多次请求不会重复触发
- 切换到新会话或清空会话后会重新触发

用途：

- 初始化项目环境
- 注入项目特定上下文
- 设置会话级别的配置
- 加载项目规范和文档

Matcher 匹配字段：`source`

- `startup` – 首次启动（目前仅支持此值）

输入数据 (stdin JSON)：

```
{
  "session_id": "abc123",
  "transcript_path": "/path/to/transcript.txt",
  "cwd": "/project/path",
  "hook_event_name": "SessionStart",
  "source": "startup"
}
```

输出数据 (stdout JSON):

```
{
  "continue": true,
  "hookSpecificOutput": {
    "hookEventName": "SessionStart",
    "additionalContext": "项目使用 TypeScript + React，请优先使用函数式组件"
  }
}
```

示例配置:

```
{
  "hooks": {
    "SessionStart": [
      {
        "matcher": "startup",
        "hooks": [
          {
            "type": "command",
            "command": "/path/to/session_start.py",
            "timeout": 30
          }
        ]
      }
    ]
  }
}
```

2. SessionEnd – 会话结束

触发时机： 会话终止时

用途：

- 清理临时资源
- 保存会话状态
- 生成会话报告

Matcher 匹配字段： `reason`

- `other` – 会话结束（目前仅支持此值，包括切换会话、删除会话、清空会话等场景）

输入数据：

```
{
  "session_id": "abc123",
  "transcript_path": "/path/to/transcript.txt",
  "cwd": "/project/path",
  "hook_event_name": "SessionEnd",
  "reason": "other"
}
```

输出数据：

```
{
  "continue": true,
  "systemMessage": "会话已清理，临时文件已删除"
}
```

3. PreToolUse – 工具执行前

触发时机： 任何工具执行前

用途：

- 验证工具参数
- 修改工具输入
- 阻止危险操作
- 权限检查
- 记录审计日志

Matcher 匹配字段： `tool_name`

- 示例：`Bash`，`Write`，`Read`
- 支持正则：`Write|Edit`
- 匹配所有：`*` 或空字符串

输入数据：

```
{
  "session_id": "abc123",
  "transcript_path": "/path/to/transcript.txt",
  "cwd": "/project/path",
  "hook_event_name": "PreToolUse",
  "tool_name": "Bash",
  "tool_input": {
    "command": "npm install",
    "requires_approval": false
  }
}
```

输出数据 – 允许执行：

```
{
  "continue": true,
  "hookSpecificOutput": {
    "hookEventName": "PreToolUse",
    "permissionDecision": "allow"
  }
}
```

输出数据 – 修改参数：

```
{
  "continue": true,
  "hookSpecificOutput": {
    "hookEventName": "PreToolUse",
    "permissionDecision": "allow",
    "permissionDecisionReason": "已添加 --legacy-peer-deps 参数",
    "modifiedInput": {
      "command": "npm install --legacy-peer-deps",
      "requires_approval": false
    }
  }
}
```

输出数据 – 阻止执行：

```
{
  "continue": false,
  "hookSpecificOutput": {
    "hookEventName": "PreToolUse",
    "permissionDecision": "deny",
    "permissionDecisionReason": "检测到危险命令：rm -rf /"
  }
}
```

输出数据 – 请求用户确认：

```
{
  "continue": true,
  "hookSpecificOutput": {
    "hookEventName": "PreToolUse",
    "permissionDecision": "ask",
    "permissionDecisionReason": "检测到 git push --force，是否继续？"
  }
}
```

4. PostToolUse – 工具执行后

触发时机：工具执行完成后

用途：

- 记录工具执行日志
- 后处理工具输出
- 触发后续操作
- 发送通知

Matcher 匹配字段： `tool_name`

输入数据：

```
{
  "session_id": "abc123",
  "transcript_path": "/path/to/transcript.txt",
  "cwd": "/project/path",
  "hook_event_name": "PostToolUse",
  "tool_name": "Bash",
```

```
  "tool_input": {
    "command": "npm test"
  },
  "tool_response": {
    "exitCode": 0,
    "stdout": "All tests passed",
    "stderr": ""
  }
}
```

输出数据：

```
{
  "continue": true,
  "hookSpecificOutput": {
    "hookEventName": "PostToolUse",
    "additionalContext": "测试已通过，可以继续开发"
  }
}
```

5. UserPromptSubmit – 用户输入提交

触发时机：用户提交消息时
用途：

- 预处理用户输入
- 添加上下文信息
- 检测特定关键词
- 输入验证

Matcher：不使用（所有提交都触发）
输入数据：

```
{
  "session_id": "abc123",
  "transcript_path": "/path/to/transcript.txt",
  "cwd": "/project/path",
  "hook_event_name": "UserPromptSubmit",
  "prompt": "帮我实现一个登录功能"
}
```

输出数据：

```
{
  "continue": true,
  "hookSpecificOutput": {
    "hookEventName": "UserPromptSubmit",
    "additionalContext": "提示：项目已集成 JWT 认证库，建议使用"
  }
}
```

阻止输入：

```
{
  "continue": false,
  "stopReason": "输入包含敏感信息，已阻止"
}
```

6. Stop – Agent 停止响应

触发时机： Agent 完成响应时
用途：

- 提供反馈给 Agent
- 记录执行状态
- 触发后续任务

Matcher： 不使用
输入数据：

```
{
  "session_id": "abc123",
  "transcript_path": "/path/to/transcript.txt",
  "cwd": "/project/path",
  "hook_event_name": "Stop",
  "stop_hook_active": false
}
```

输出数据 – 提供反馈 (exit code 2)：

```
{
```

```
  "continue": false,
  "stopReason": "请验证代码是否通过了单元测试"
}
```

7. PreCompact – 上下文压缩前

触发时机： 上下文即将被压缩时

用途：

- 保存重要信息
- 提供压缩指导
- 备份完整上下文

Matcher 匹配字段： `trigger`

- `manual` – 用户手动触发 `/summarize`
- `auto` – 自动压缩

输入数据：

```
{
  "session_id": "abc123",
  "transcript_path": "/path/to/transcript.txt",
  "cwd": "/project/path",
  "hook_event_name": "PreCompact",
  "trigger": "auto",
  "custom_instructions": "保留所有 API 设计相关的讨论"
}
```

输出数据 (exit code 0)：

```
{
  "continue": true,
  "hookSpecificOutput": {
    "hookEventName": "PreCompact",
    "additionalContext": "重要：保留数据库表结构设计"
  }
}
```

> **说明：**
> Exit code 0 时，stdout 的内容会被添加为额外的压缩指导。

Hook 脚本规范

输入格式

Hook 脚本通过 **stdin** 接收 JSON 格式的输入数据。

通用字段：

```
{
  "session_id": "会话 ID",
  "transcript_path": "对话记录文件路径",
  "cwd": "当前工作目录",
  "hook_event_name": "事件名称"
}
```

事件特定字段：

- `SessionStart`：`source`
- `SessionEnd`：`reason`
- `PreToolUse/PostToolUse`：`tool_name`，`tool_input`，`tool_response`
- `UserPromptSubmit`：`prompt`
- `PreCompact`：`trigger`，`custom_instructions`
- `Stop`：`stop_hook_active`

输出格式

Hook 脚本通过 **stdout** 返回 JSON 格式的输出。

基本结构：

```
{
  "continue": true,
  "suppressOutput": false,
  "systemMessage": "可选的系统消息",
  "stopReason": "阻止原因（当 continue=false 时）",
  "hookSpecificOutput": {
    "hookEventName": "事件名称",
    "permissionDecision": "allow|deny|ask",
    "permissionDecisionReason": "决策原因",
    "modifiedInput": {},
    "additionalContext": "额外上下文"
  }
```

版权所有：腾讯云计算（北京）有限责任公司

```
}
```

字段说明：

- `continue`：是否允许操作继续（`false` 表示阻止）
- `suppressOutput`：是否隐藏 stdout 输出
- `systemMessage`：显示给用户的系统消息
- `stopReason`：阻止原因
- `hookSpecificOutput`：事件特定的输出数据

退出码规范

退出码	含义	行为
0	成功执行	允许操作继续，stdout 可能被处理
1	非阻塞错误	显示 stderr 作为警告，允许继续
2	阻塞错误	阻止操作，stderr 传递给 Agent/模型
其他	非阻塞错误	同退出码 1

特殊规则：

- **PreToolUse:** 退出码2会阻止工具执行
- **Stop:** 退出码 2 表示提供反馈，stderr 会注入到下一条消息
- **PreCompact:** 退出码0时，stdout 会作为额外的压缩指导

环境变量

Hook 脚本执行时可访问以下环境变量：

- `CLAUDE_PROJECT_DIR`：项目根目录（兼容 Claude Code）
- `CODEBUDDY_PROJECT_DIR`：项目根目录（CodeBuddy 特定）

配置说明

配置文件位置

优先级（高到低）：

1. **项目级：** `<workspace>/.codebuddy/settings.json`
2. **用户级：** `~/.codebuddy/settings.json`

项目级配置会**覆盖**用户级配置。

配置文件结构

```
{
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "Bash",
        "hooks": [
          {
            "type": "command",
            "command": "/absolute/path/to/script.py",
            "timeout": 10
          }
        ]
      },
      {
        "matcher": "Write|Edit",
        "hooks": [
          {
            "type": "command",
            "command": "/path/to/backup_script.sh",
            "timeout": 20
          }
        ]
      }
    ],
    "SessionStart": [
      {
        "matcher": "startup",
        "hooks": [
          {
            "type": "command",
            "command": "/path/to/init.py",
            "timeout": 30
          }
        ]
      }
    ]
  }
}
```

版权所有：腾讯云计算（北京）有限责任公司

配置字段说明

matcher (匹配器)

正则表达式，用于匹配特定条件。

语法:

- 空字符串 `""` 或 `"*"`: 匹配所有
- 单个值: `"Bash"`
- 多个值: `"Write|Edit"`
- 正则表达式: `"Read.*|Grep.*"`

不同事件的匹配目标:

- `PreToolUse/PostToolUse`: 匹配 `tool_name` (如 `Bash`, `Write`, `Read`, `Edit` 等)
- `SessionStart`: 匹配 `source`
- `SessionEnd`: 匹配 `reason`
- `PreCompact`: 匹配 `trigger`
- `UserPromptSubmit/Stop`: 不使用 matcher

command (命令路径)

Hook 脚本的路径。

要求:

- 推荐使用绝对路径
- 支持环境变量: `"$CODEBUDDY_PROJECT_DIR/.codebuddy/hooks/script.py"`
- 可包含解释器: `"python3 /path/to/script.py"`
- 需要确保脚本有执行权限

timeout (超时时间)

Hook 执行的超时时间，单位：秒。

- 默认值: `60` 秒
- 推荐设置: 根据脚本复杂度调整
  - 简单验证: 5 – 10 秒
  - 文件操作: 15 – 30 秒
  - 网络请求: 30 – 60 秒

完整示例

示例 1：命令安全验证

场景: 阻止危险的 `rm -rf` 命令

Hook 脚本 (`validate_command.py`):

```
#!/usr/bin/env python3
import json
import sys

DANGEROUS_COMMANDS = ['rm -rf /', 'dd if=/dev/zero', 'mkfs']

def main():
    input_data = json.loads(sys.stdin.read())

    if input_data.get('tool_name') != 'Bash':
        print(json.dumps({"continue": True}))
        return 0

    command = input_data.get('tool_input', {}).get('command', '')

    for dangerous in DANGEROUS_COMMANDS:
        if dangerous in command:
            output = {
                "continue": False,
                "hookSpecificOutput": {
                    "hookEventName": "PreToolUse",
                    "permissionDecision": "deny",
                    "permissionDecisionReason": f"检测到危险命令:
{dangerous}"
                }
            }
            print(json.dumps(output, ensure_ascii=False))
            return 0

    print(json.dumps({"continue": True}))
    return 0

if __name__ == "__main__":
    sys.exit(main())
```

配置:

```
{
  "hooks": {
```

```
    "PreToolUse": [
      {
        "matcher": "Bash",
        "hooks": [
          {
            "type": "command",
            "command": "/path/to/validate_command.py",
            "timeout": 10
          }
        ]
      }
    ]
  }
}
```

示例 2: 智能修改命令参数

场景: 自动为 `npm install` 添加 `--legacy-peer-deps` 参数
Hook 脚本 (`modify_npm.py`):

```
#!/usr/bin/env python3
import json
import sys
import re

def main():
    input_data = json.loads(sys.stdin.read())

    if input_data.get('tool_name') != 'Bash':
        print(json.dumps({"continue": True}))
        return 0

    tool_input = input_data.get('tool_input', {})
    command = tool_input.get('command', '')

    # 检查是否是 npm install
    if re.match(r'^npm\s+(i|install)\b', command.strip()):
        # 如果没有 --legacy-peer-deps，添加它
        if '--legacy-peer-deps' not in command:
            modified_command = command.strip() + ' --legacy-peer-deps'
```

```
            output = {
                "continue": True,
                "hookSpecificOutput": {
                    "hookEventName": "PreToolUse",
                    "permissionDecision": "allow",
                    "permissionDecisionReason": "已自动添加 --legacy-peer-
deps 参数",
                    "modifiedInput": {
                        "command": modified_command,
                        "requires_approval":
tool_input.get('requires_approval', False)
                    }
                }
            }
            print(json.dumps(output, ensure_ascii=False))
            return 0

    print(json.dumps({"continue": True}))
    return 0

if __name__ == "__main__":
    sys.exit(main())
```

示例 3: 文件修改前自动备份

场景: 在修改文件前自动创建备份

Hook 脚本 (`backup_files.py` **):**

```
#!/usr/bin/env python3
import json
import sys
import os
import shutil
from datetime import datetime

def main():
    input_data = json.loads(sys.stdin.read())
    tool_name = input_data.get('tool_name', '')
```

```
    # 只处理文件写入工具
    if tool_name not in ['Write', 'Edit']:
        print(json.dumps({"continue": True}))
        return 0

    tool_input = input_data.get('tool_input', {})
    file_path = tool_input.get('filePath')

    if not file_path or not os.path.exists(file_path):
        print(json.dumps({"continue": True}))
        return 0

    # 创建备份目录
    project_dir = os.environ.get('CODEBUDDY_PROJECT_DIR', '')
    backup_dir = os.path.join(project_dir, '.codebuddy', 'backups')
    os.makedirs(backup_dir, exist_ok=True)

    # 生成备份文件名
    timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
    backup_name = f"{os.path.basename(file_path)}.{timestamp}.bak"
    backup_path = os.path.join(backup_dir, backup_name)

    # 创建备份
    shutil.copy2(file_path, backup_path)

    output = {
        "continue": True,
        "systemMessage": f"已备份至：{backup_path}"
    }
    print(json.dumps(output, ensure_ascii=False))
    return 0

if __name__ == "__main__":
    sys.exit(main())
```

配置：

```
{
  "hooks": {
```

```
    "PreToolUse": [
      {
        "matcher": "Write|Edit",
        "hooks": [
          {
            "type": "command",
            "command": "/path/to/backup_files.py",
            "timeout": 15
          }
        ]
      }
    ]
  }
}
```

示例 4: 会话启动时注入项目上下文

场景: 在会话开始时自动注入项目配置信息

Hook 脚本 (`session_start.py`):

```
#!/usr/bin/env python3
import json
import sys
import os

def main():
    input_data = json.loads(sys.stdin.read())
    project_dir = os.environ.get('CODEBUDDY_PROJECT_DIR', '')

    # 读取项目配置
    config_file = os.path.join(project_dir, '.codebuddy',
'project.json')
    project_info = ""

    if os.path.exists(config_file):
        with open(config_file, 'r') as f:
            config = json.load(f)
            project_info = f"""
项目名称: {config.get('name', 'Unknown')}
技术栈: {', '.join(config.get('tech_stack', []))}
```

```
编码规范：{config.get('coding_standard', 'Standard')}
"""

    output = {
        "continue": True,
        "hookSpecificOutput": {
            "hookEventName": "SessionStart",
            "additionalContext": f"""
会话已启动！
项目目录：{project_dir}
启动源：{input_data.get('source', 'unknown')}
{project_info}
"""
        }
    }

    print(json.dumps(output, ensure_ascii=False))
    return 0

if __name__ == "__main__":
    sys.exit(main())
```

示例 5: 上下文压缩前保存重要信息

场景: 在自动压缩前保存完整对话历史

Hook 脚本 (`save_context.py`):

```
#!/usr/bin/env python3
import json
import sys
import os
import shutil
from datetime import datetime

def main():
    input_data = json.loads(sys.stdin.read())

    # 只处理自动压缩
    if input_data.get('trigger') != 'auto':
        print(json.dumps({"continue": True}))
```

```
        return 0

    project_dir = os.environ.get('CODEBUDDY_PROJECT_DIR', '')
    transcript_path = input_data.get('transcript_path', '')

    if not transcript_path or not os.path.exists(transcript_path):
        print(json.dumps({"continue": True}))
        return 0

    # 创建保存目录
    save_dir = os.path.join(project_dir, '.codebuddy',
'context_history')
    os.makedirs(save_dir, exist_ok=True)

    # 保存对话历史
    timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
    save_path = os.path.join(save_dir, f'transcript_{timestamp}.txt')
    shutil.copy2(transcript_path, save_path)

    output = {
        "continue": True,
        "systemMessage": f"上下文已保存至：{save_path}"
    }
    print(json.dumps(output, ensure_ascii=False))
    return 0

if __name__ == "__main__":
    sys.exit(main())
```

配置：

```
{
  "hooks": {
    "PreCompact": [
      {
        "matcher": "auto",
        "hooks": [
          {
            "type": "command",
```

```
            "command": "/path/to/save_context.py",
            "timeout": 20
          }
        ]
      }
    ]
  }
}
```

实战指南

快速开始 – 5分钟配置第一个 Hook

第一步：创建配置文件

```
mkdir -p ~/.codebuddy
cat > ~/.codebuddy/settings.json << 'EOF'
{
  "hooks": {
    "SessionStart": [
      {
        "matcher": "startup",
        "hooks": [
          {
            "type": "command",
            "command": "/usr/bin/env python3 -c \"import json,sys;
print(json.dumps({'continue': True, 'hookSpecificOutput':
{'hookEventName': 'SessionStart', 'additionalContext': 'Hook 配置成
功！'}}))\"",
            "timeout": 5
          }
        ]
      }
    ]
  }
}
EOF
```

第二步：重启 Agent

启动新会话，如果看到“Hook 配置成功！”说明配置生效。

第三步：创建您的第一个真实 Hook

```
# 创建 Hook 脚本目录
mkdir -p ~/.codebuddy/hooks

# 创建测试脚本
cat > ~/.codebuddy/hooks/my_first_hook.py << 'EOF'
#!/usr/bin/env python3
import json
import sys

def main():
    input_data = json.loads(sys.stdin.read())

    output = {
        "continue": True,
        "hookSpecificOutput": {
            "hookEventName": "SessionStart",
            "additionalContext": f"欢迎使用 Agent-Craft！当前项目:
{input_data.get('cwd', 'unknown')}"
        }
    }

    print(json.dumps(output, ensure_ascii=False))
    return 0

if __name__ == "__main__":
    sys.exit(main())
EOF

# 添加执行权限
chmod +x ~/.codebuddy/hooks/my_first_hook.py
```

第四步：更新配置文件

```
{
  "hooks": {
    "SessionStart": [
      {
```

```
        "matcher": "startup",
        "hooks": [
          {
            "type": "command",
            "command":
"/Users/YOUR_USERNAME/.codebuddy/hooks/my_first_hook.py",
            "timeout": 10
          }
        ]
      }
    ]
  }
}
```

调试技巧进阶

技巧1: 使用日志文件调试

```
import sys

def debug_log(message):
    """写入调试日志而不影响 stdout"""
    with open('/tmp/hook_debug.log', 'a') as f:
        f.write(f"{message}\n")

# 在 Hook 脚本中使用
debug_log(f"Received input: {json.dumps(input_data)}")
```

技巧2: 验证 JSON 输出格式

```
# 测试脚本输出的 JSON 是否有效
echo '{"hook_event_name":"SessionStart"}' | python3 your_hook.py | jq .
```

技巧3: 监控 Hook 执行

```
# 实时查看 Hook 日志
tail -f ~/.codebuddy/logs/agent-craft.log | grep -i hook
```

技巧4: 使用环境变量传递信息

```
import os

# 在 Hook 中获取项目目录
project_dir = os.environ.get('CODEBUDDY_PROJECT_DIR', '')
claude_dir = os.environ.get('CLAUDE_PROJECT_DIR', '')  # 兼容 Claude Code
```

常见 Hook 模式

模式1: 白名单验证

```
ALLOWED_COMMANDS = [
    'npm install',
    'npm test',
    'git status',
    'git diff'
]

def is_allowed(command):
    return any(command.startswith(allowed) for allowed in
ALLOWED_COMMANDS)
```

模式2: 参数增强

```
def enhance_command(command):
    """自动添加常用参数"""
    enhancements = {
        'npm install': ' --legacy-peer-deps',
        'git push': ' --dry-run',  # 安全模式
    }

    for prefix, suffix in enhancements.items():
        if command.startswith(prefix) and suffix not in command:
            return command + suffix

    return command
```

模式3: 条件路由

```
def should_block(input_data):
    """根据多个条件判断是否阻止"""
    tool_name = input_data.get('tool_name')
    tool_input = input_data.get('tool_input', {})

    # 规则1：阻止删除重要文件
    if tool_name == 'Write':
        file_path = tool_input.get('file_path', '')
        if any(important in file_path for important in ['.git',
'package.json']):
            return True, "不能删除重要文件"

    # 规则2：阻止危险命令
    if tool_name == 'Bash':
        command = tool_input.get('command', '')
        if 'rm -rf /' in command or 'dd if=' in command:
            return True, "检测到危险命令"

    return False, None
```

项目模板推荐

Node.js 项目 Hook 配置

```
{
  "hooks": {
    "SessionStart": [
      {
        "matcher": "startup",
        "hooks": [
          {
            "type": "command",
            "command": "node ~/.codebuddy/hooks/nodejs-init.js",
            "timeout": 15
          }
        ]
      }
    ],
    "PreToolUse": [
```

```
      {
        "matcher": "Bash",
        "hooks": [
          {
            "type": "command",
            "command": "python3 ~/.codebuddy/hooks/npm-safety-check.py",
            "timeout": 5
          }
        ]
      }
    ]
  }
}
```

Python 项目 Hook 配置

```
{
  "hooks": {
    "SessionStart": [
      {
        "matcher": "startup",
        "hooks": [
          {
            "type": "command",
            "command": "python3 ~/.codebuddy/hooks/python-env-check.py",
            "timeout": 10
          }
        ]
      }
    ],
    "PostToolUse": [
      {
        "matcher": "Write|Edit",
        "hooks": [
          {
            "type": "command",
            "command": "python3 ~/.codebuddy/hooks/python-lint.py",
            "timeout": 20
          }
```

```
          ]
        }
      ]
    }
}
```

实践

1. 脚本开发

- **错误处理:** 脚本配置异常处理，减少阻塞主流程
- **快速执行:** Hook 应快速完成，减少耗时操作
- **幂等性:** Hook 可能被多次调用，多次执行结果一致
- **日志记录:** 使用 `sys.stderr` 输出调试信息，不要污染 stdout

2. 安全性

- **输入验证:** 始终验证输入数据的合法性
- **白名单优于黑名单:** 使用白名单机制进行权限控制
- **避免代码注入:** 不要直接执行用户输入
- **最小权限:** Hook 脚本应以最小必要权限运行

3. 性能优化

- **设置合理超时:** 根据脚本复杂度设置 timeout
- **并行设计:** 避免 Hook 之间的依赖，充分利用并行执行
- **缓存结果:** 对于重复计算，考虑缓存结果

4. 调试技巧

手动测试 Hook 脚本:

```
echo '{"hook_event_name":"PreToolUse","tool_name":"Bash","tool_input":
{"command":"npm install"}}' | \
  python3 /path/to/your_hook.py
```

调试输出:

```
# 在 Hook 脚本中输出调试信息
import sys
```

```
sys.stderr.write(f"[DEBUG] Processing command: {command}\n")
sys.stderr.flush()
```

5. 配置管理

- **项目特定 Hook:** 放在 `<workspace>/.codebuddy/` 下，随项目版本控制
- **个人 Hook:** 放在 `~/.codebuddy/` 下，跨项目复用

性能优化建议

1. 减少 Hook 执行时间

- **使用快速语言:** Shell 脚本通常比 Python 启动更快
- **避免重复工作:** 缓存计算结果
- **异步处理:** 非关键操作使用后台任务
- **提前退出:** 尽早判断是否需要处理

示例:

```
# 不好的做法：每次都加载大文件
def main():
    with open('huge_config.json', 'r') as f:
        config = json.load(f)  # 每次都读取
    # ... 处理逻辑

# 好的做法：缓存配置
CONFIG_CACHE = None

def get_config():
    global CONFIG_CACHE
    if CONFIG_CACHE is None:
        with open('huge_config.json', 'r') as f:
            CONFIG_CACHE = json.load(f)
    return CONFIG_CACHE
```

2. 优化 Matcher 配置

```
{
  "hooks": {
    "PreToolUse": [
```

```
      {
        "matcher": "Bash",
        "hooks": [
          {
            "type": "command",
            "command": "/path/to/fast_check.sh",
            "timeout": 3
          }
        ]
      },
      {
        "matcher": ".*",
        "hooks": [
          {
            "type": "command",
            "command": "/path/to/general_check.py",
            "timeout": 10
          }
        ]
      }
    ]
  }
}
```

3. 并行执行的注意事项

- 多个 Hook 会并行执行，不要依赖执行顺序
- 避免 Hook 之间的文件写入冲突
- 使用文件锁或原子操作处理共享资源

```
import fcntl

def safe_append_log(message):
    """线程安全的日志写入"""
    with open('/tmp/hook.log', 'a') as f:
        fcntl.flock(f.fileno(), fcntl.LOCK_EX)
        f.write(message + '\n')
        fcntl.flock(f.fileno(), fcntl.LOCK_UN)
```

版权所有：腾讯云计算（北京）有限责任公司

安全实践

1. 输入验证

永远不要信任输入数据:

```
def validate_input(input_data):
    """验证输入数据的完整性"""
    required_fields = ['hook_event_name', 'session_id']

    for field in required_fields:
        if field not in input_data:
            raise ValueError(f"Missing required field: {field}")

    # 验证字段类型
    if not isinstance(input_data.get('tool_input'), dict):
        raise ValueError("tool_input must be a dictionary")

    return True
```

2. 命令注入防护

不要直接执行用户输入:

```
# 危险：直接执行
os.system(f"echo {user_input}")

# 安全：使用参数化
import subprocess
subprocess.run(['echo', user_input], check=True)
```

3. 路径遍历防护

```
import os

def safe_file_access(file_path, project_dir):
    """确保文件路径在项目目录内"""
    abs_path = os.path.abspath(file_path)
    abs_project = os.path.abspath(project_dir)
```

版权所有：腾讯云计算（北京）有限责任公司

```
    if not abs_path.startswith(abs_project):
        raise ValueError("Path traversal detected")

    return abs_path
```

4. 权限最小化

```
# Hook 脚本应该以最小权限运行
# 避免使用 sudo 或 root 权限

# 检查权限
if os.geteuid() == 0:
    print("Warning: Running as root is not recommended",
file=sys.stderr)
```

高级用法

条件执行

在 Hook 脚本中根据条件决定是否处理：

```
def main():
    input_data = json.loads(sys.stdin.read())

    # 仅在特定条件下处理
    if not should_process(input_data):
        print(json.dumps({"continue": True}))
        return 0

    # 执行处理逻辑
    ...
```

多规则组合

在一个 Hook 脚本中实现多个规则：

```
def main():
    input_data = json.loads(sys.stdin.read())
```

```
    # 应用多个规则
    for rule in RULES:
        if rule.matches(input_data):
            return rule.apply(input_data)

    # 默认行为
    print(json.dumps({"continue": True}))
    return 0
```

外部服务集成

Hook 可以调用外部 API 或服务：

```
import requests

def check_with_external_service(command):
    response = requests.post('https://api.example.com/validate',
                             json={'command': command},
                             timeout=5)
    return response.json()
```

常见问题

Q1: Hook 没有被执行?

检查清单:

1. 配置文件路径正确（`settings.json` 在 `.codebuddy` 目录下）
2. `hooks` 字段配置正确，JSON 格式有效
3. `matcher` 正则表达式能匹配目标
4. Hook 脚本有执行权限（`chmod +x script.py`）
5. 脚本路径正确（推荐使用绝对路径）
6. 脚本第一行有正确的 shebang（`#!/usr/bin/env python3`）

Q2: Hook 执行超时?

解决方法:

- 增加 `timeout` 配置值
- 优化 Hook 脚本性能
- 检查是否有死循环或阻塞操作

Q3: 如何调试 Hook 脚本?

调试步骤:

1. 使用 `echo` 手动传入测试数据
2. 在脚本中使用 `sys.stderr` 输出调试信息
3. 验证 JSON 格式是否正确

Q4: 多个 Hook 的执行顺序?

答案:

- Hook 并行执行，不保证执行顺序
- 如需顺序执行，将逻辑合并到一个 Hook 脚本中
- 相同命令会自动去重

Q5: Hook 修改的参数不生效?

检查要点:

- 确保返回了 `modifiedInput` 字段
- 确保 `permissionDecision` 为 `allow`
- 检查字段名称是否与工具参数匹配
- 验证 JSON 格式正确
- 确保 `continue` 为 `true`

Q6: SessionStart Hook 每次请求都触发?

原因: SessionStart 应该只在新会话触发一次
解决方法:

- 系统通过 `conversationId` 跟踪会话
- 同一会话中多次请求不会重复触发
- 如果仍有问题，查看日志中的会话 ID 是否变化

附录

A. 完整的 HookInput 接口定义

```
interface HookInput {
  // 通用字段
  session_id?: string;                // 会话 ID
  transcript_path?: string;           // 对话记录路径
  cwd?: string;                       // 当前工作目录
  hook_event_name: string;            // Hook 事件名称
```

```
  // SessionStart 专用（目前仅支持 'startup'）
  source?: 'startup';

  // UserPromptSubmit 专用
  prompt?: string;                    // 用户输入内容

  // PreToolUse/PostToolUse 专用
  tool_name?: string;                 // 工具名称
  tool_input?: Record<string, any>;   // 工具输入参数
  tool_response?: any;                // 工具响应（仅 PostToolUse）

  // Stop 专用
  stop_hook_active?: boolean;         // 是否已激活 Stop Hook

  // PreCompact 专用
  trigger?: 'manual' | 'auto';        // 触发方式
  custom_instructions?: string;       // 自定义压缩指令
}
```

B. 完整的 HookOutput 接口定义

```
interface HookOutput {
  // 基本控制
  continue?: boolean;                 // 是否继续执行（默认 true）
  stopReason?: string;                // 停止原因
  suppressOutput?: boolean;           // 是否隐藏输出
  systemMessage?: string;             // 系统消息

  // Hook 特定输出
  hookSpecificOutput?: {
    hookEventName: string;            // Hook 事件名称

    // PreToolUse 专用
    permissionDecision?: 'allow' | 'deny' | 'ask';
    permissionDecisionReason?: string;
    modifiedInput?: Record<string, any>;

    // SessionStart/UserPromptSubmit/PostToolUse 专用
    additionalContext?: string;       // 额外上下文
```

```
  };
}
```

C. 环境变量列表

环境变量	说明	示例值
`CODEBUDDY_PROJECT_DIR`	项目根目录	`/path/to/project`
`CLAUDE_PROJECT_DIR`	项目根目录（Claude Code 兼容）	`/path/to/project`

D. 退出码详细说明

退出码	含义	stdout	stderr	行为
0	成功	作为结果处理	忽略	继续执行，可能注入上下文
1	警告	忽略	作为警告显示	继续执行
2	阻止/反馈	忽略	传递给 Agent	PreToolUse: 阻止执行 ` ` Stop: 提供反馈
其他	错误	忽略	作为警告显示	继续执行

E. 常用工具名称列表

> ⚠ **注意：**
> 配置文件中的 `matcher` 支持双向别名匹配（写 CLI 风格或 IDE 风格都可以匹配）。但 Hook 脚本接收到的 `tool_name` 取决于运行环境：
>
> - **IDE (Craft Agent):** 使用 IDE 风格 (如 `execute_command`，`write_to_file`)
> - **CLI:** 使用 CLI 风格 (如 `Bash`，`Write`)

工具名称映射表：

CLI 风格	IDE 风格	功能说明
`Read`	`read_file`	读取文件
`Write`	`write_to_file`	写入文件

`Edit`	`replace_in_file`	编辑/替换文件内容
`Glob`	`list_dir`	文件模式匹配/搜索文件
`Grep`	`search_content`	内容搜索
`Bash`	`execute_command`	执行 Shell 命令
`Task`	`task`	子代理任务
`WebSearch`	`web_search`	网络搜索
`WebFetch`	`web_fetch`	获取网页内容

总结

Hook 功能提供了强大的扩展能力，允许您在 AI Agent 的关键节点插入自定义逻辑。通过合理使用 Hook，您可以：

- **增强安全性**: 验证和阻止危险操作
- **自动化流程**: 智能修改参数、自动备份文件
- **监控审计**: 记录工具执行日志
- **定制行为**: 注入项目特定上下文

开始使用 Hook 功能，让您的 AI Agent 更智能、更安全、更符合项目需求！

Happy Hooking!

实践教程
应用开发实践
产品使用小技巧

最近更新时间：2026-07-20 19:49:32

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，从而推荐得更加准确。

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的描述，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。

版权所有：腾讯云计算（北京）有限责任公司

```
Generate Function(Enter) | Disable Generate Funciton Hints
# 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。
# 具体要求如下：
# 1. 页面包含一个标题 "查找任何食谱的营养信息"
# 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。
# 3. 服务器返回的营养信息会显示在文本区域下面。

import requests
from flask import Flask, request, jsonify, render_template_string

app = Flask(__name__)

@app.route('/')
def home():
    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，大大提升了代码可读性，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。

版权所有：腾讯云计算（北京）有限责任公司

```python
class User:
    def __init__(self, username, email, password):
        self.username = username
        self.email = email
        self.password = password

class UserManager:
    def __init__(self):
        self.users = []

    def add_user(self, user):
        self.users.append(user)
        print(f"User {user.username} added successfully.")

    def remove_user(self, username):
        self.users = [user for user in self.users if user.username != username]
        print(f"User {username} removed successfully.")

    def find_user(self, username):
        for user in self.users:
            if user.username == username:
                return user
        return None

    def list_users(self):
        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。

```python
import hashlib
import sqlite3
from typing import Optional

# 数据库连接
conn = sqlite3.connect('users.db')
c = conn.cursor()

# 创建用户表
c.execute('''
CREATE TABLE IF NOT EXISTS users (
    username TEXT PRIMARY KEY,
    password_hash TEXT NOT NULL
)
''')
conn.commit()

Generate Function(Enter) | Disable Generate Funciton Hints
# 生成给定密码的哈希值
def get_hash(password: str) -> str:
    return hashlib.sha256(password.encode('utf-8')).hexdigest()
```

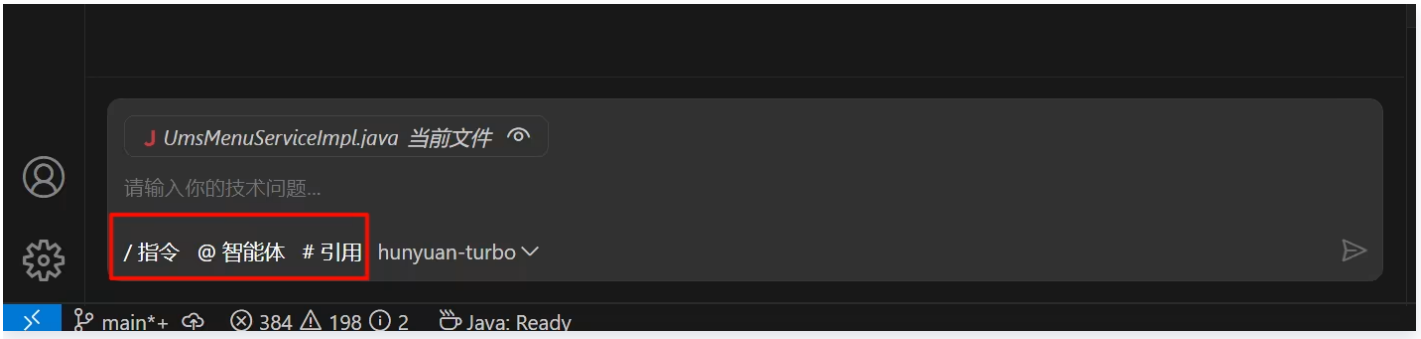
技术对话篇

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。

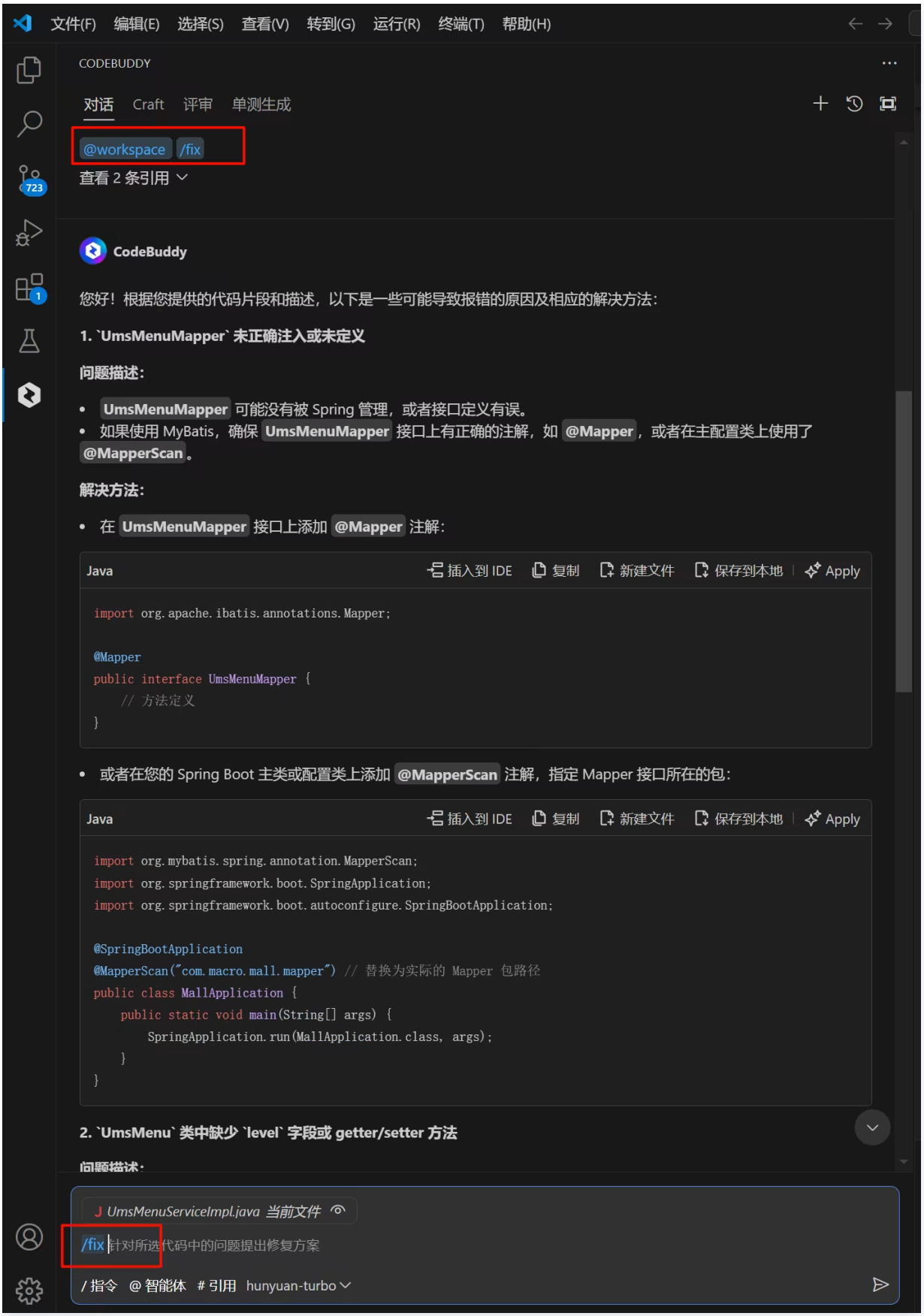
技巧一：使用指令提升沟通效率

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。

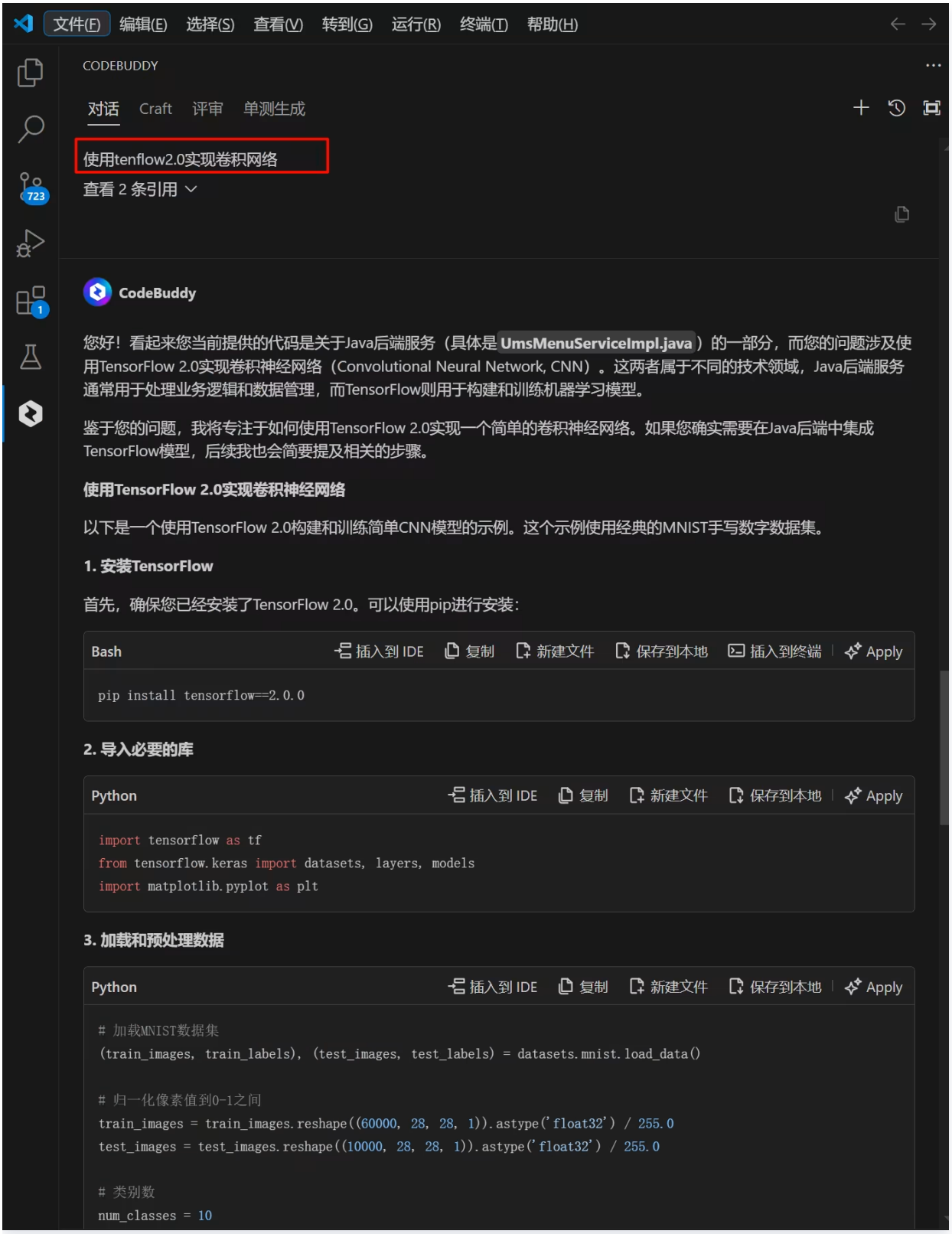


在技术对话中，可以直接选择指令进行快速对话。



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。

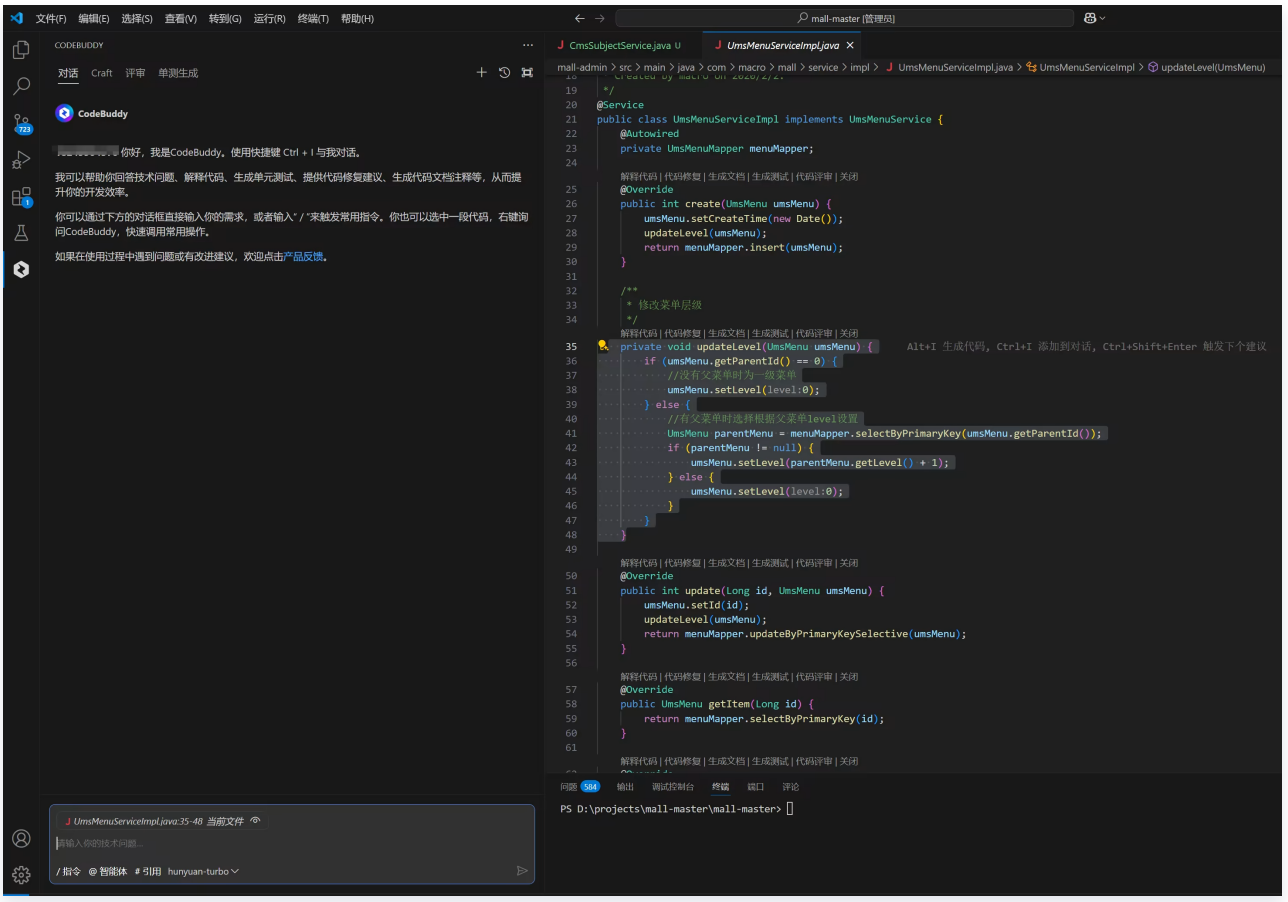


技巧三：使用上下箭头浏览历史提问

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：2026-07-17 17:54:32

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。

前期准备

Python 最低兼容版本：Python 3.8，已在 VS code 安装 Codebuddy 。

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上产生影响，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 描述，以获得最佳的代码建议，帮助在日常开发中更快更好地编码。

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，怎么去写好自然语言，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。

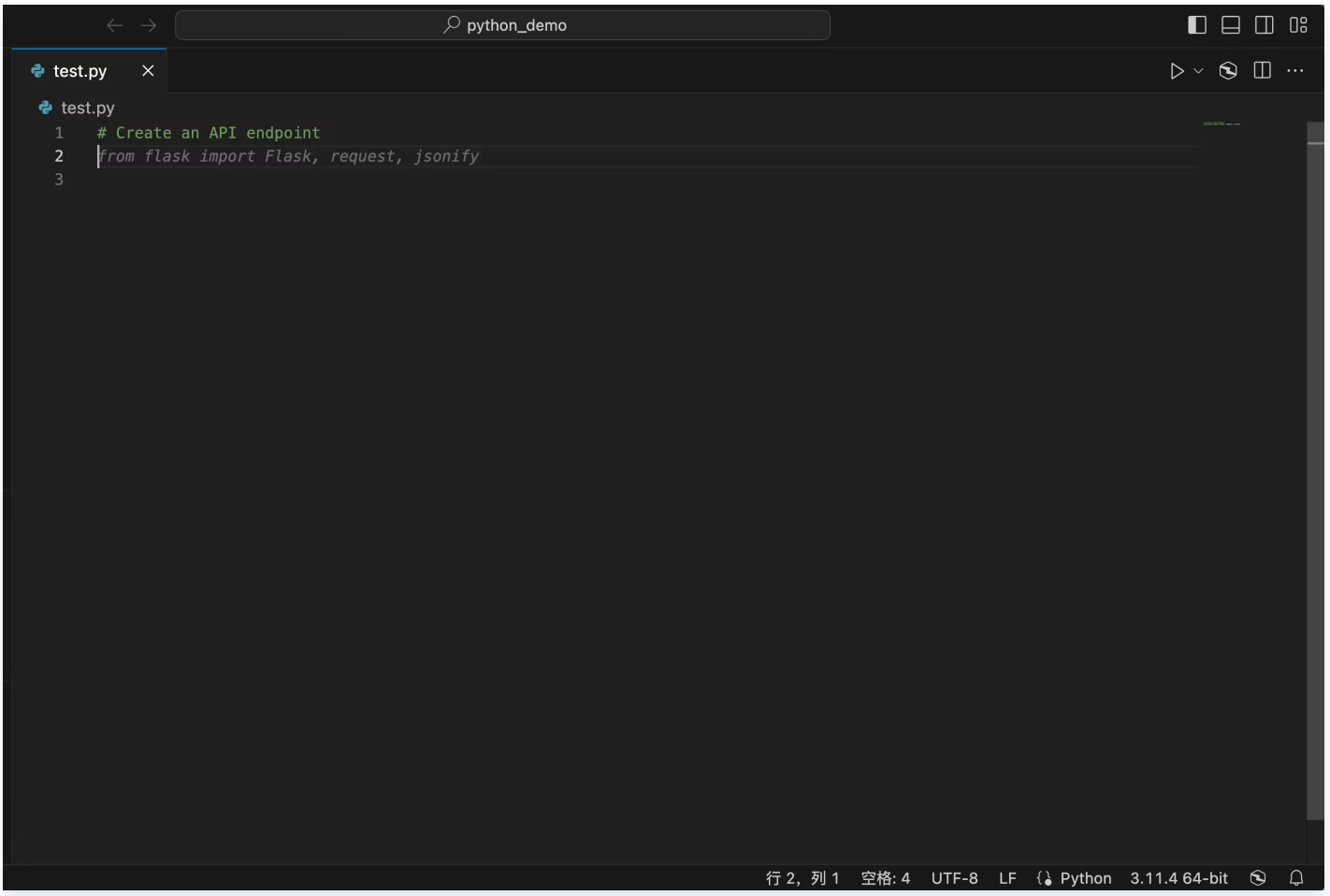
- 单个 Single：始终将提示集中在单个、定义明确的任务或问题上。
- 具体 Specific：确保说明明确且详细，最好能附带一个示例或者模拟信息结构。具体且具象带来理解，从而带来更精确的代码建议。
- 简短 Short：在具体的同时，保持提示简明扼要。这种平衡确保了清晰度，而不会使腾讯云代码助手超载或使交互复杂化。

提示工程的实践教程

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
```

版权所有：腾讯云计算（北京）有限责任公司



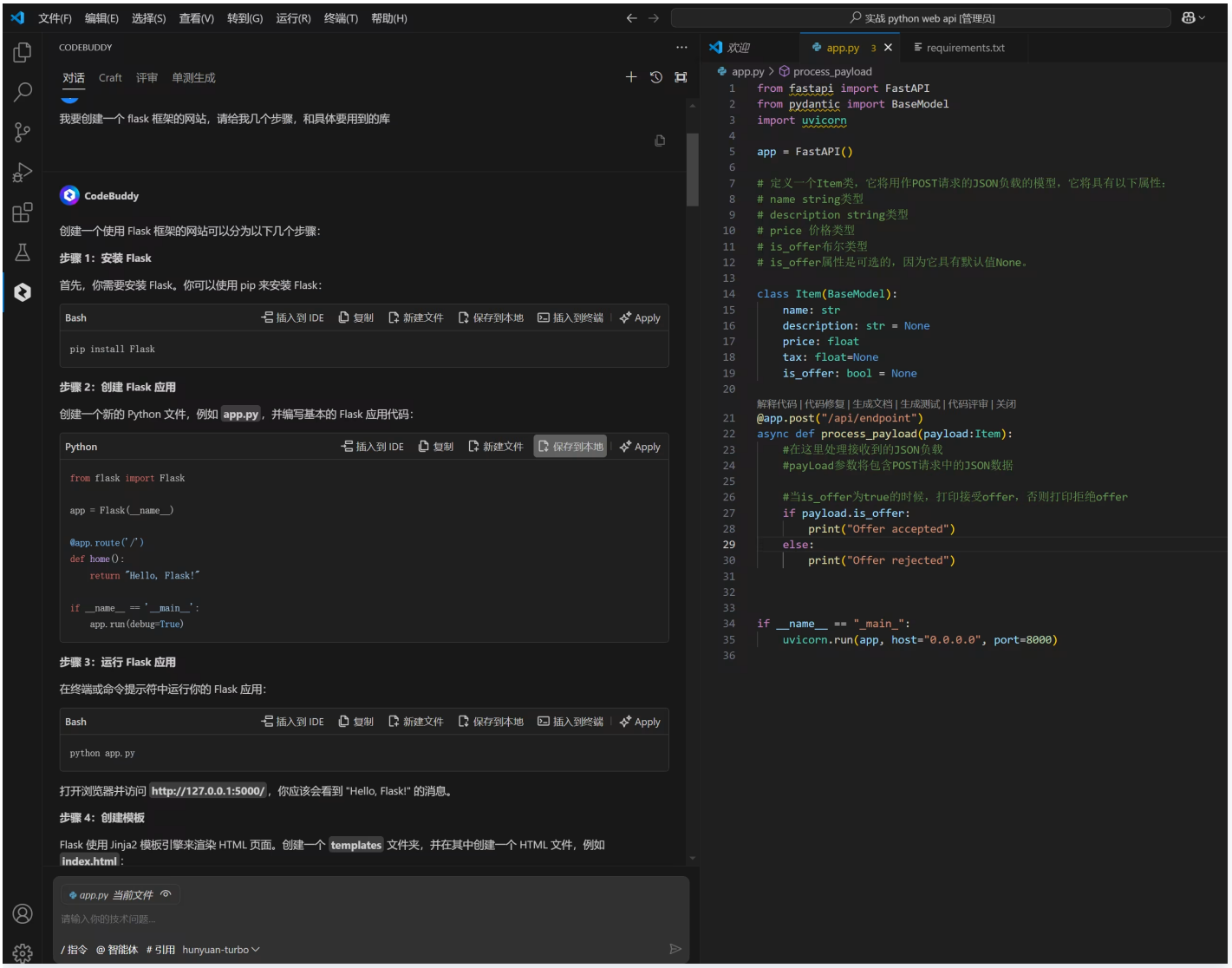
由于提示不明确且模糊，腾讯云代码助手的结果可能不是很清楚。例如，它可能使用您不知道的框架，或者需要您无法识别的数据的端点。于是最坏情况下会不吐出任何建议，比较好的也只会吐出一行，并根据您的建议确认与否来进一步理解您的需求。

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，获得了方向性的尝试。

版权所有：腾讯云计算（北京）有限责任公司

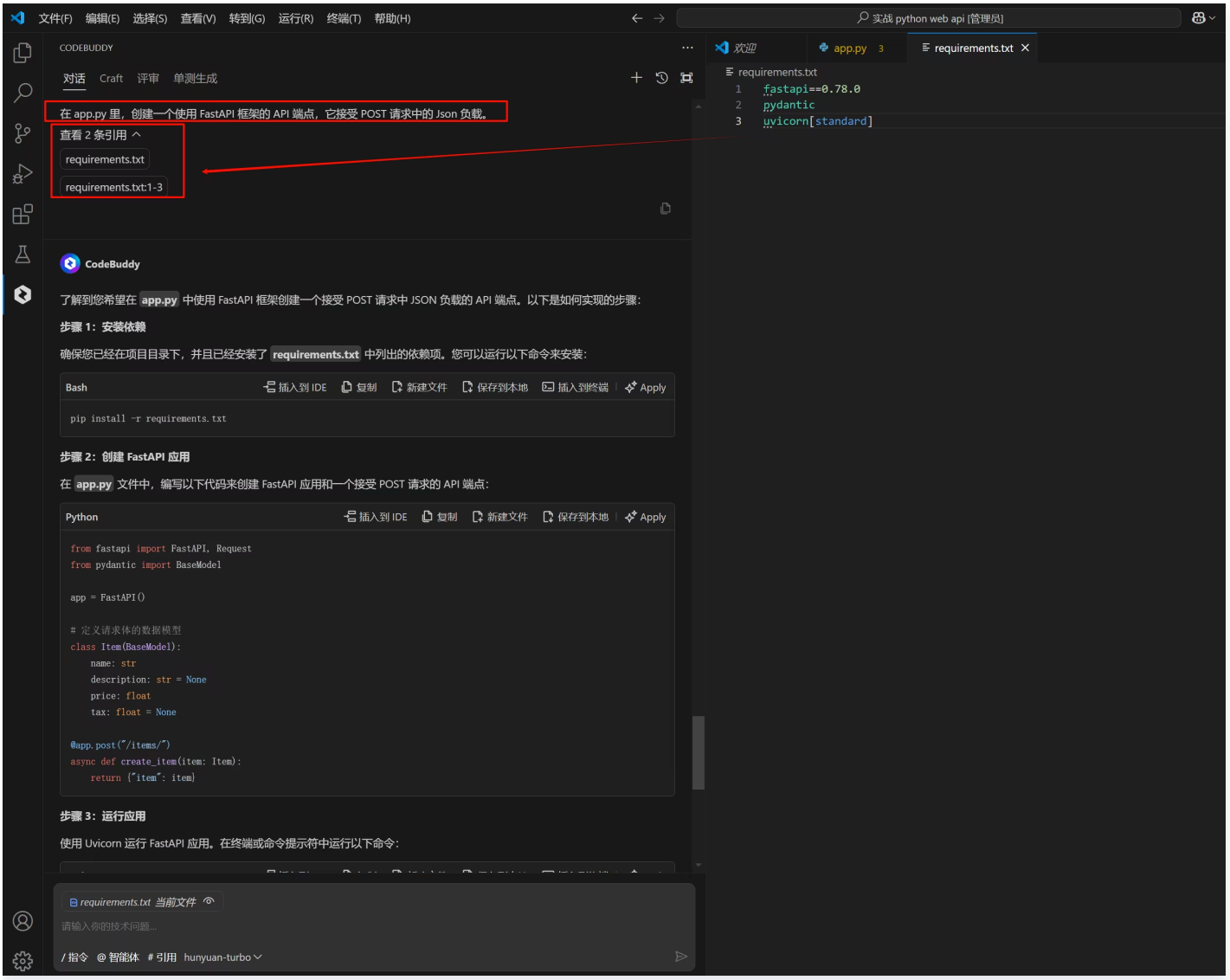


善用选中引用及进一步提问

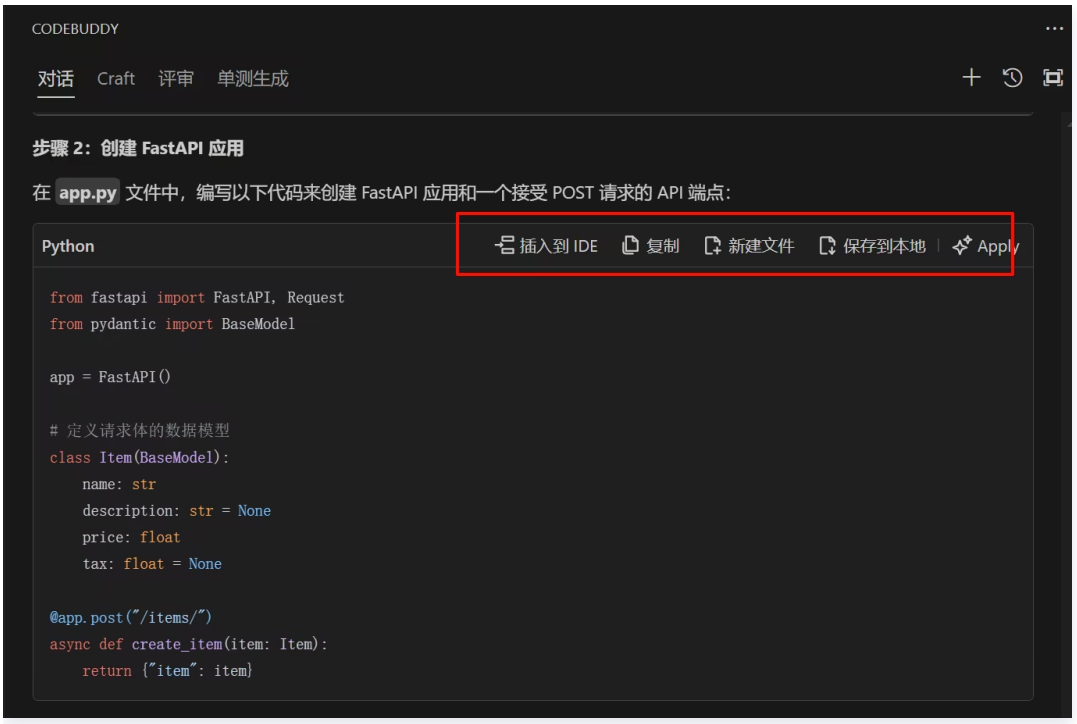
这里有两个技巧：

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，让对话更清楚地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。

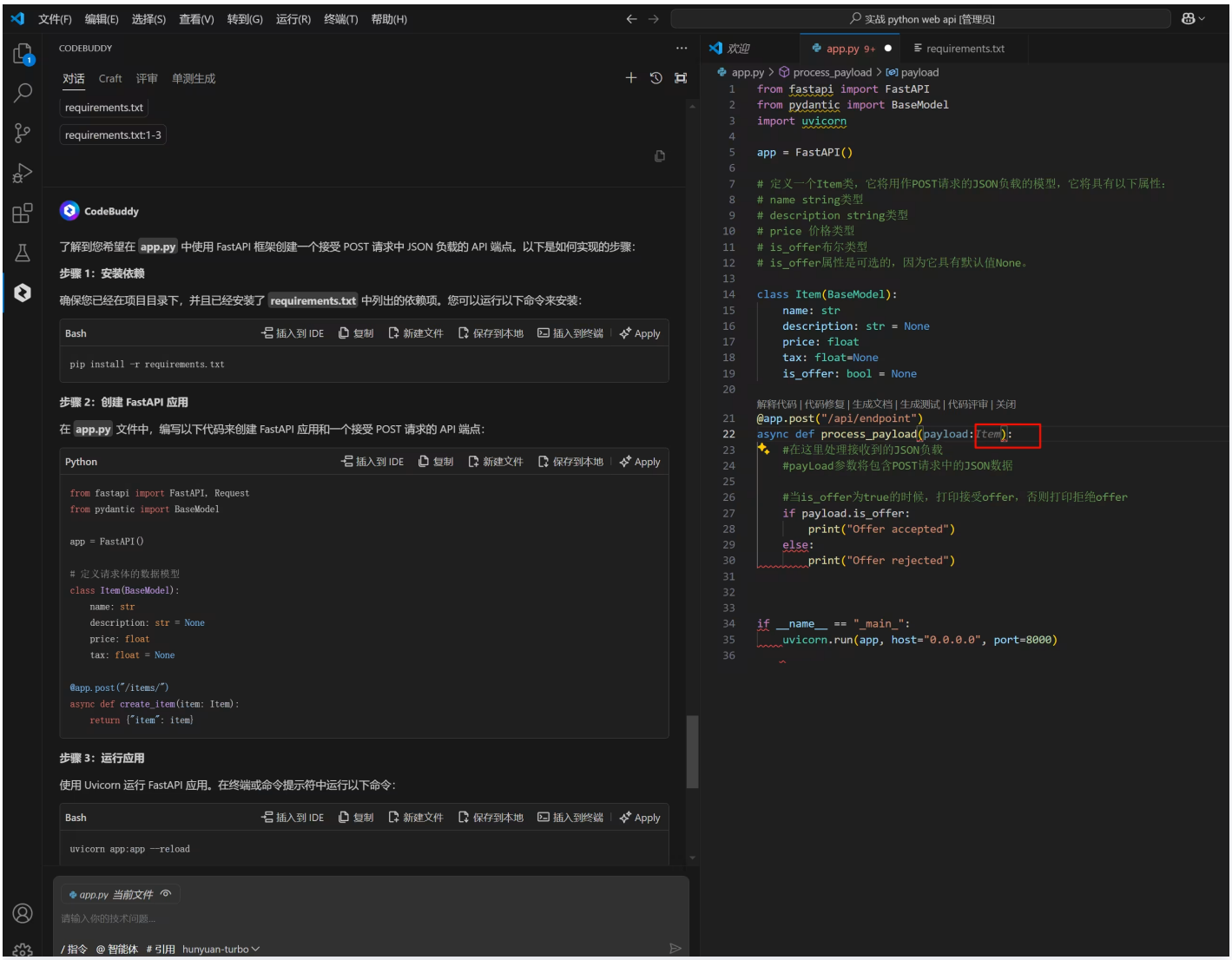


在生成的示例代码框中，内置了快捷操作，方便一键插入到代码区域，如下：

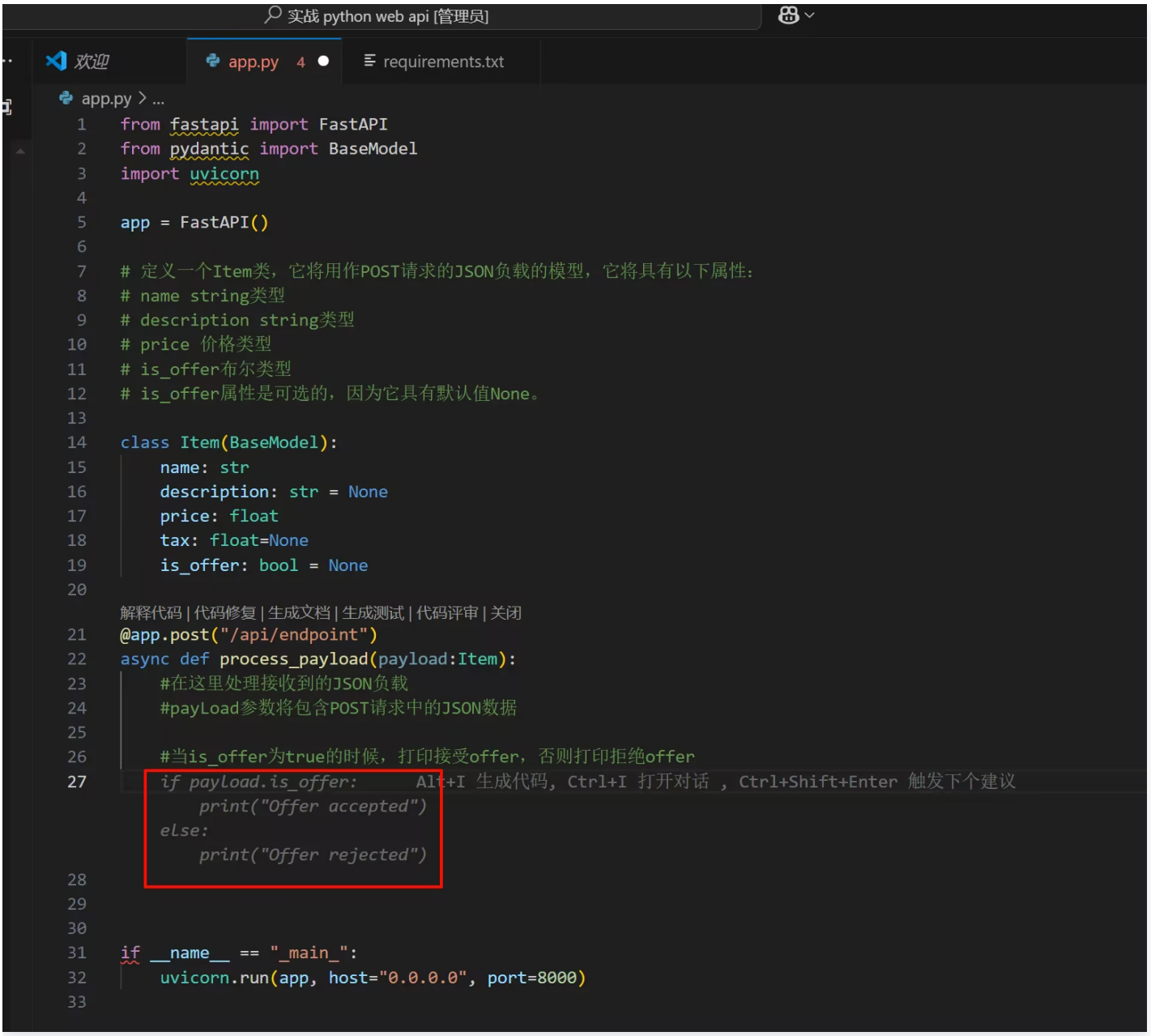


代码补全提示词描述，快速完成细节调整

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。您只需要再按一下回车。即可触发14行的完整补全。
接下来，删除 payload 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。



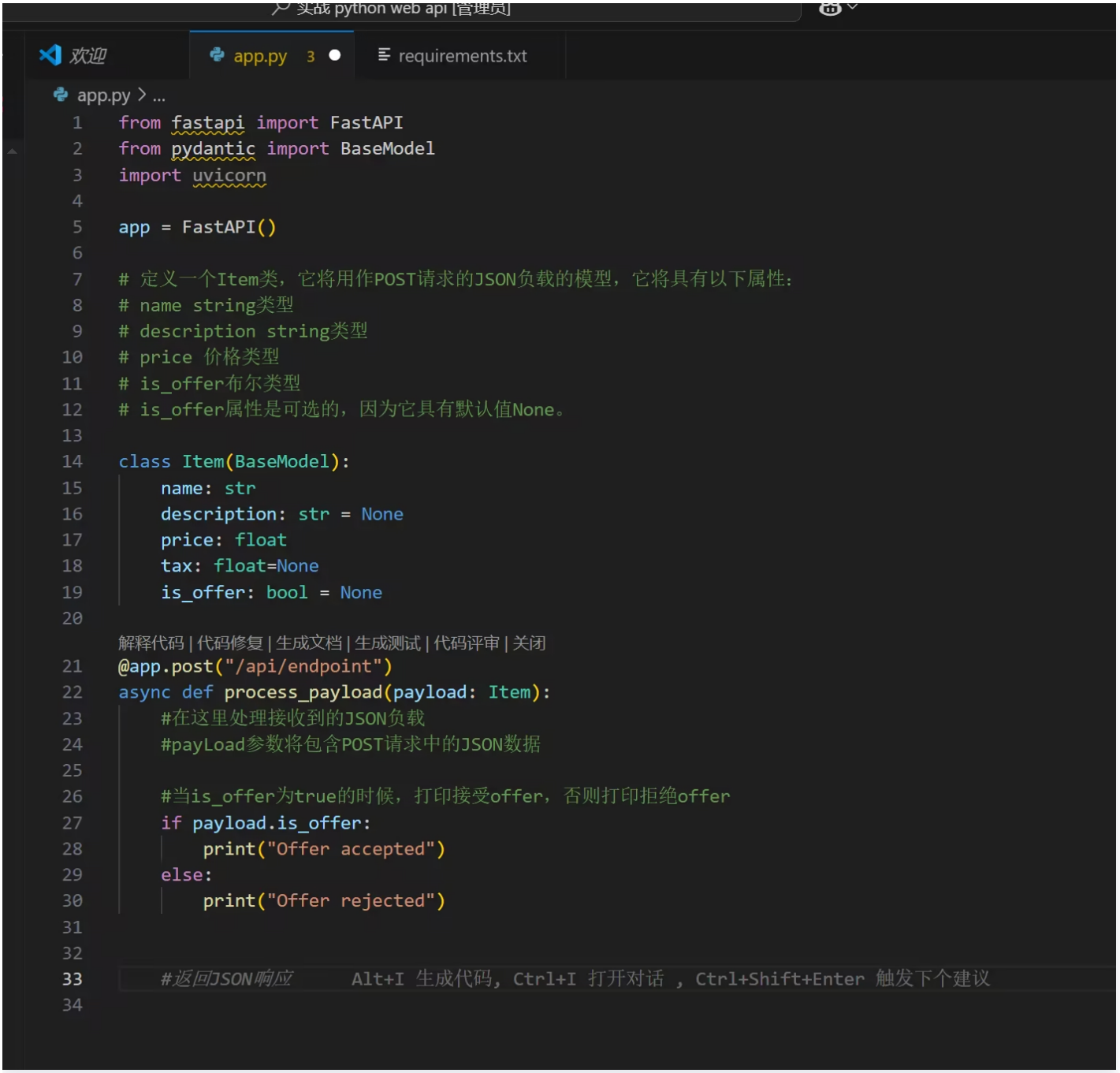
接下来继续进行补全，输入业务逻辑，要保持 Short（第三个S）简短。可能您这个需求很大，但由于当前为补全场景，需要更快地拆解需求并通过简短精炼的语句给出描述。如下：



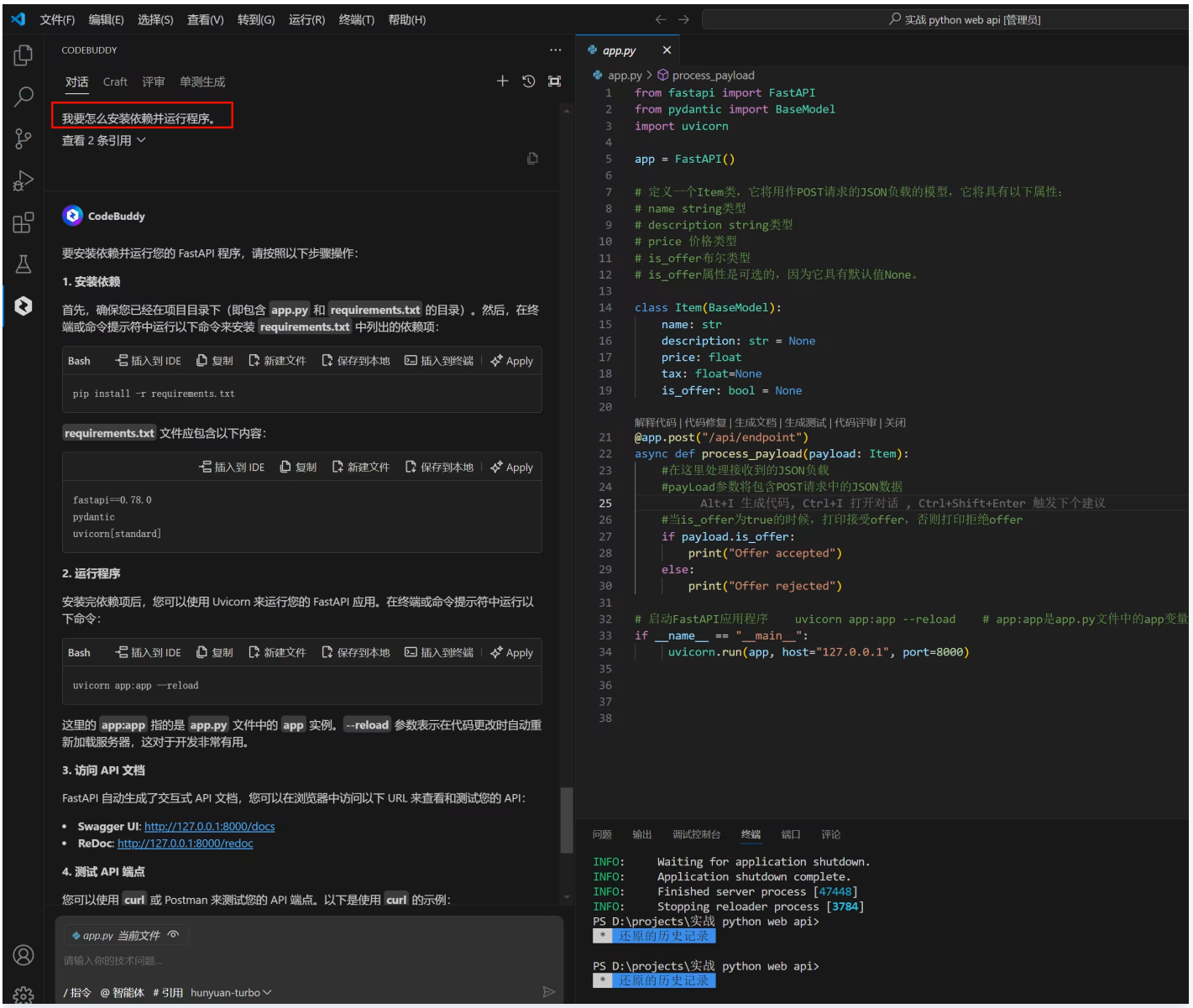
连续回车，代码助手会做一些强制刷新逻辑。当然后面也会提供快捷键和按钮，来更快捷的强制触发补全。其好处有：

- 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。

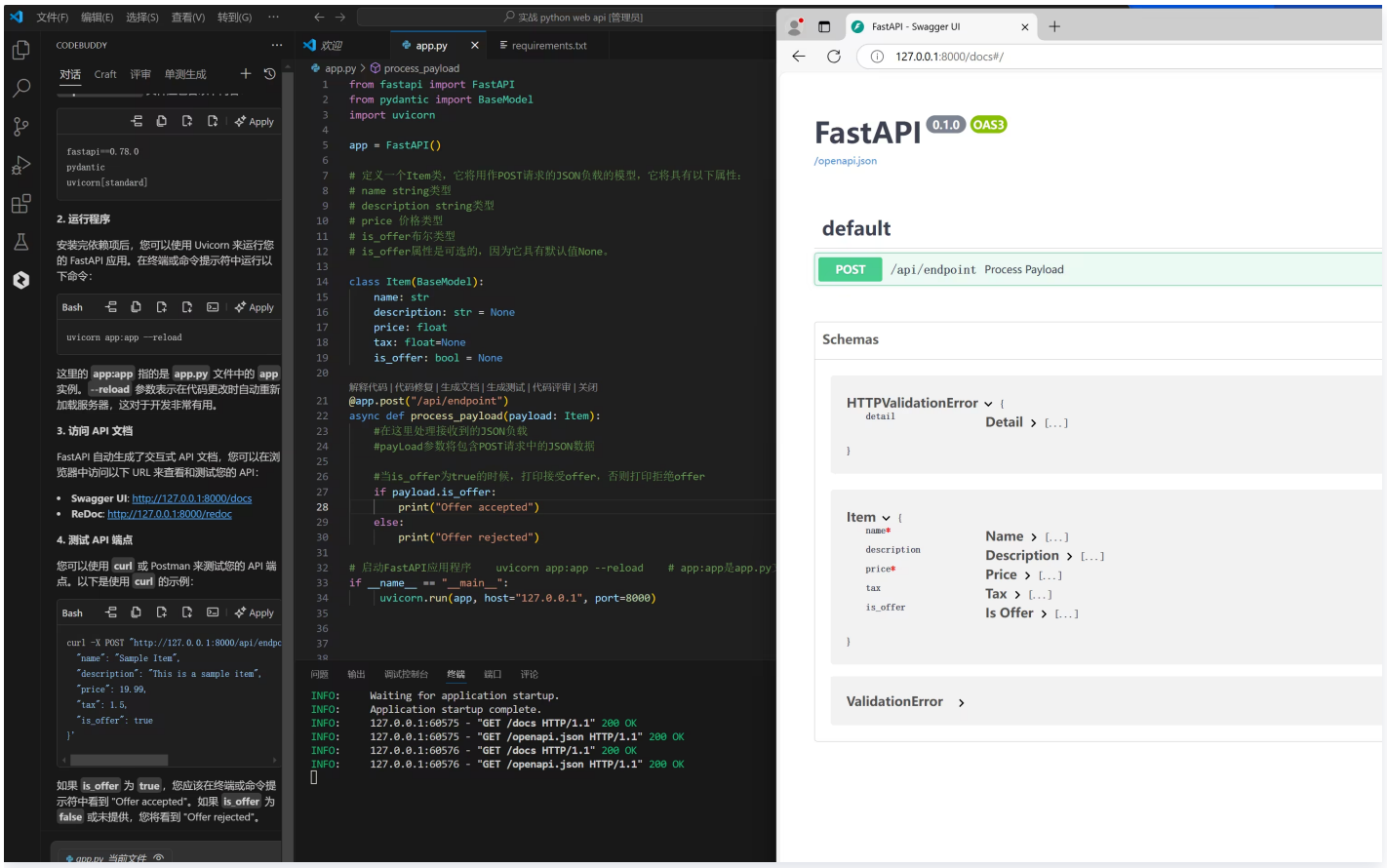


代码完成编写后不知道怎么运行，可以在对话框中，向代码助手提问“我要怎么安装依赖并运行程序”。如下图：

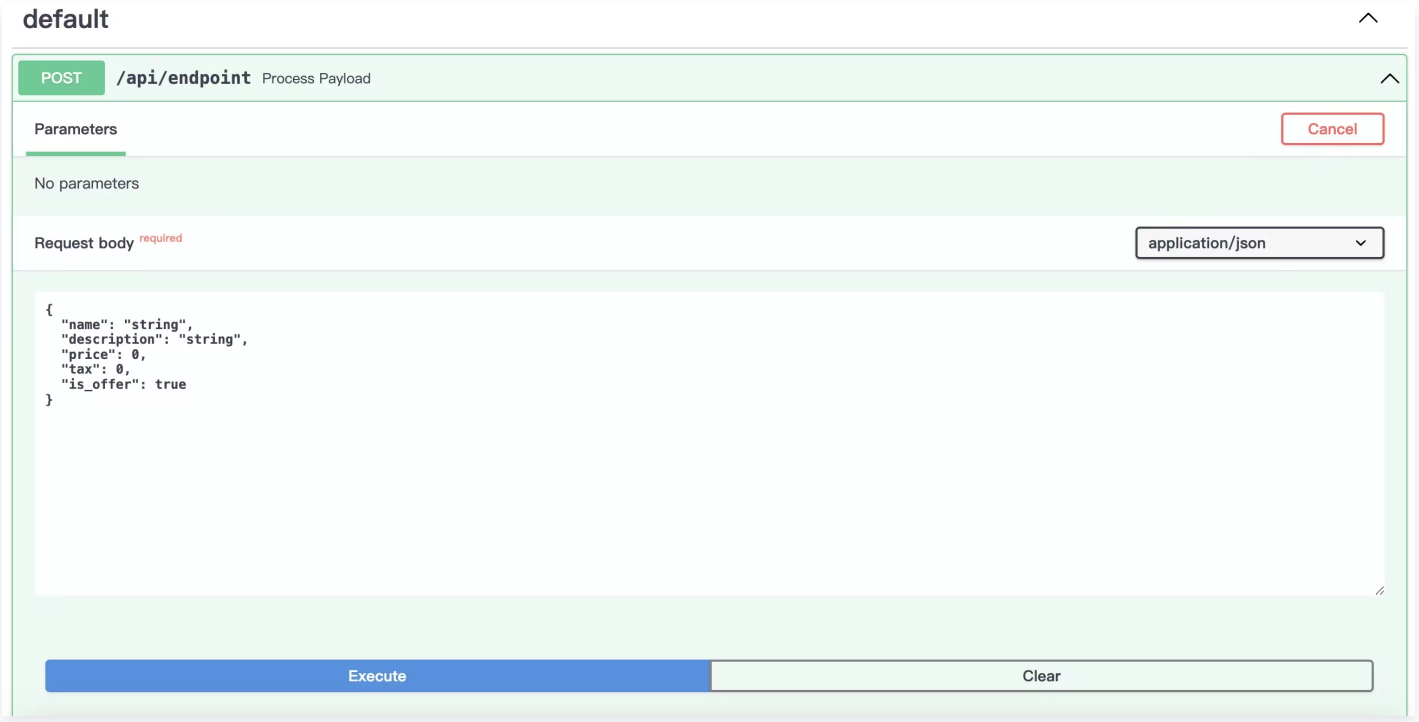


运行效果

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` 来确认该端点是否正常工作，运行成功。



来试试业务逻辑写的对不对。单击 **Execute**，运行正确。



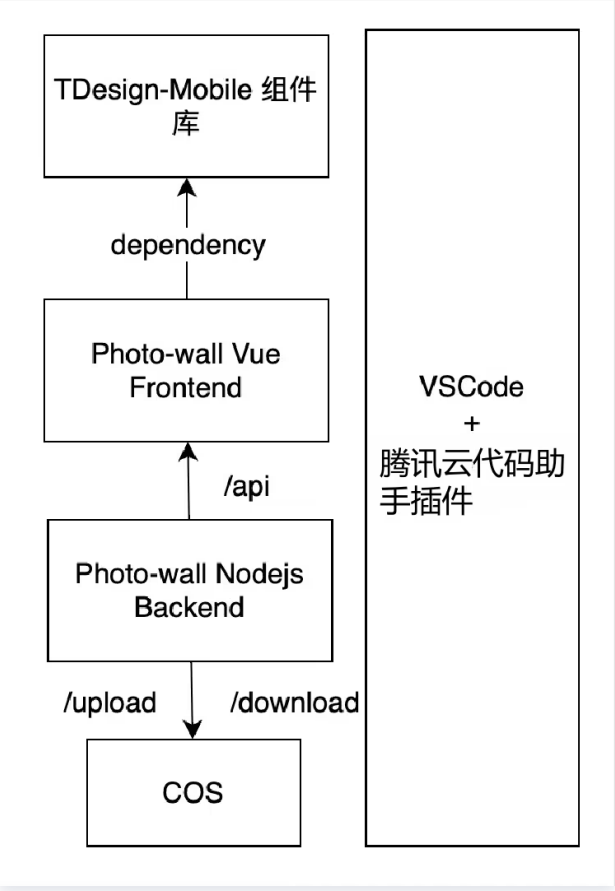
快速搭建在线相册应用

最近更新时间：2026-07-17 19:38:00

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。

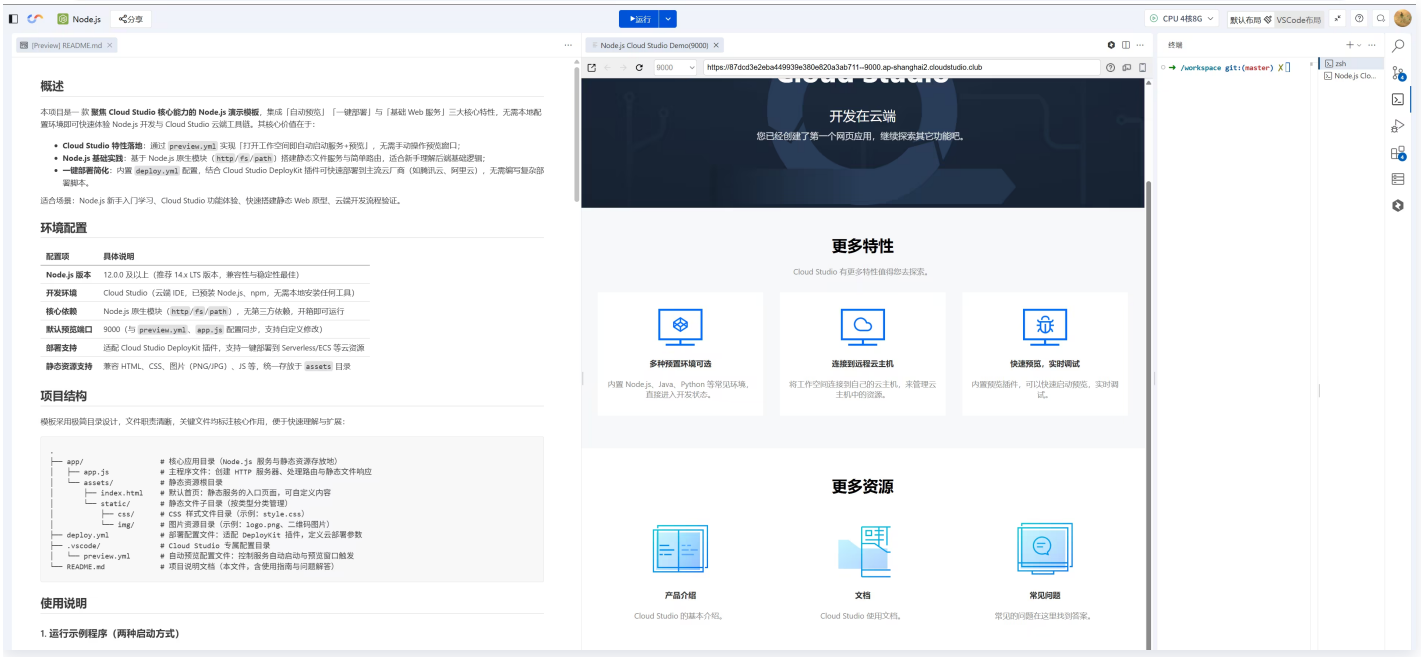


启动一个开发环境

打开 腾讯云 Cloud Studio ，单击右上角**创建应用**。选择一个 Node.js 开发环境，单击**创建**之后，即可进入开发环境。

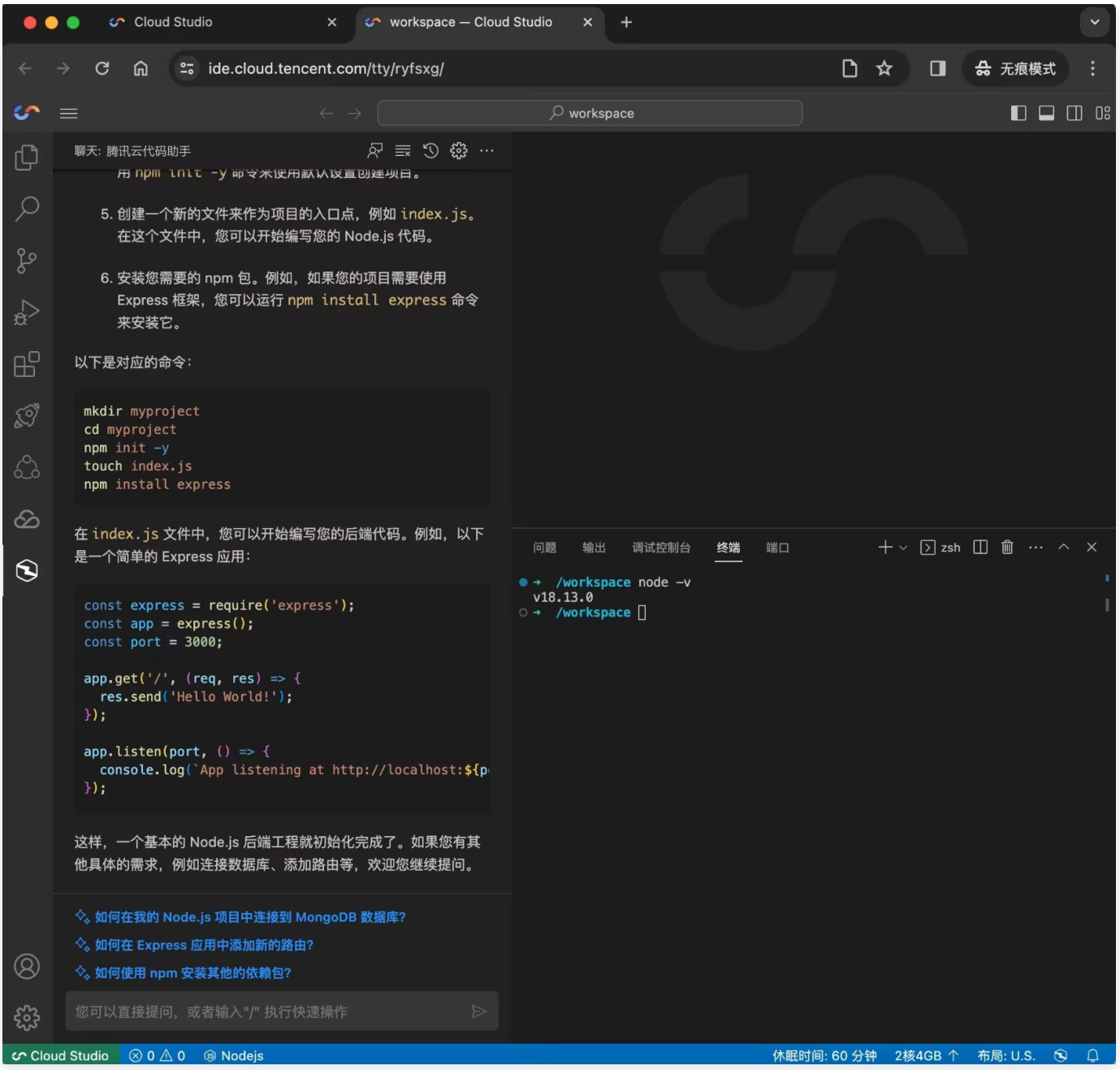
版权所有：腾讯云计算（北京）有限责任公司





后端应用

1. 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。**

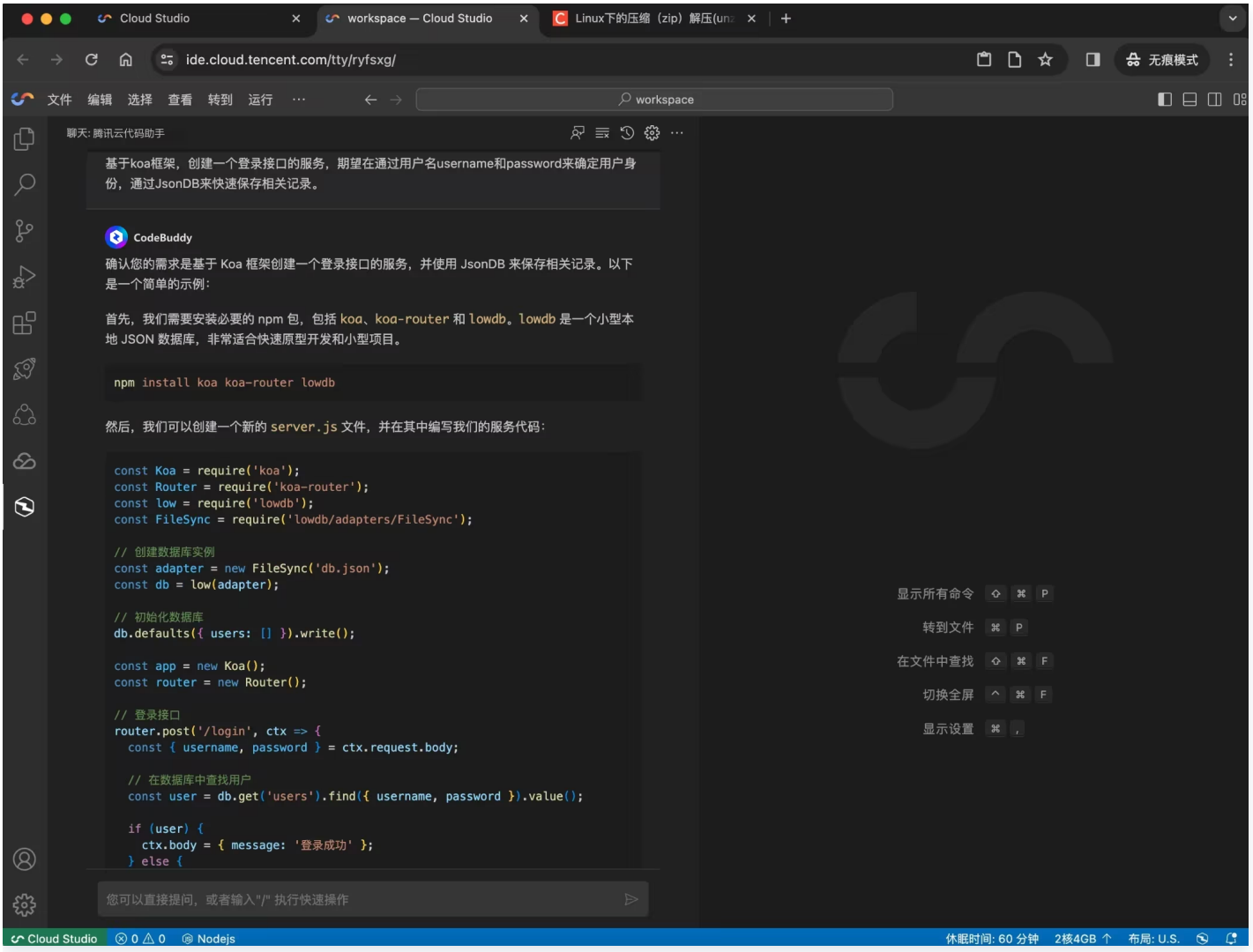


后端服务至少应包含：

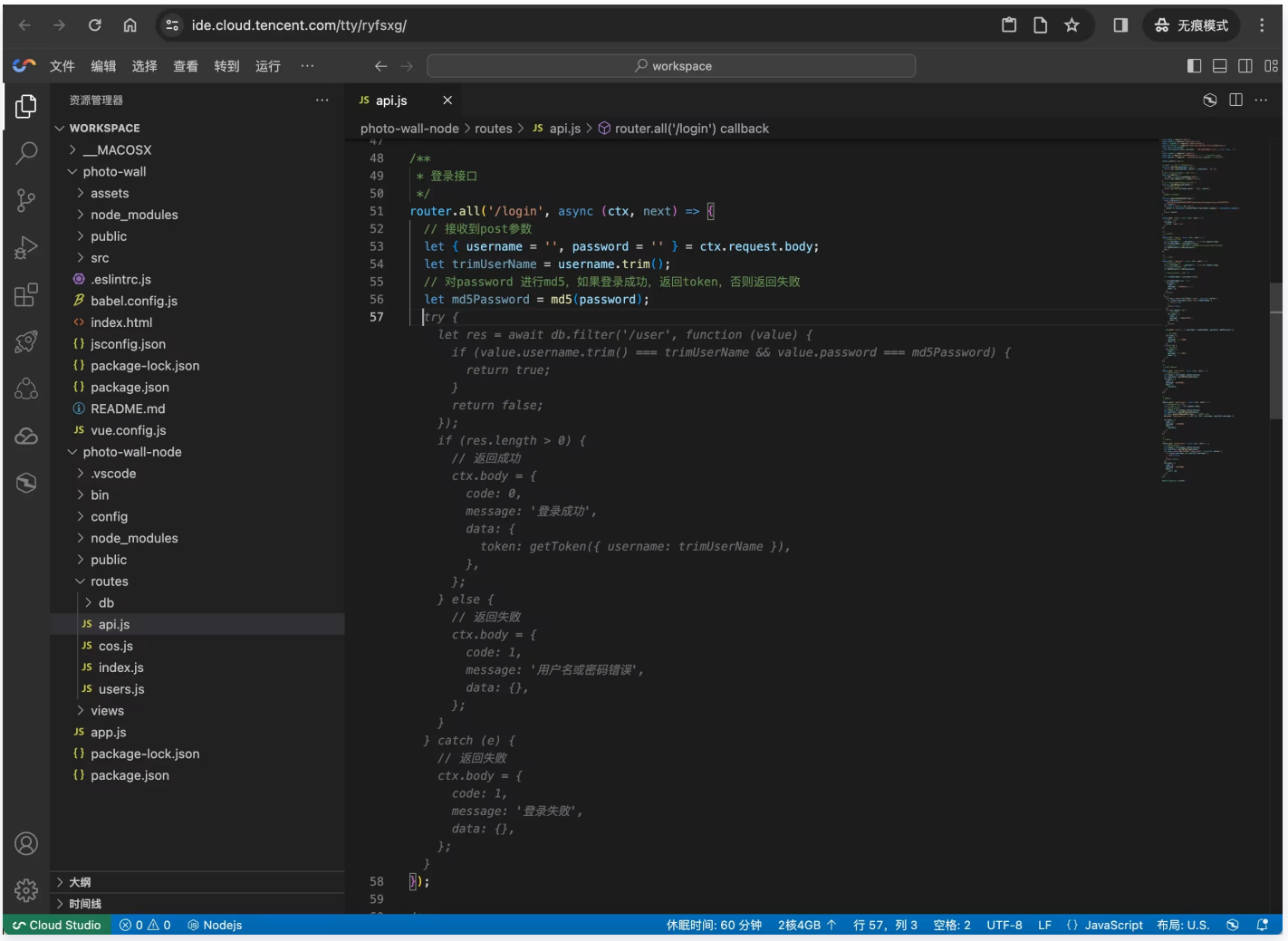
- 登录服务。
- 注册服务。
- 图片预览服务。
- 图片上传保存服务。

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。

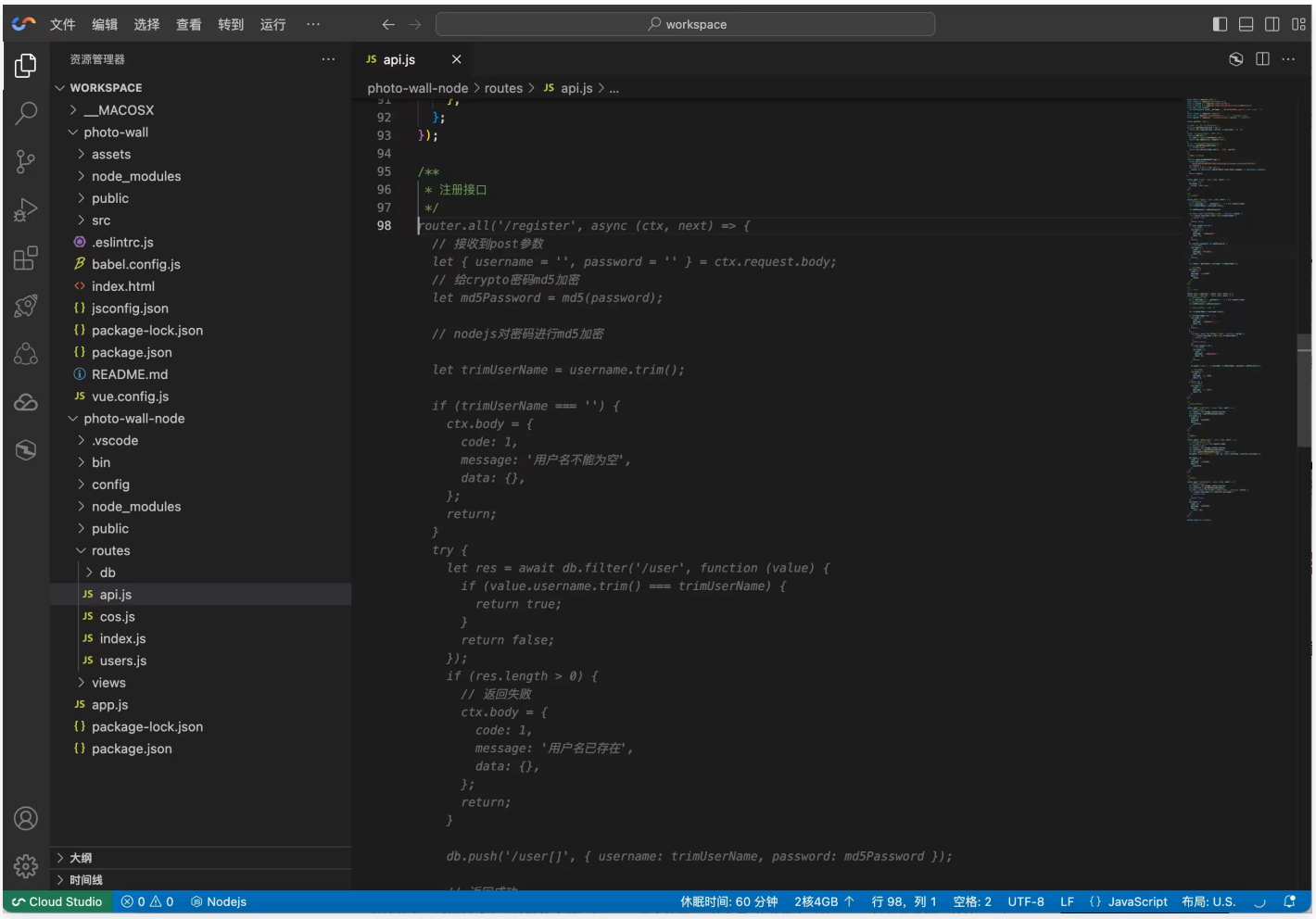


接着插入到当前工程，进行代码补全的代码微调，例如我们要对传过来的密码进行 MD5 加解密，我通过行内注释输入 // 对 password 进行 md5，如果登录成功，返回 token，否则返回失败，接着代码助手触发了代码补全，智能根据当前光标上下文，进行行或者块补全。如下：



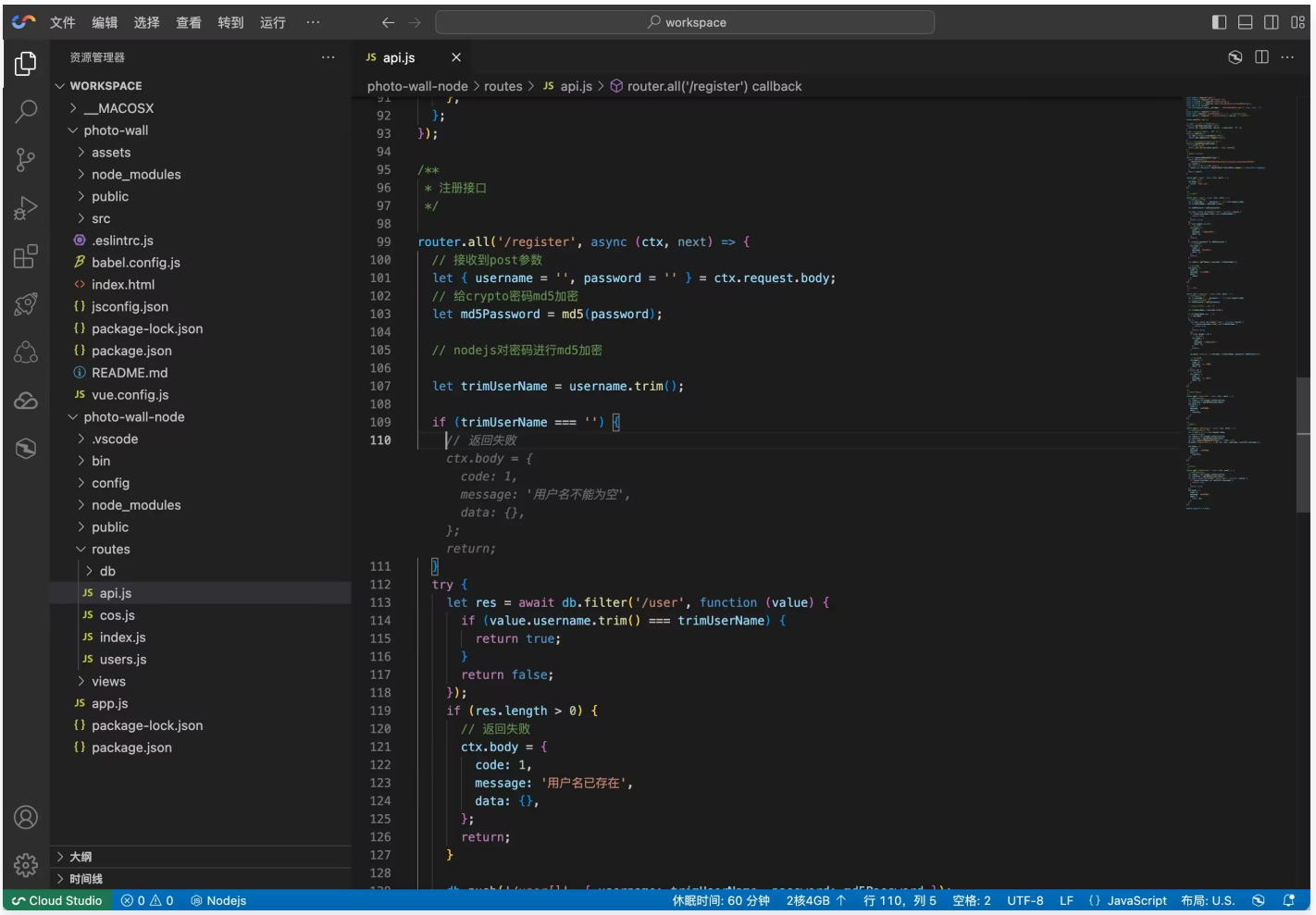
注册服务

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。



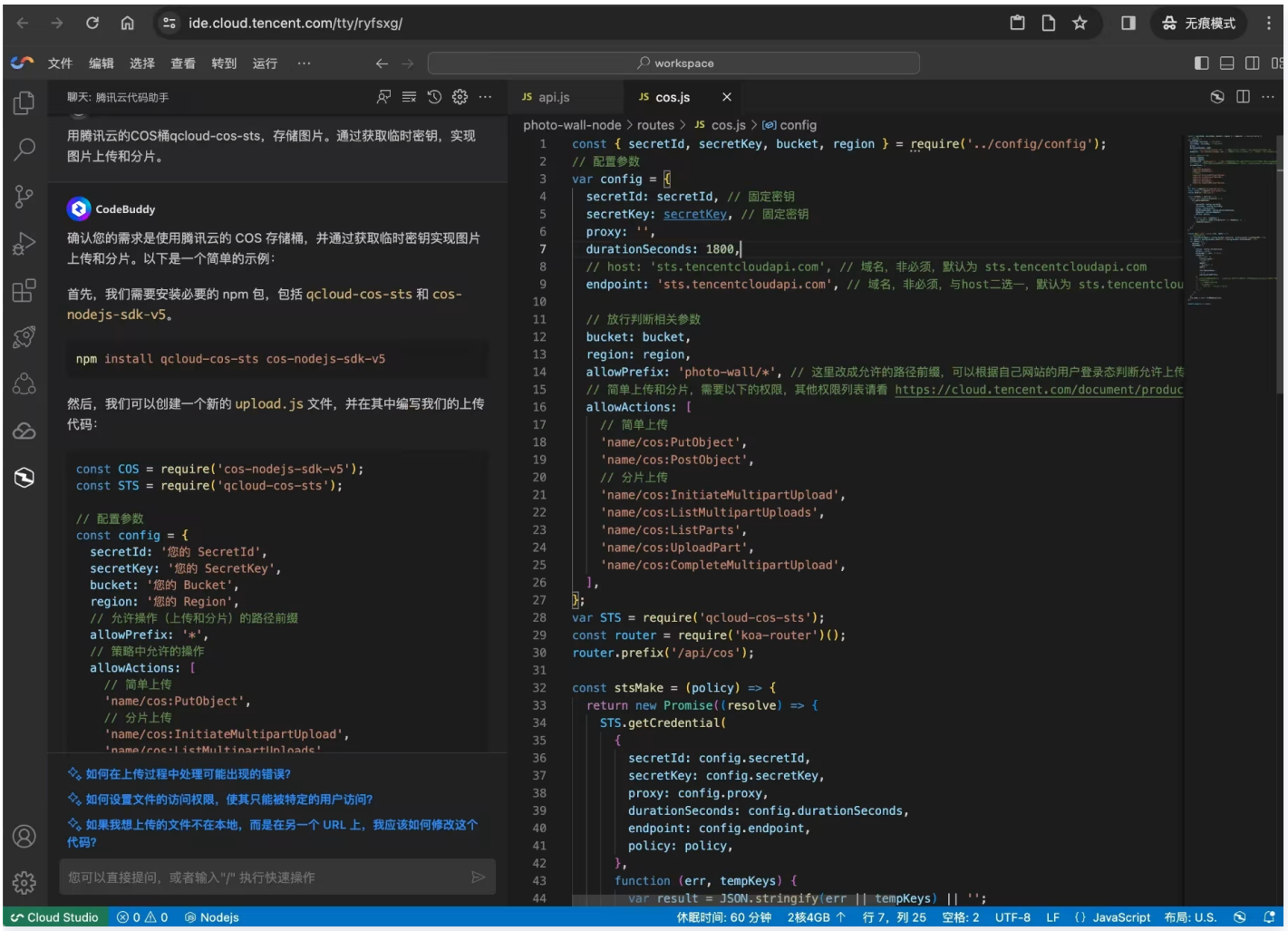
此处对代码逻辑进行调整。例如针对109行做一些代码调整，增加兜底逻辑。

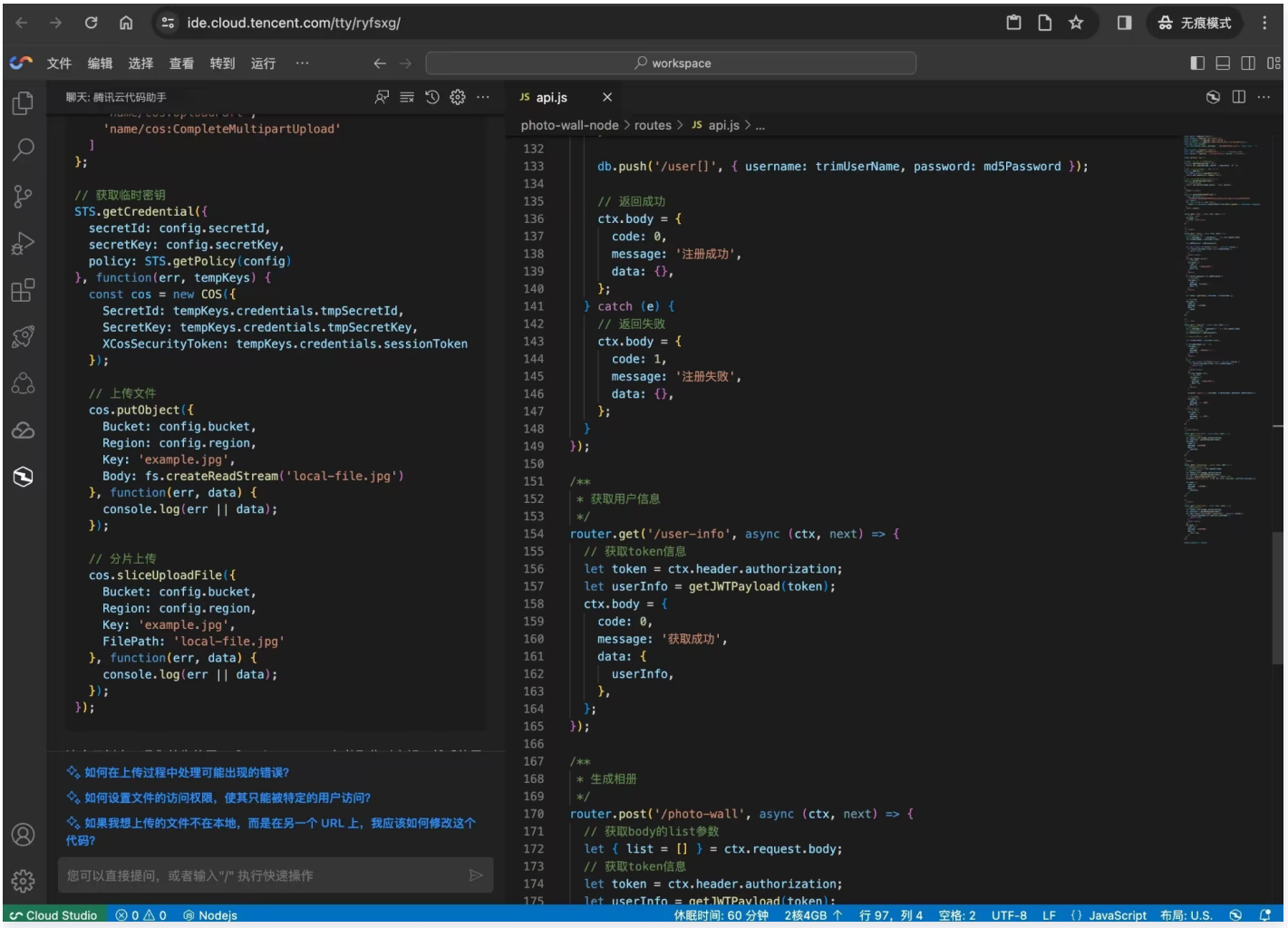
版权所有：腾讯云计算（北京）有限责任公司



COS 桶

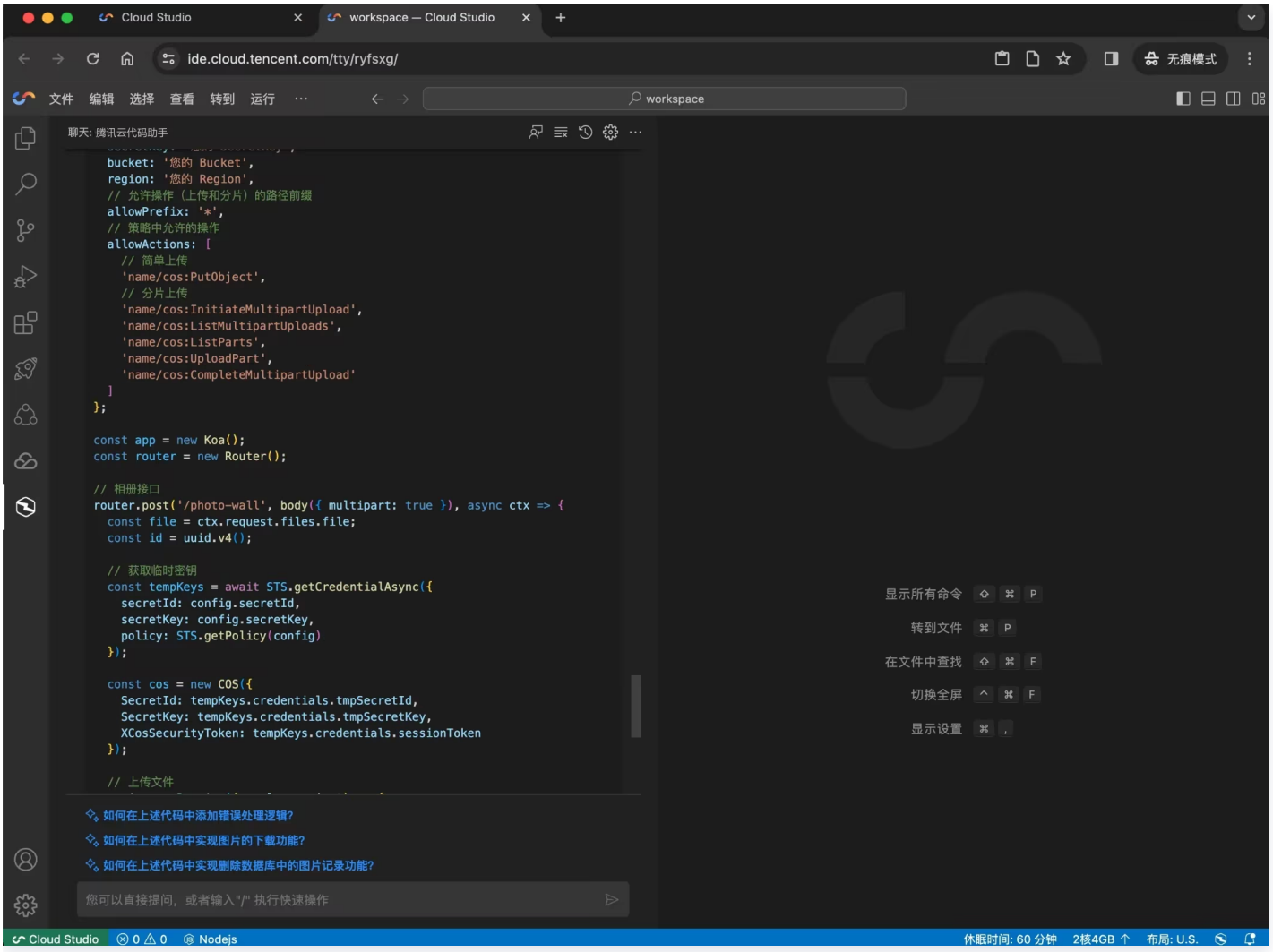
上传的图片需要存放到 COS 桶，接下来向代码助手提问：用腾讯云的 COS 桶 qcloud-cos-sts，存储图片。通过获取临时密钥，实现图片上传和分片。


ide.cloud.tencent.com/tty/ryfsxg/
无痕模式
文件 编辑 选择 查看 转到 运行 ···
workspace
聊天: 腾讯云代码助手
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。
CodeBuddy
确认您的需求是使用腾讯云的 COS 存储桶，并通过获取临时密钥实现图片上传和分片。以下是一个简单的示例：
首先，我们需要安装必要的 npm 包，包括 qcloud-cos-sts 和 cos-nodejs-sdk-v5。
npm install qcloud-cos-sts cos-nodejs-sdk-v5
然后，我们可以创建一个新的 upload.js 文件，并在其中编写我们的上传代码：
const COS = require('cos-nodejs-sdk-v5');
const STS = require('qcloud-cos-sts');
// 配置参数
const config = {
secretId: '您的 SecretId',
secretKey: '您的 SecretKey',
bucket: '您的 Bucket',
region: '您的 Region',
// 允许操作（上传和分片）的路径前缀
allowPrefix: '*',
// 策略中允许的操作
allowActions: [
// 简单上传
'name/cos:PutObject',
// 分片上传
'name/cos:InitiateMultipartUpload',
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
api.js
cos.js
photo-wall-node > routes > cos.js > config
const { secretId, secretKey, bucket, region } = require('../config/config');
// 配置参数
var config = {
secretId: secretId, // 固定密钥
secretKey: secretKey, // 固定密钥
proxy: '',
durationSeconds: 1800,
// host: 'sts.tencentcloudapi.com', // 域名，非必须，默认为 sts.tencentcloudapi.com
endpoint: 'sts.tencentcloudapi.com', // 域名，非必须，与host二选一，默认为 sts.tencentclou
// 放行判断相关参数
bucket: bucket,
region: region,
allowPrefix: 'photo-wall/*', // 这里改成允许的路径前缀，可以根据自己网站的用户登录态判断允许上传
// 简单上传和分片，需要以下的权限，其他权限列表请看 https://cloud.tencent.com/document/produc
allowActions: [
// 简单上传
'name/cos:PutObject',
'name/cos:PostObject',
// 分片上传
'name/cos:InitiateMultipartUpload',
'name/cos:ListMultipartUploads',
'name/cos:ListParts',
'name/cos:UploadPart',
'name/cos:CompleteMultipartUpload',
],
};
var STS = require('qcloud-cos-sts');
const router = require('koa-router')();
router.prefix('/api/cos');
const stsMake = (policy) => {
return new Promise((resolve) => {
STS.getCredential(
{
secretId: config.secretId,
secretKey: config.secretKey,
proxy: config.proxy,
durationSeconds: config.durationSeconds,
endpoint: config.endpoint,
policy: policy,
},
function (err, tempKeys) {
var result = JSON.stringify(err || tempKeys) || '';
Cloud Studio
0 0
Nodejs
休眠时间: 60 分钟
2核4GB
行 7，列 25
空格: 2
UTF-8
LF
JavaScript
布局: U.S.

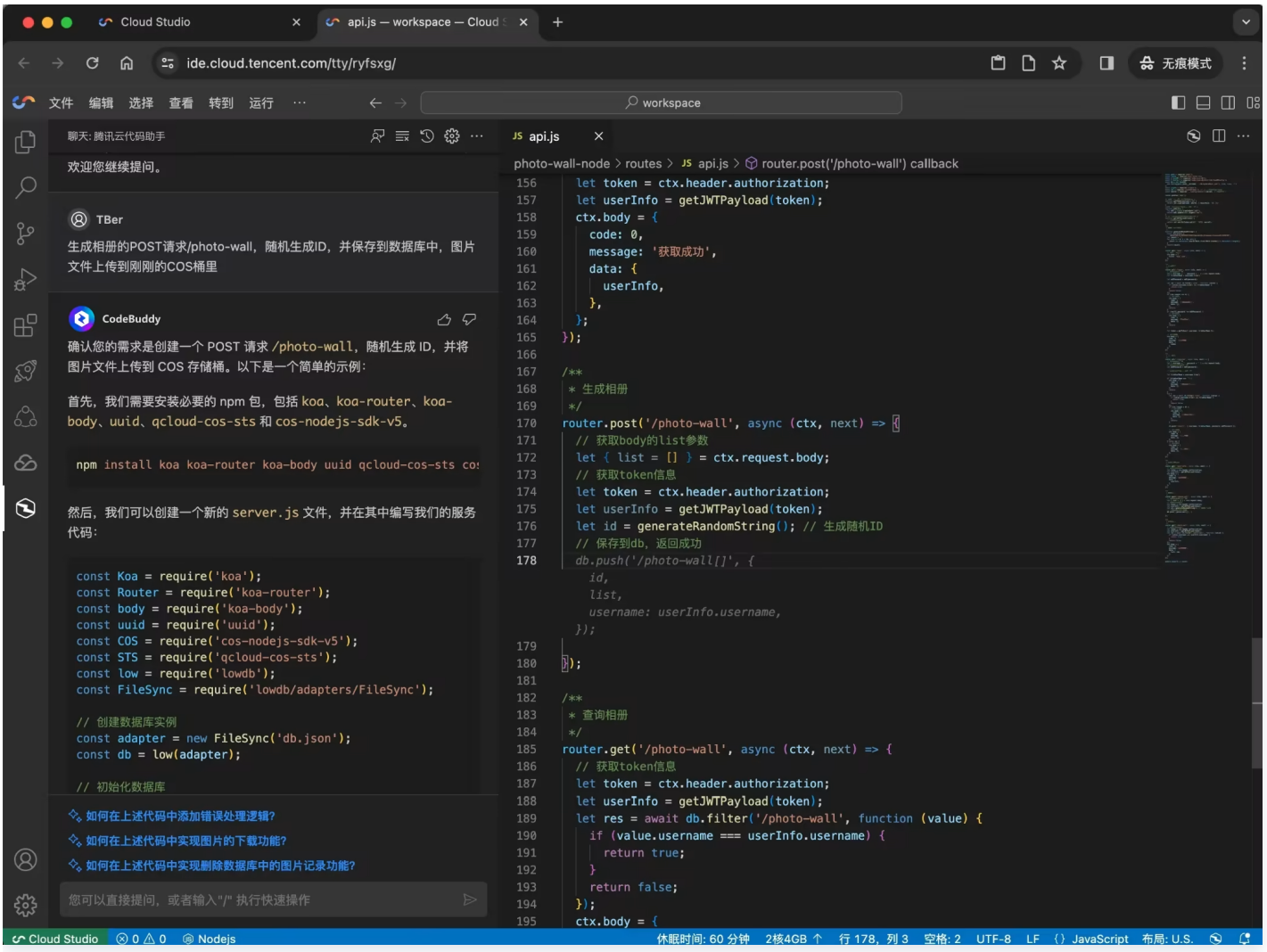


图片预览服务

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。

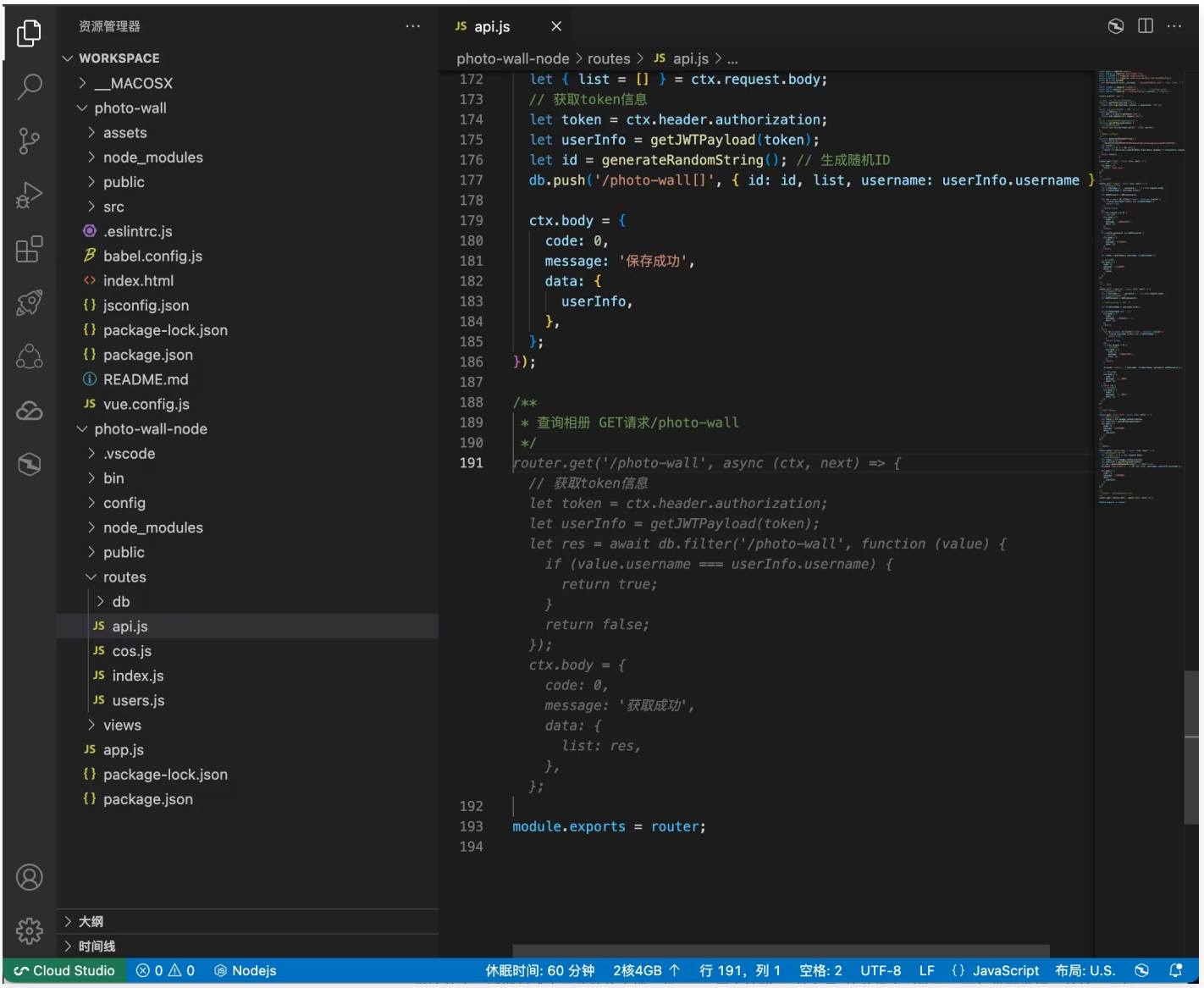


找到核心代码逻辑，复制到编辑器中，触发补全进行代码微调。



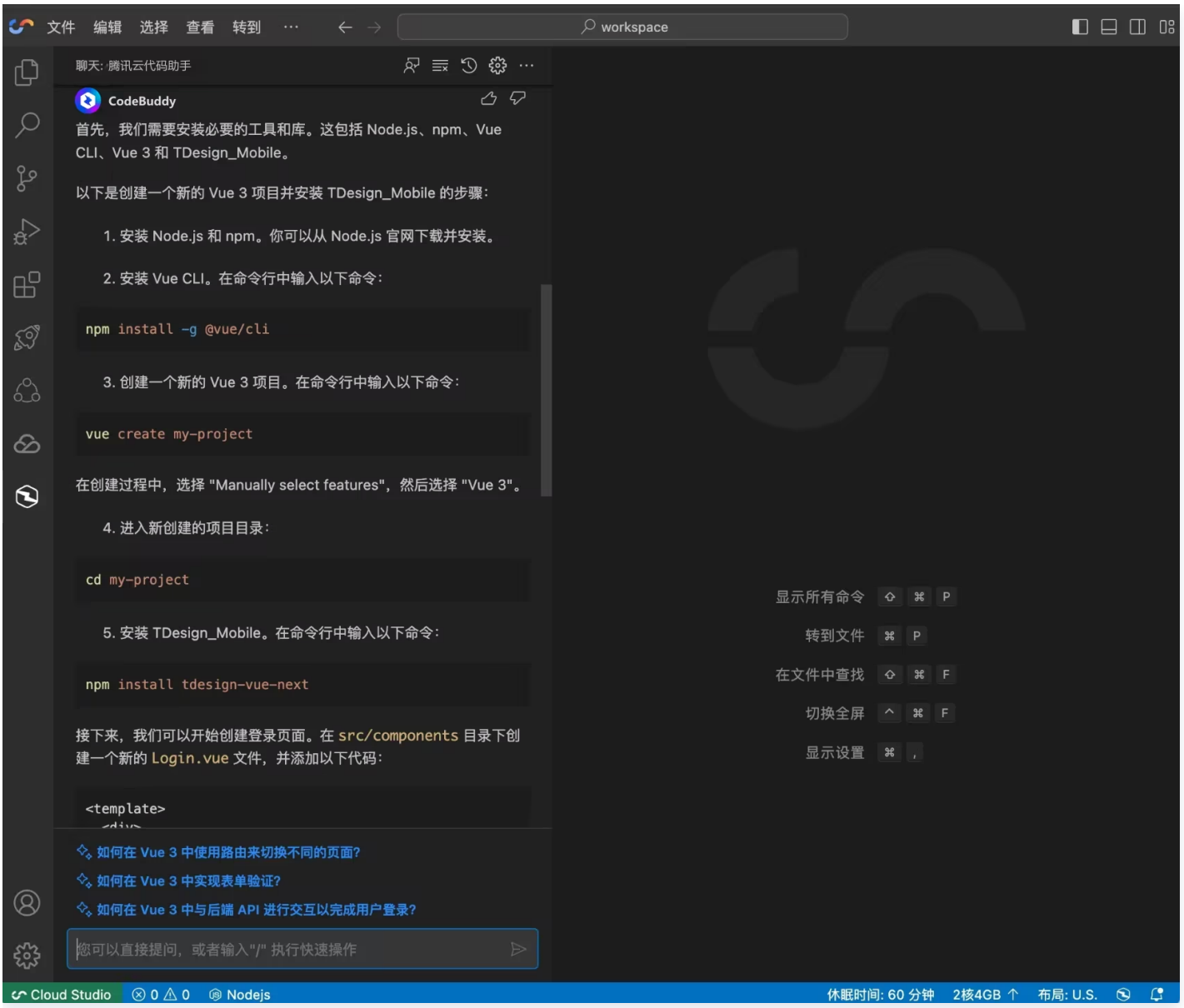
图片相册列表服务

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。



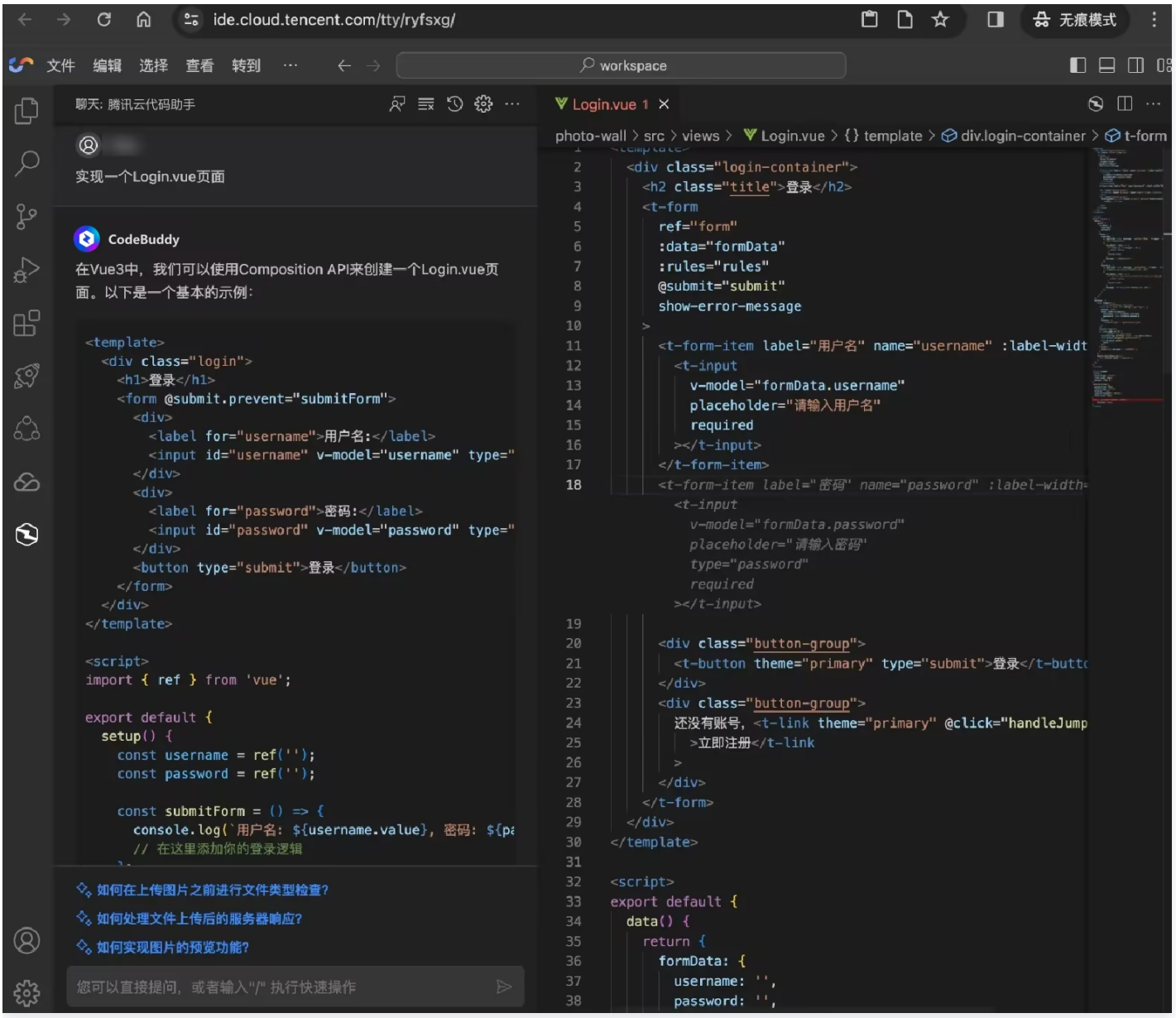
前端应用

前端应用采用 Koa2+TDesign_Mobile 组件库 。



Vue 页面

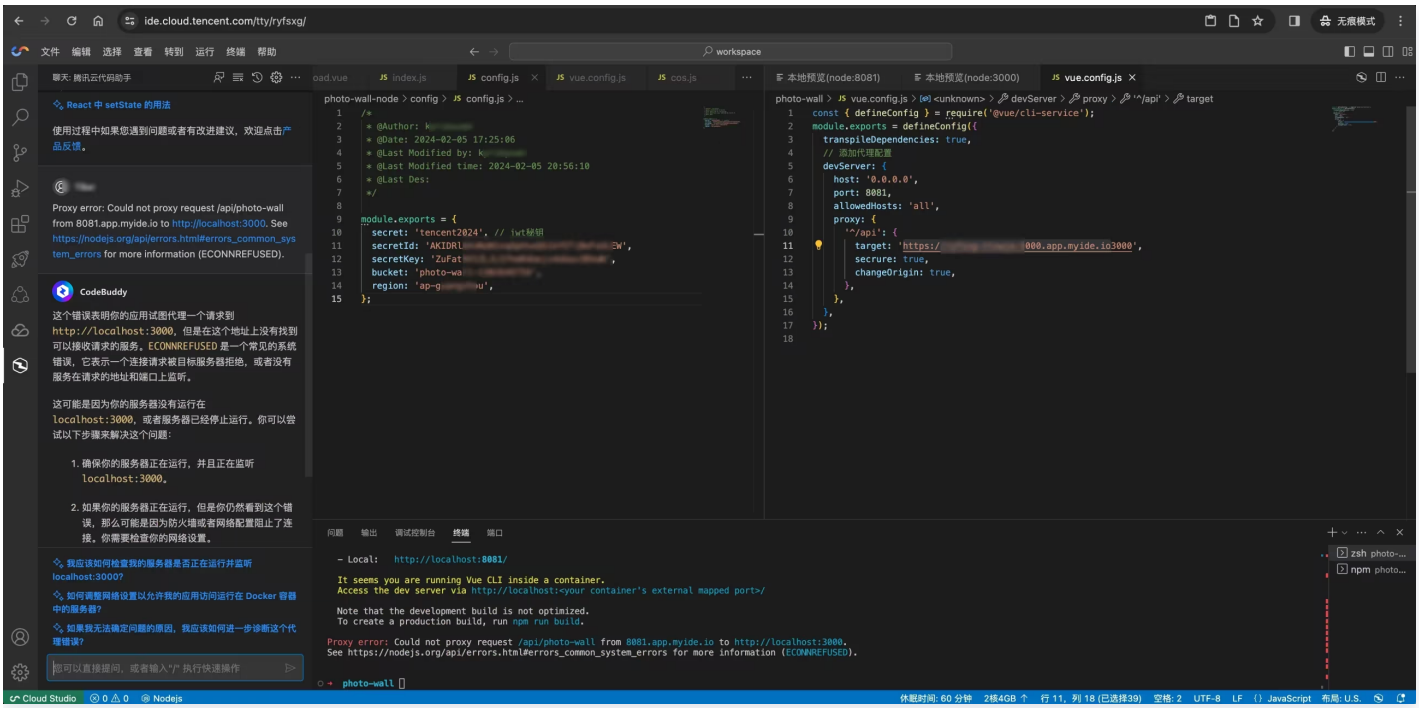
创建对应几个 Vue 文件，如图，通过对话实现一个 Login.vue 页面先去生成个大概。然后通过插入代码之后，再进行代码补全微调，从而生成自己想要的代码逻辑。



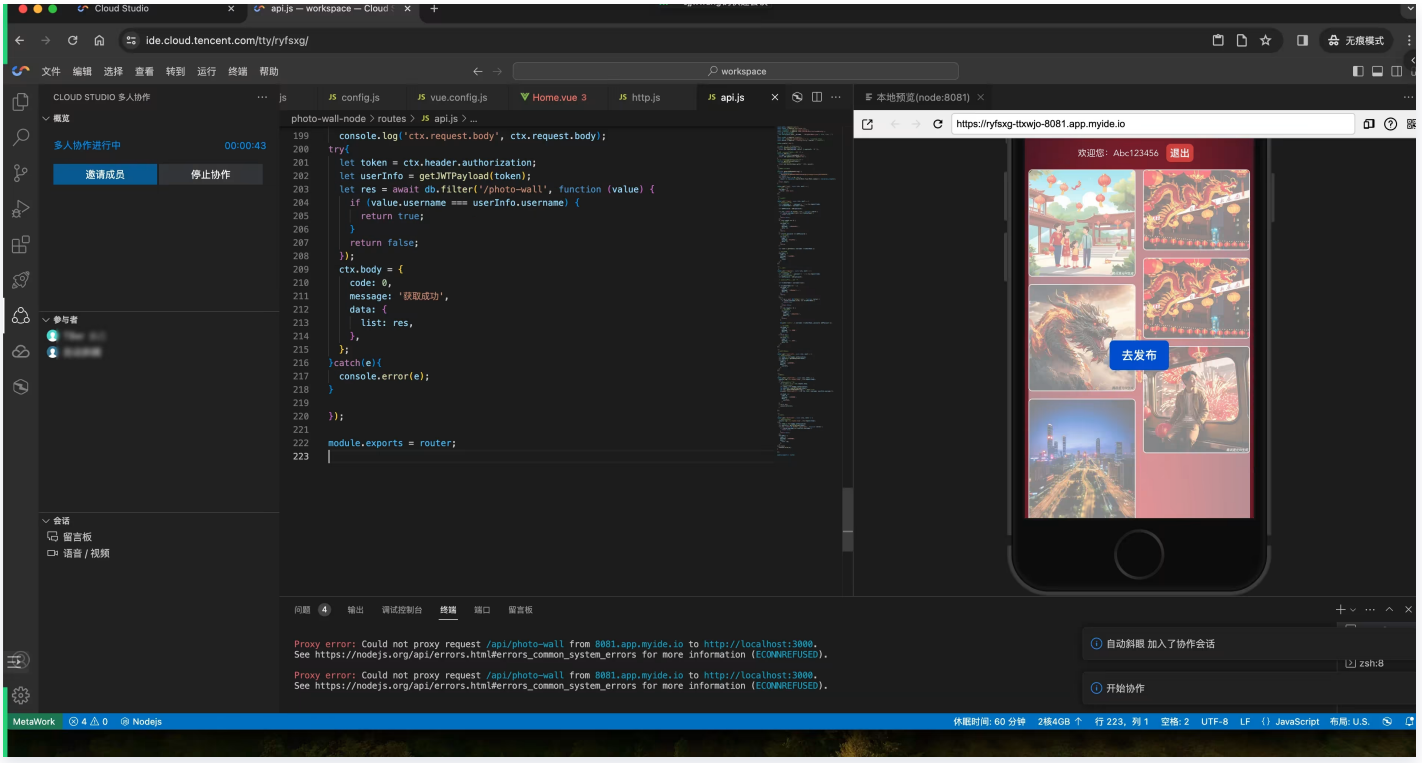
操作方式类似，由于篇幅有限，这里不一一截取。通过组合拳，对话+补全，加上多文件能力，使得前后端的开发工作效率得到提升加速。

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：



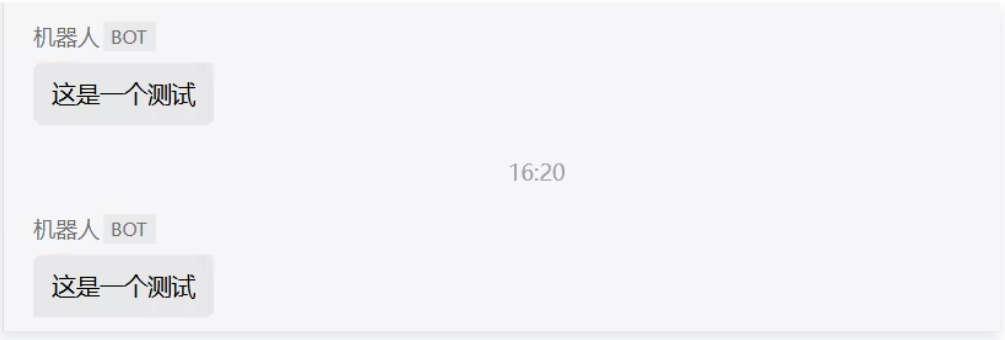
快速制作企微提醒机器人

最近更新时间：2026-07-17 17:54:32

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。本文将以制作一个企微提醒机器人为例，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。

版权所有：腾讯云计算（北京）有限责任公司

- 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的**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。

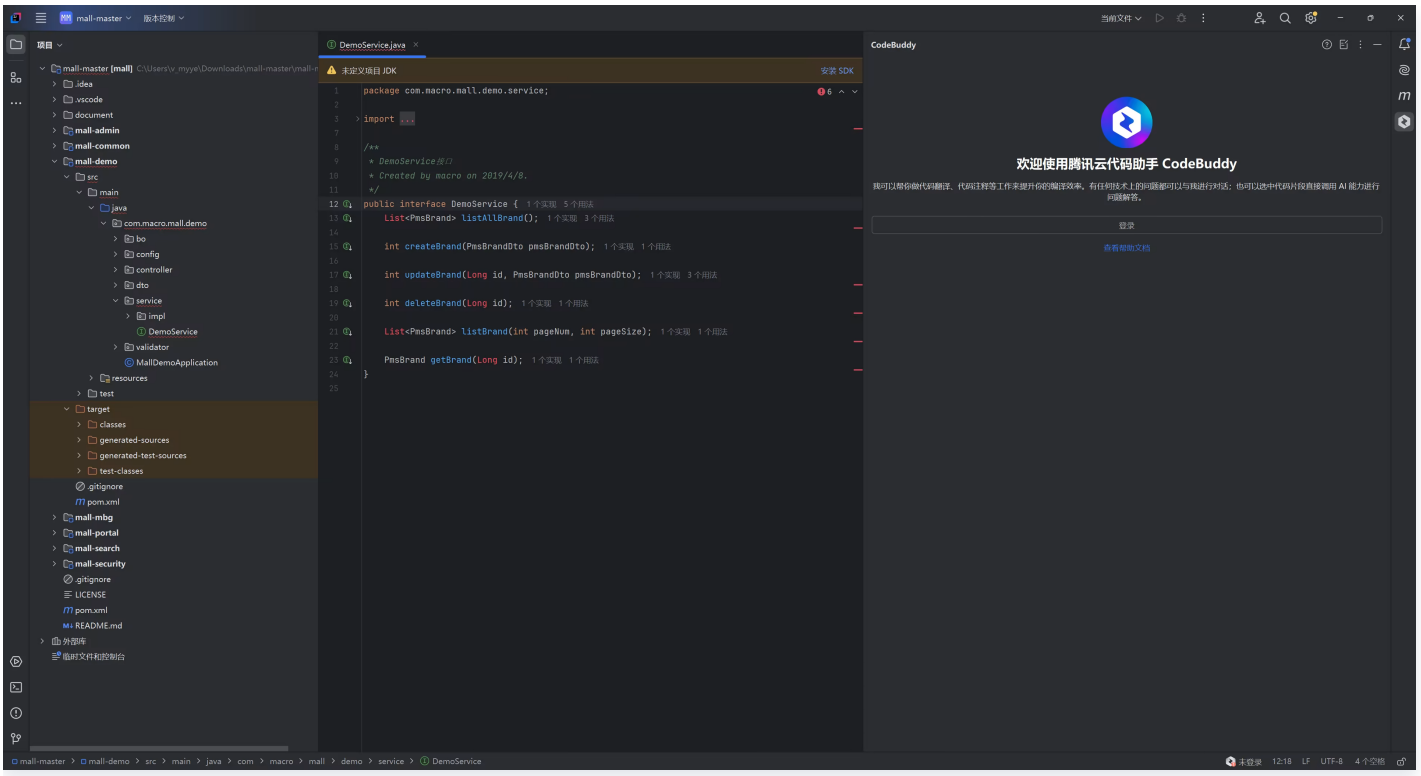


2. 打开 JetBrains 软件，例如我使用的是 IntelliJ IDEA，在设置里找到插件，然后搜索腾讯云代码助手或 Tencent Cloud CodeBuddy 然后进行安装，安装完成后，需要重启。



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，进行登录操作。

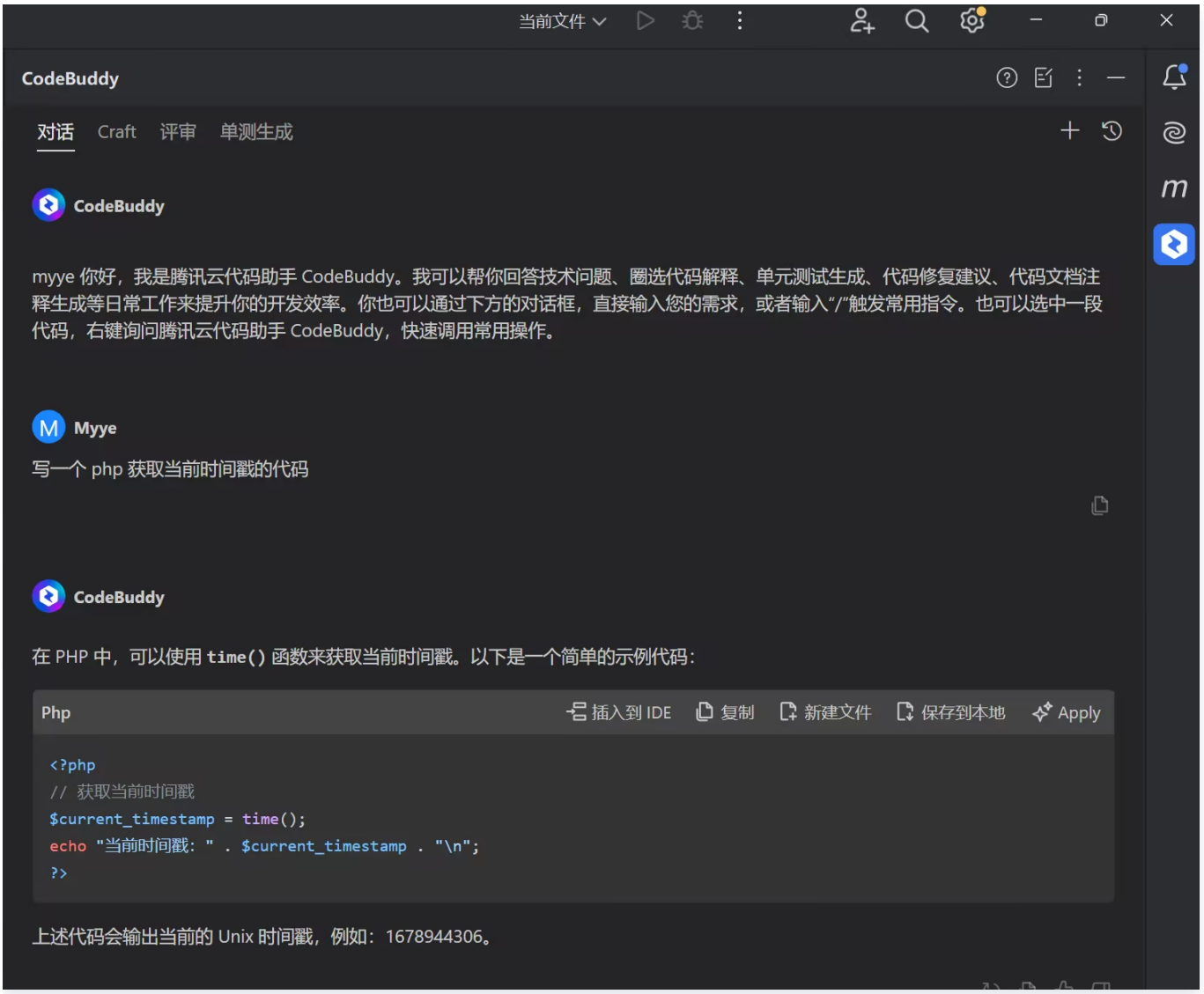


然后单击侧边栏，进入助手聊天界面。



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”。

版权所有：腾讯云计算（北京）有限责任公司



进入企业微信，创建机器人

创建机器人

进入企微工作台，搜索智能机器人并进入页面，单击创建机器人。



按照指引创建完成后，复制 Webhook 地址。



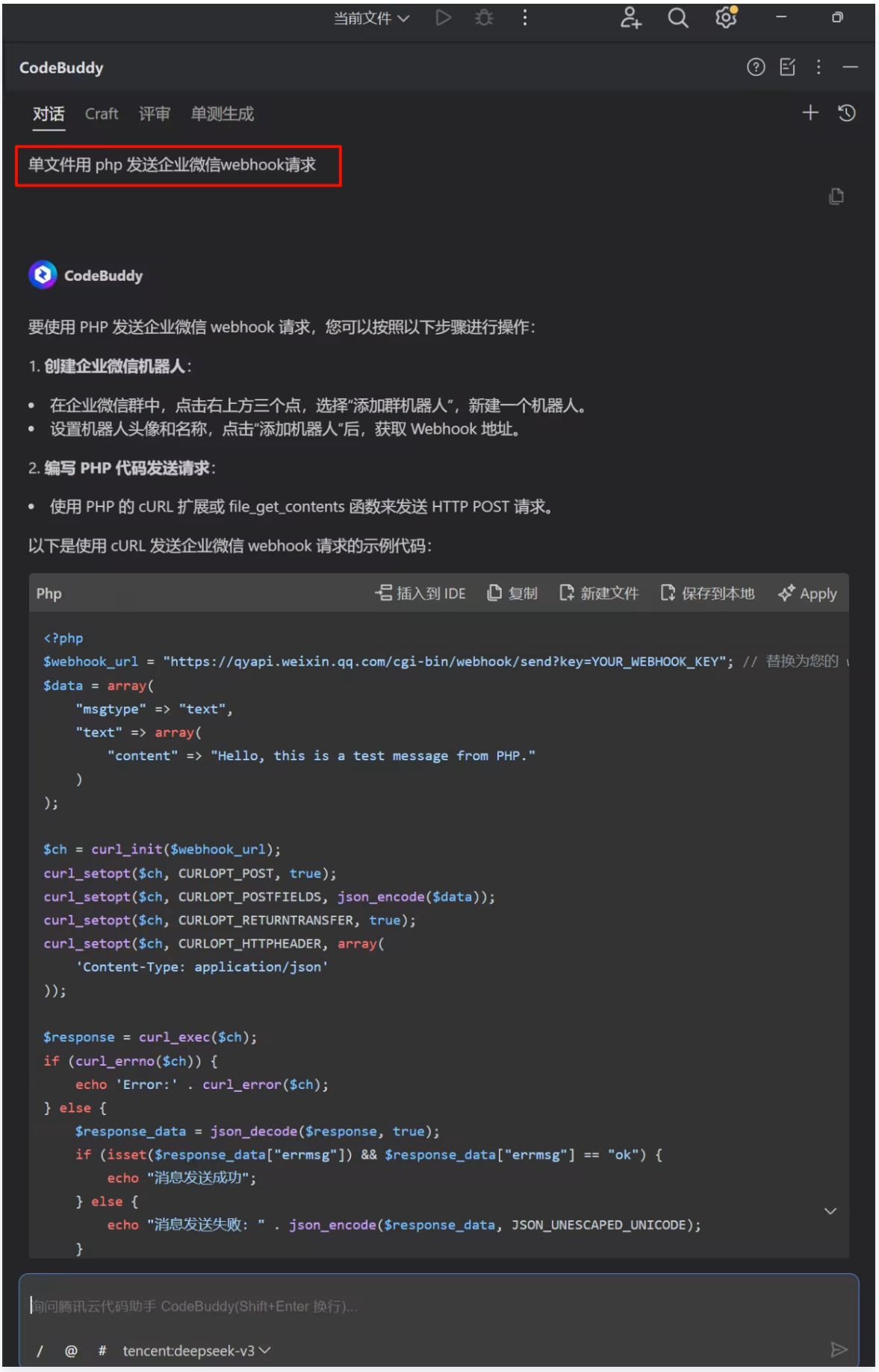
复制机器人参数

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。


当前文件
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：
1. 创建企业微信机器人：
在企业微信群中，点击右上方三个点，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。
2. 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

2. 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的 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
```

3. 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=xxx"; // 替换为您的 webhook 地址
```

4. 然后运行代码（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）。

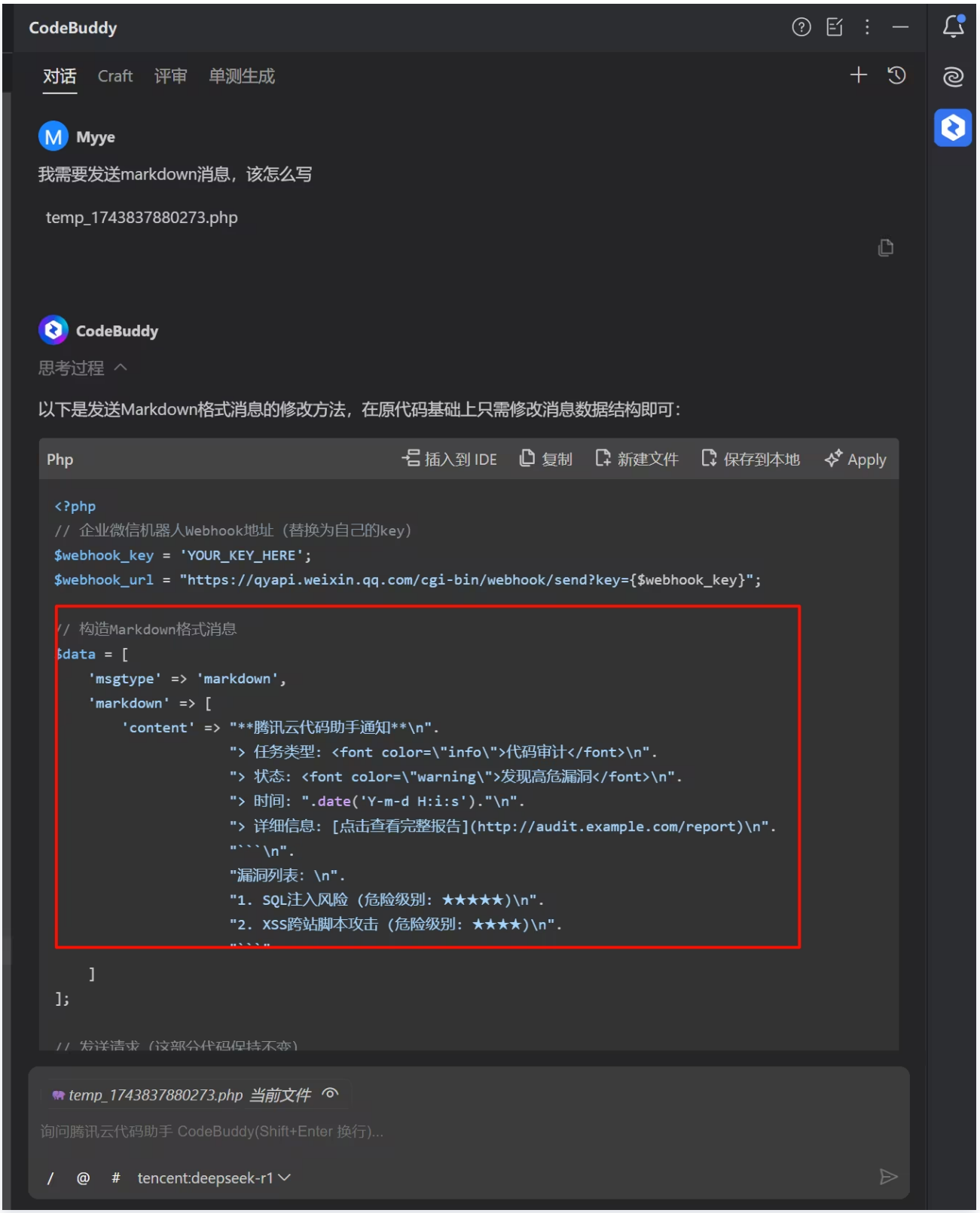
   浏览器运行一下代码。



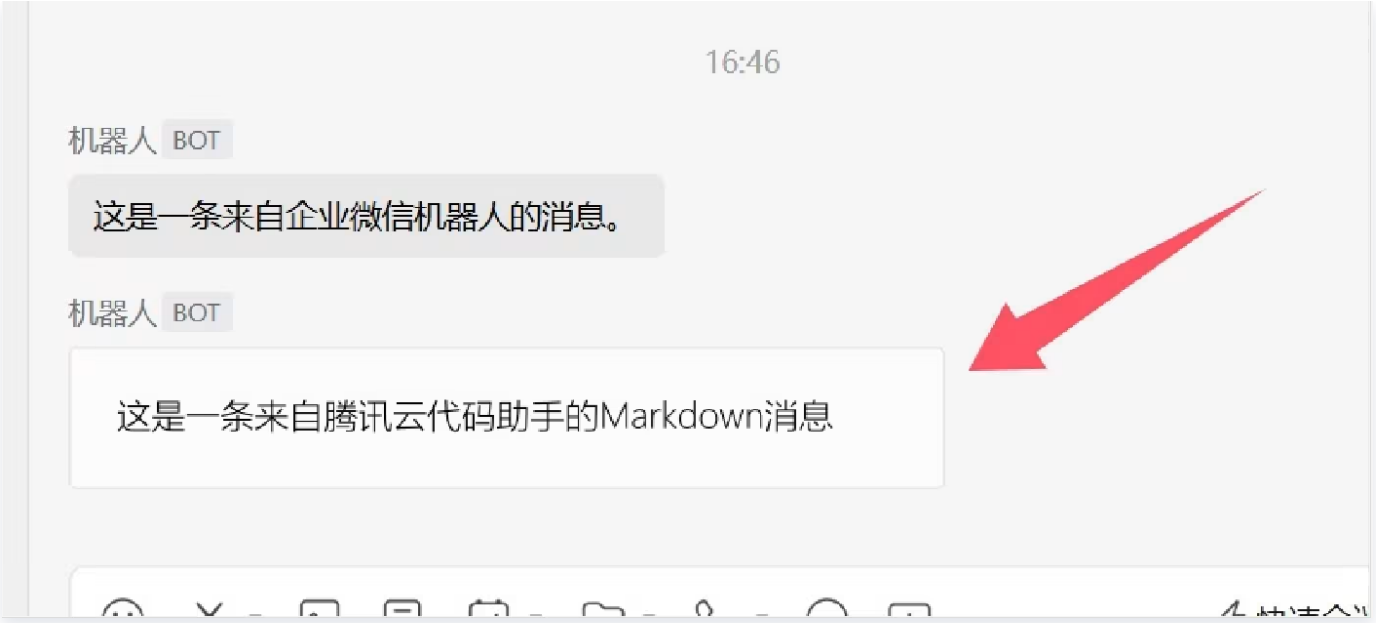
   运行发送成功。



5. 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向助手提问。应用代码时把 content 改成自己想要的内容即可。



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
<?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'
//    ]
//];
```

```php
// 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，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。

版权所有：腾讯云计算（北京）有限责任公司

快速开发一个微信小程序游戏

最近更新时间：2026-07-17 19:38:00

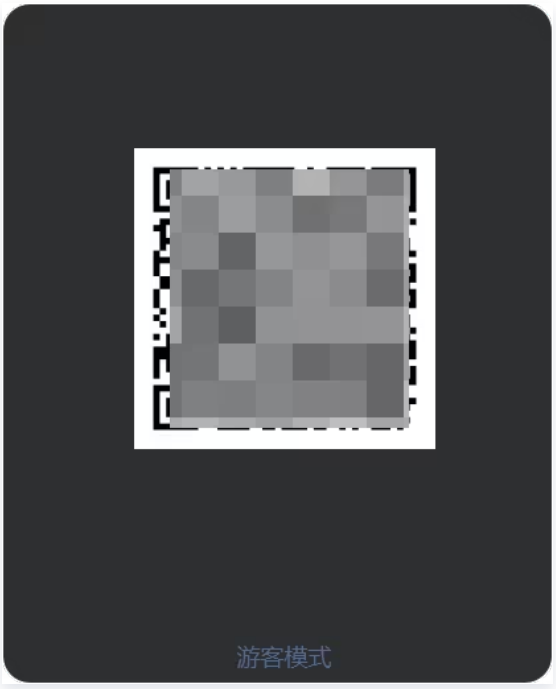
腾讯云代码助手已经支持在微信开发者工具 IDE 下的编码辅助，在代码辅助的效率上有很大提升，通过自然语言描述就可以完成最终应用。并且，支持了全新软件开发智能体 Craft，输入自然语言指令，支持文件多选，AI 深度理解代码工程，自主完成工程多文件代码生成和改写，快速落地可执行应用。现在，为您介绍如何利用软件开发智能体 Craft 来快速开发一个微信小程序游戏——**五子棋**。

准备工作

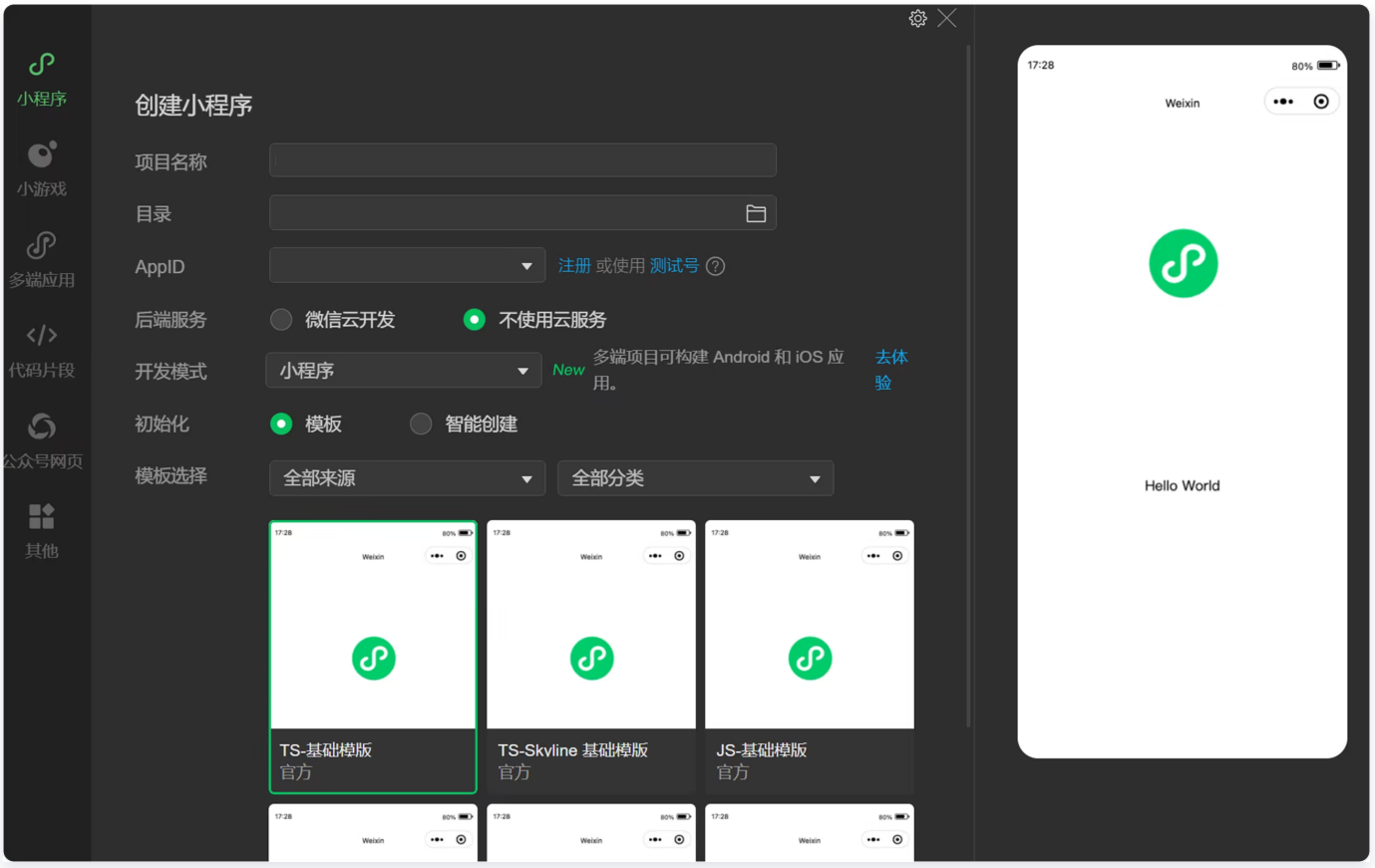
1. 请前往 微信开发者工具下载网址 下载微信开发者工具并安装，建议下载最新稳定版，例如现在的最新稳定版是 **2.01.2510290**。



2. 想要了解更多开发资料可前往 小程序开发微信官网文档 查看相关参考文档。
3. 安装完微信开发者工具后，打开并微信扫码登录。



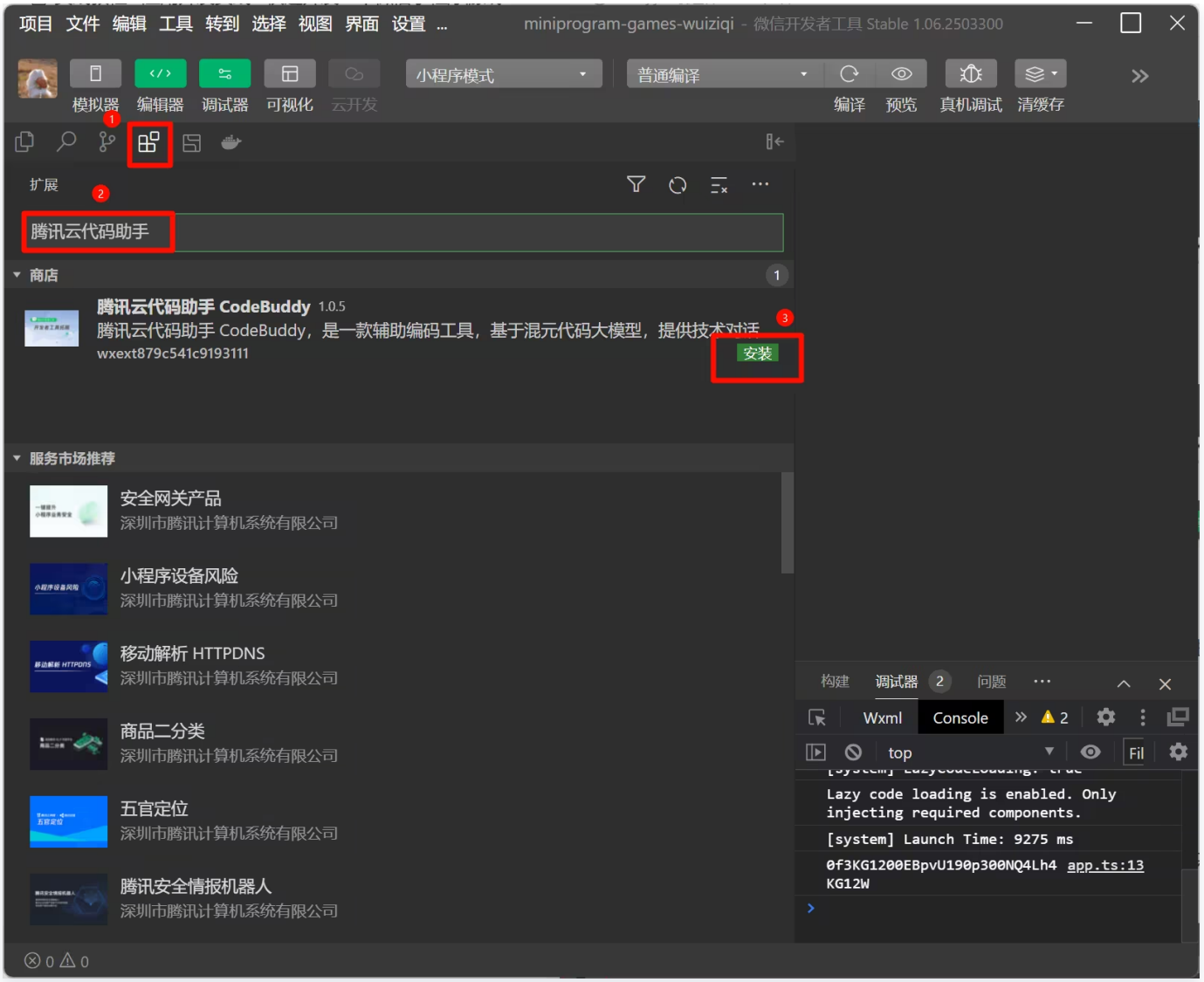
4. 创建小程序项目，单击左侧导航栏中的**小程序**，出现如下页面，填写信息创建小程序项目。



- 项目名称：可自定义。
- 目录：可自定义。
- AppID：如果没有注册小程序账号可以单击**注册**前往进行注册；或者直接单击**测试号**使用测试号来创建。
- 后端服务：因为小程序游戏还不需要后端，所以这里选择了不使用云服务。
- 开发模式：选择**小程序**。
- 初始化：选择**模板**来创建。
- 模板选择：选择 **TS-基础模板**即可。

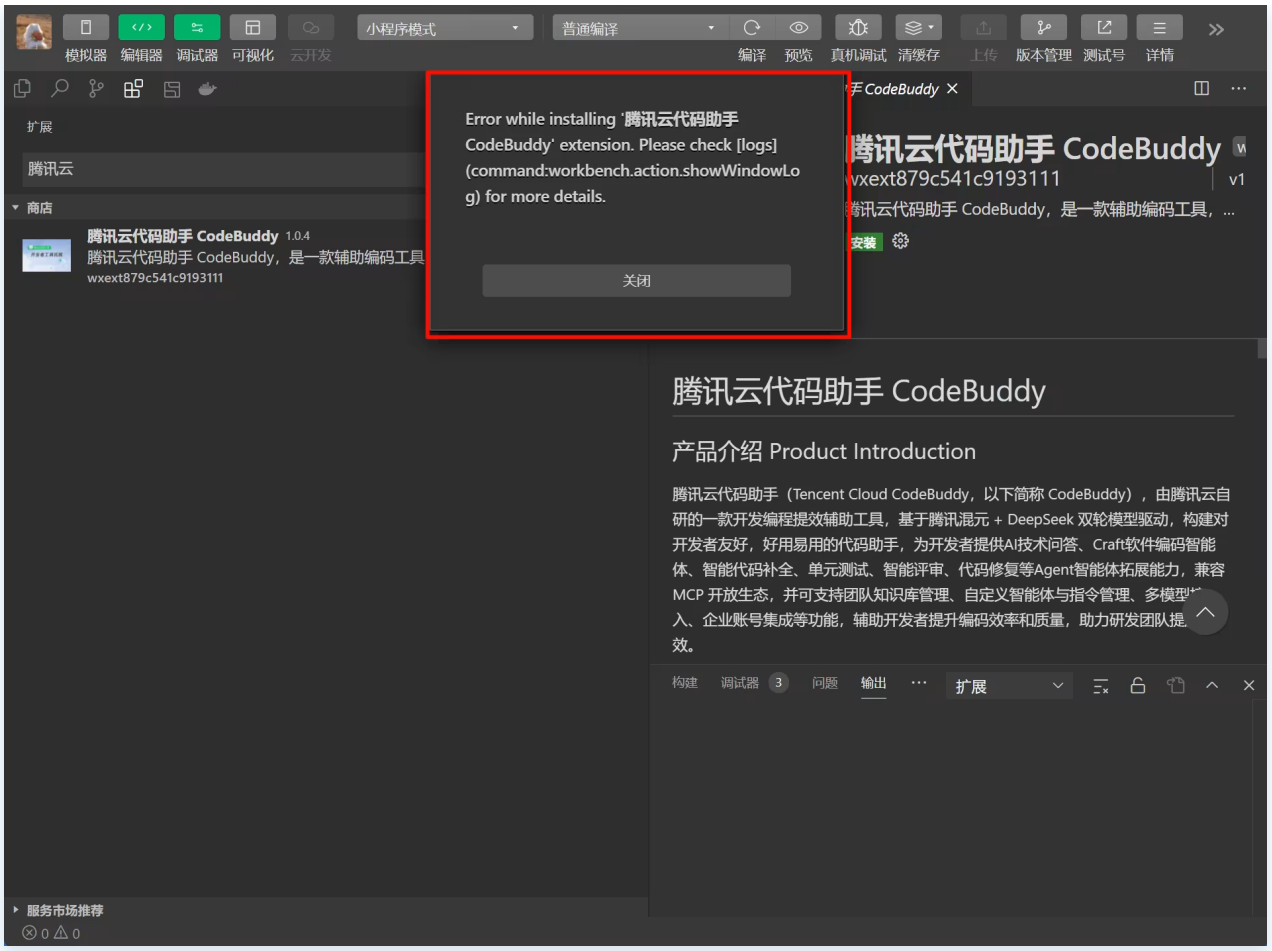
填写完信息后单击**创建**即可。

5. 创建完小程序项目后，在**扩展**中，输入**腾讯云代码助手**或 **CodeBuddy** 安装腾讯云代码助手插件。

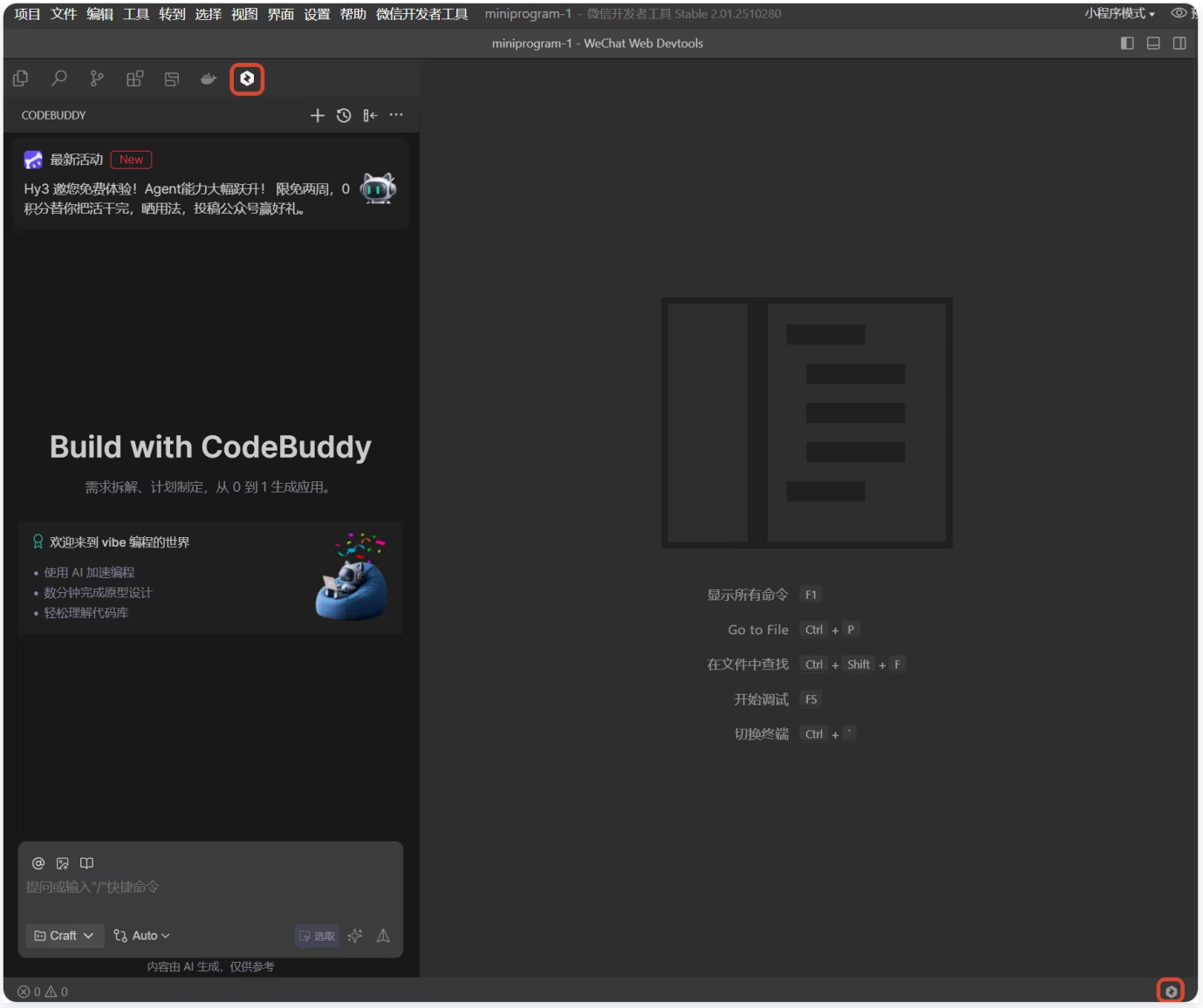


说明：

- 如果搜索不到插件，建议进行清除开发者工具缓存操作：在启动页右上角点击**设置** > **清除缓存** > 勾选所有选项后确认。
- 安装插件完成后，建议重启微信开发者工具；如果出现安装异常，例如下图所示，可参考 不同端功能常见问题 进行解决。

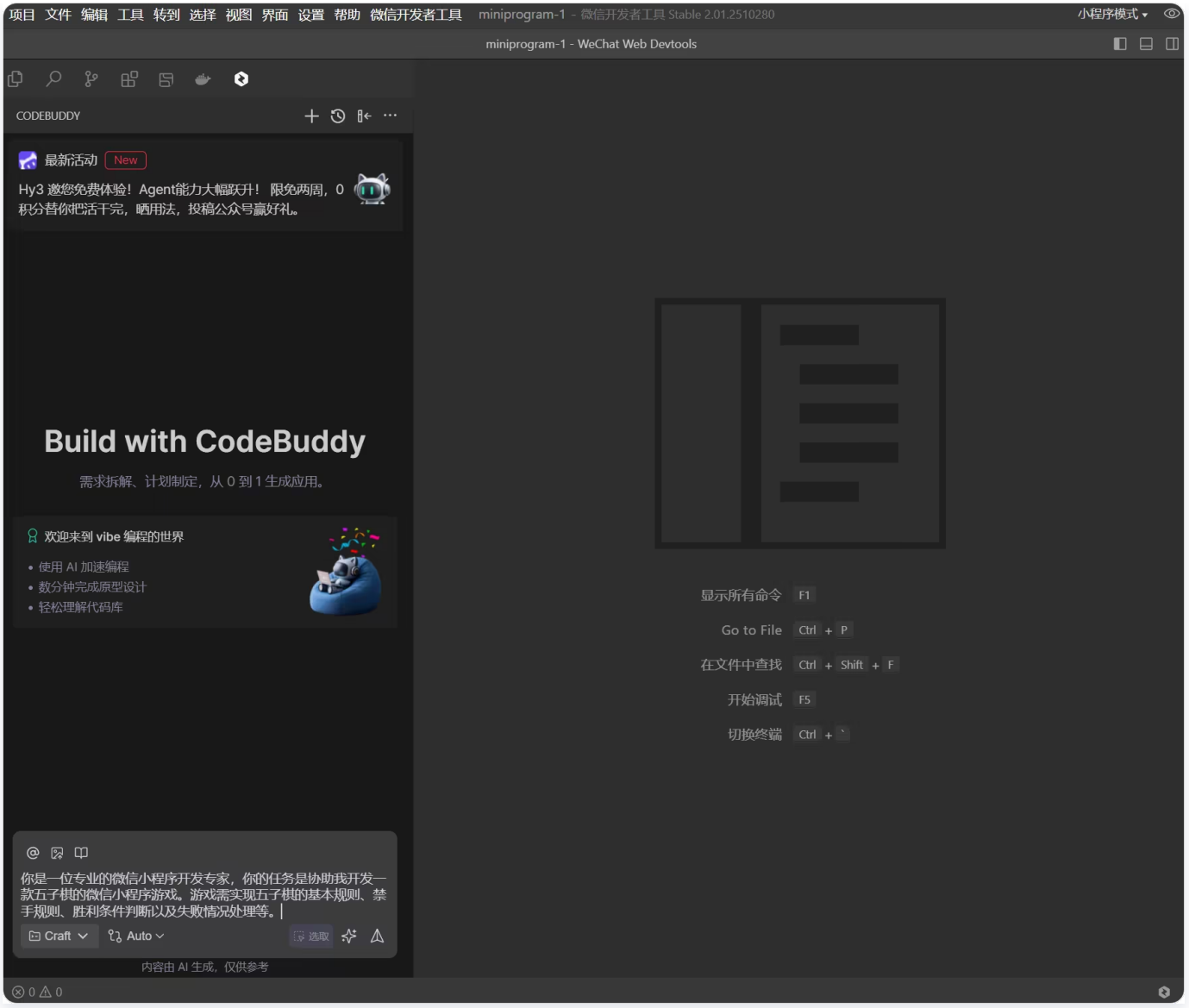


6. 单击 CodeBuddy icon，或底部右下方区域的 icon，即可打开插件，开启编码之旅。



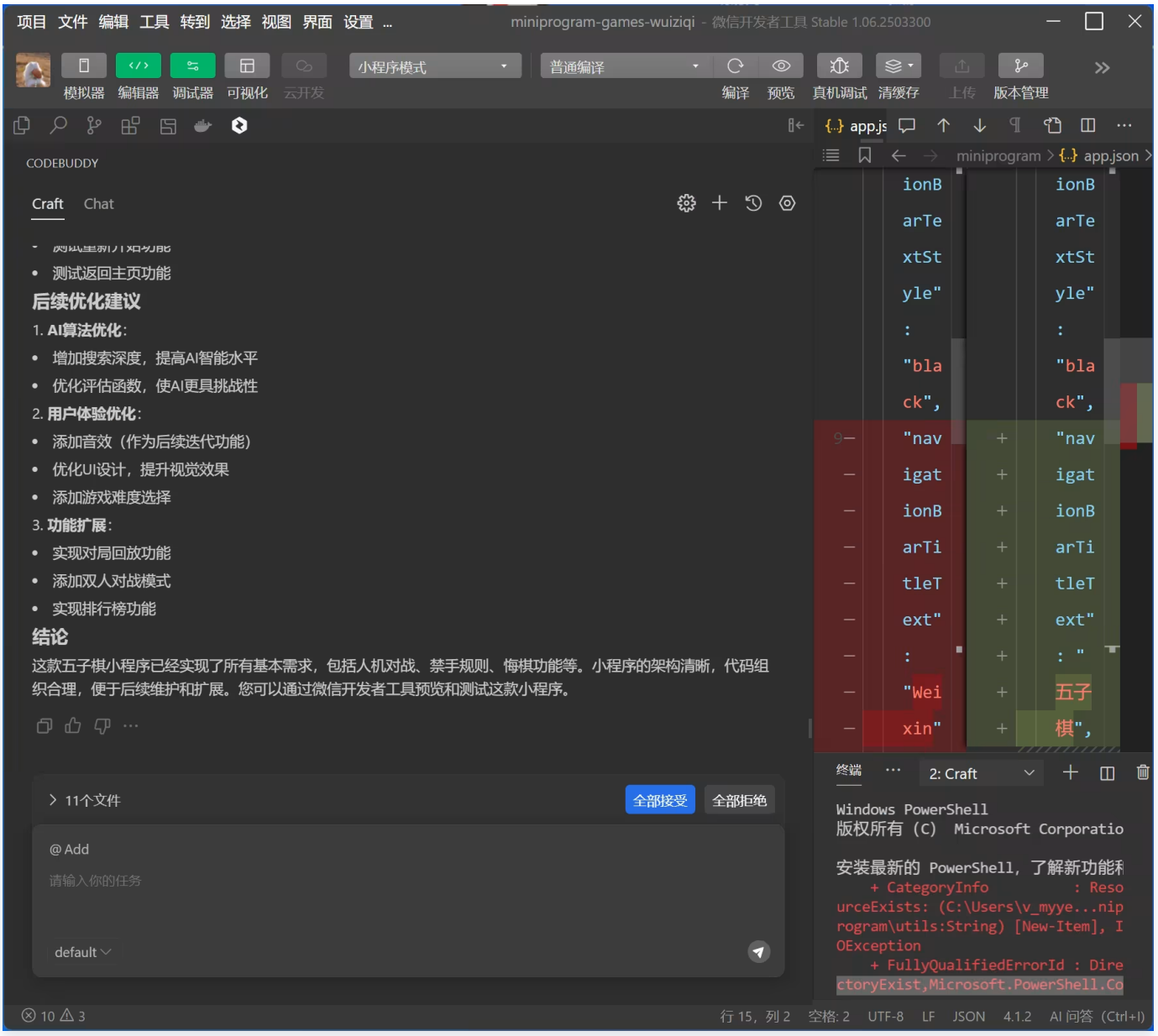
开始开发

1. 输入描述需求提示词。可参考下面的提示词。

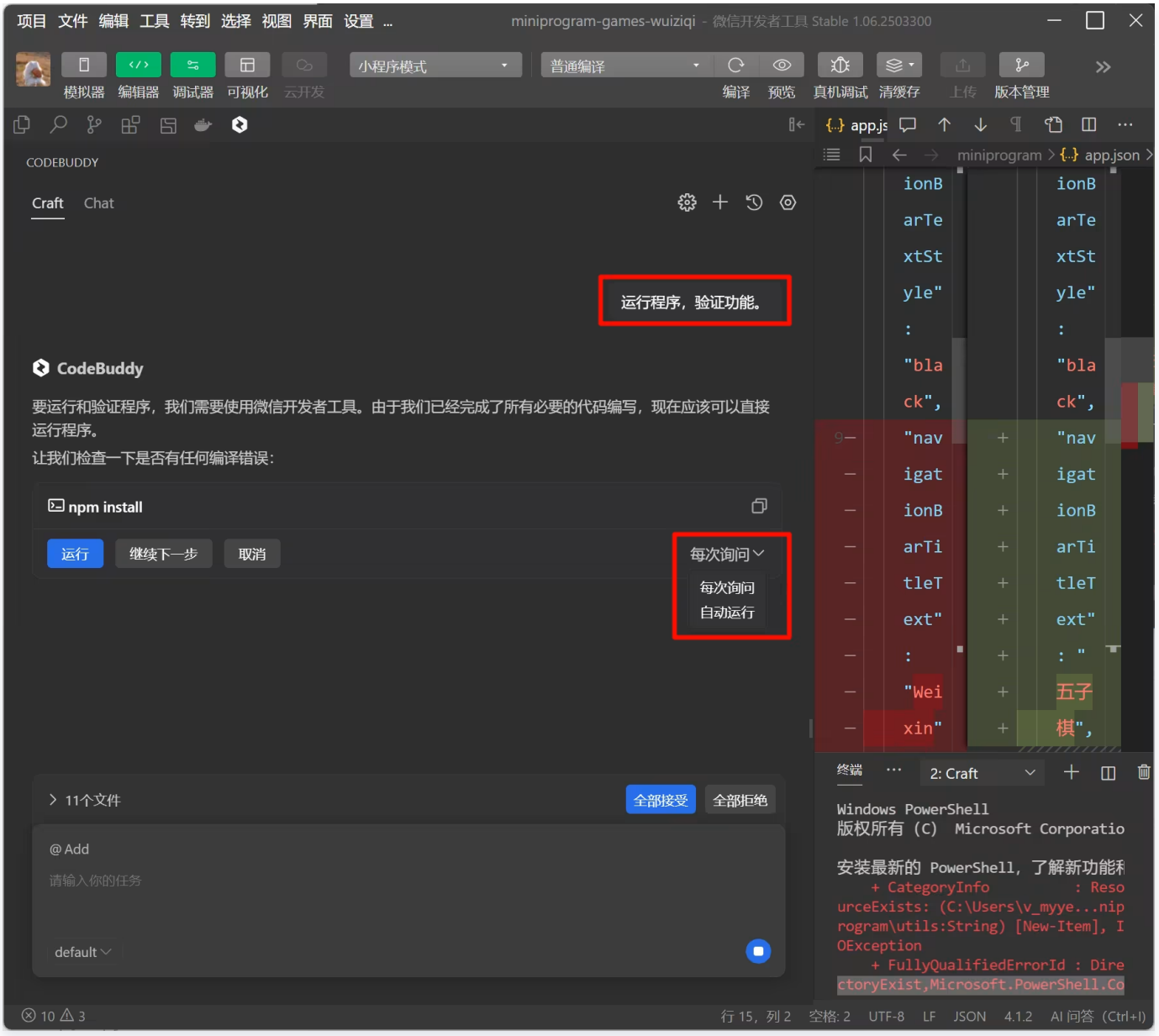


你是一位专业的微信小程序开发专家，你的任务是协助我开发一款五子棋的微信小程序游戏。游戏需实现五子棋的基本规则、禁手规则、胜利条件判断以及失败情况处理等。

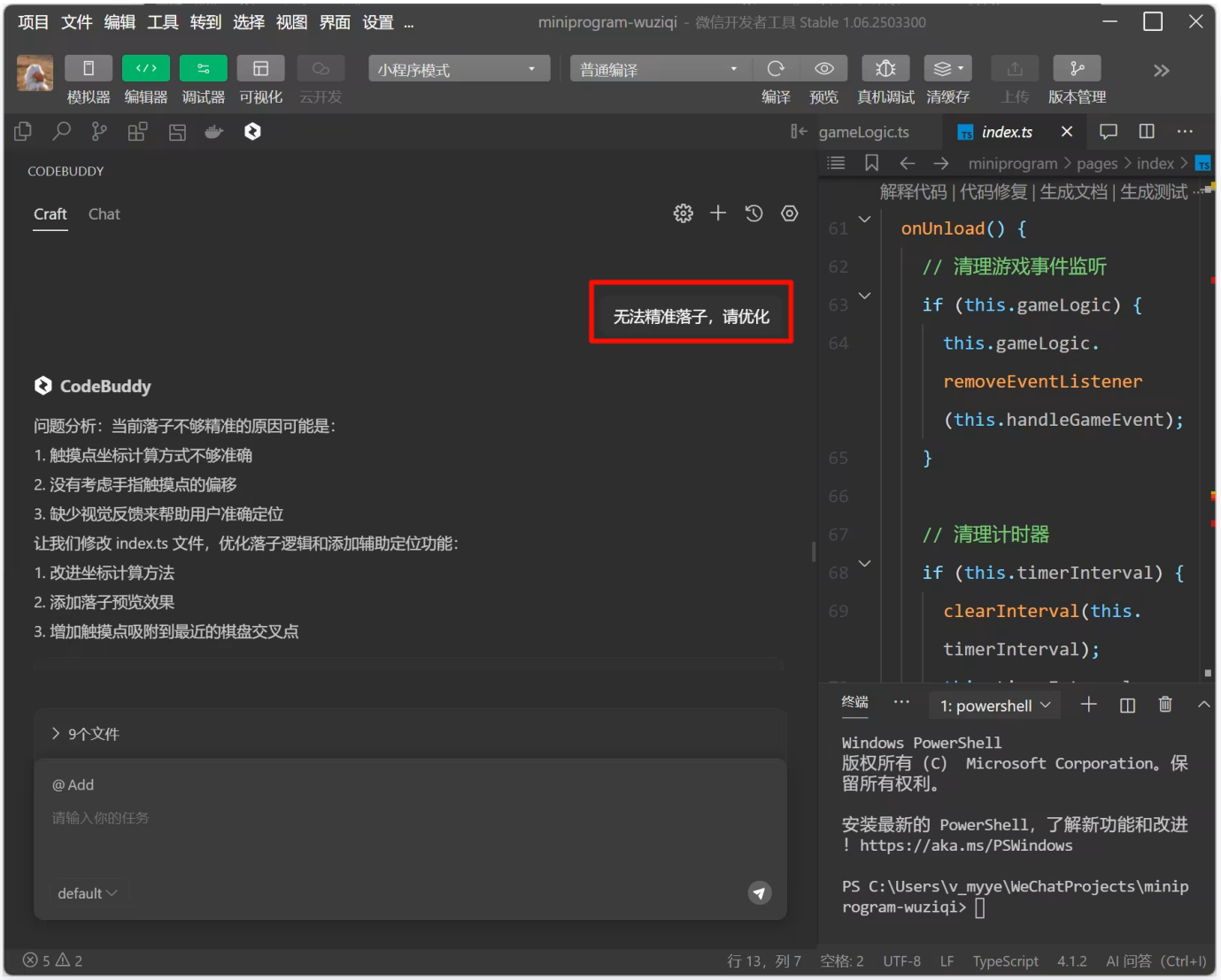
2. 在输入需求描述后，开始执行任务，完成了小程序游戏的开发工作。



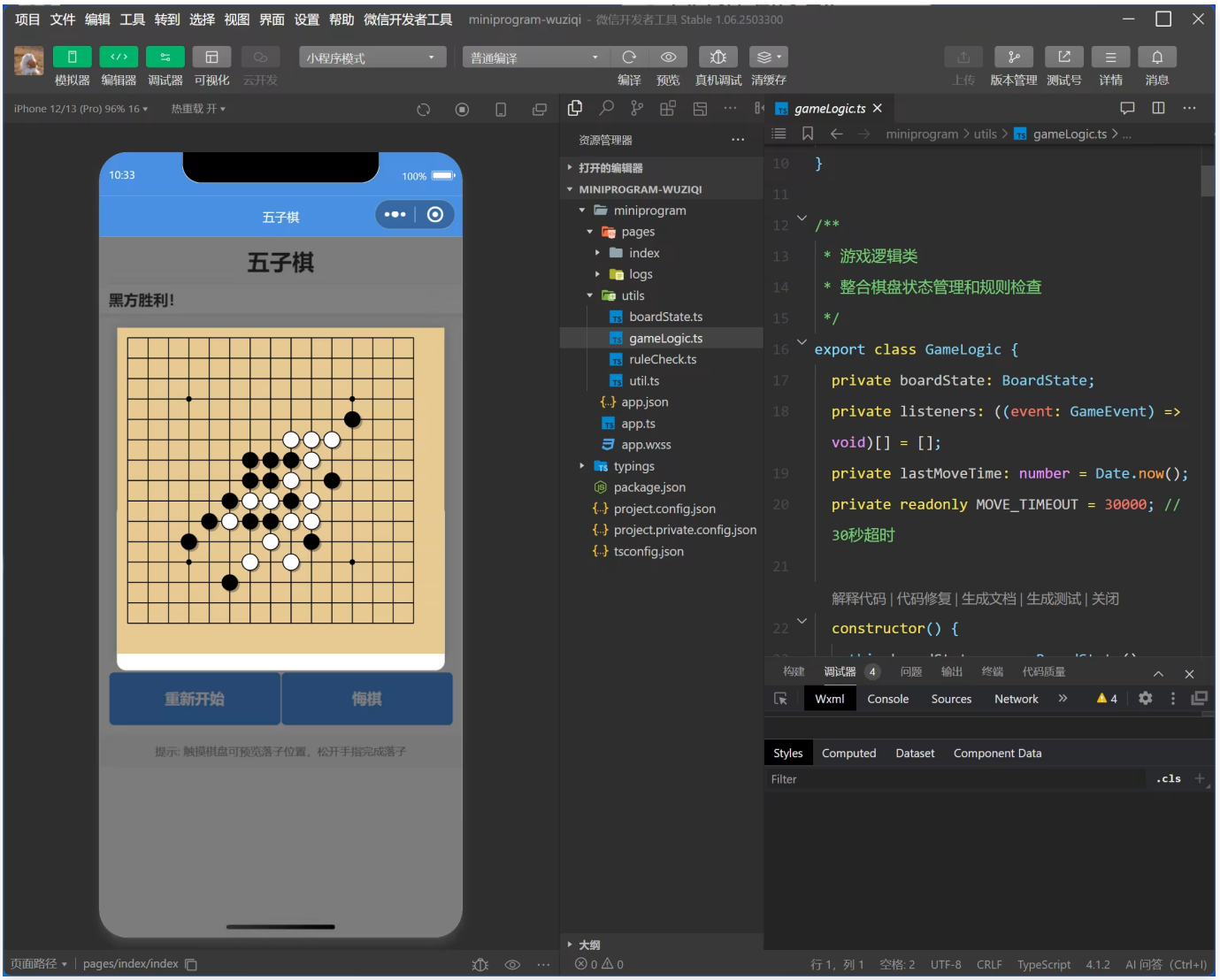
3. 接下来，运行程序，验证程序是否有编译错误。当输入运行程序的需求后，Craft 会向您询问是否运行。这里有两种模式可以选择：**每次询问**和**自动运行**，如果不需要每次向您询问，可以选择设置成**自动运行**，设置后将全局生效，后续的对话将自动运行。



4. 编译运行程序后，发现无法精准落子，此时可以继续输入问题与 Craft 进行对话，让 Craft 继续优化代码。



5. 重新编译运行程序，可以看到五子棋的微信小程序游戏精准落子，功能正常。您还可以继续基于实际需求进行版本迭代。



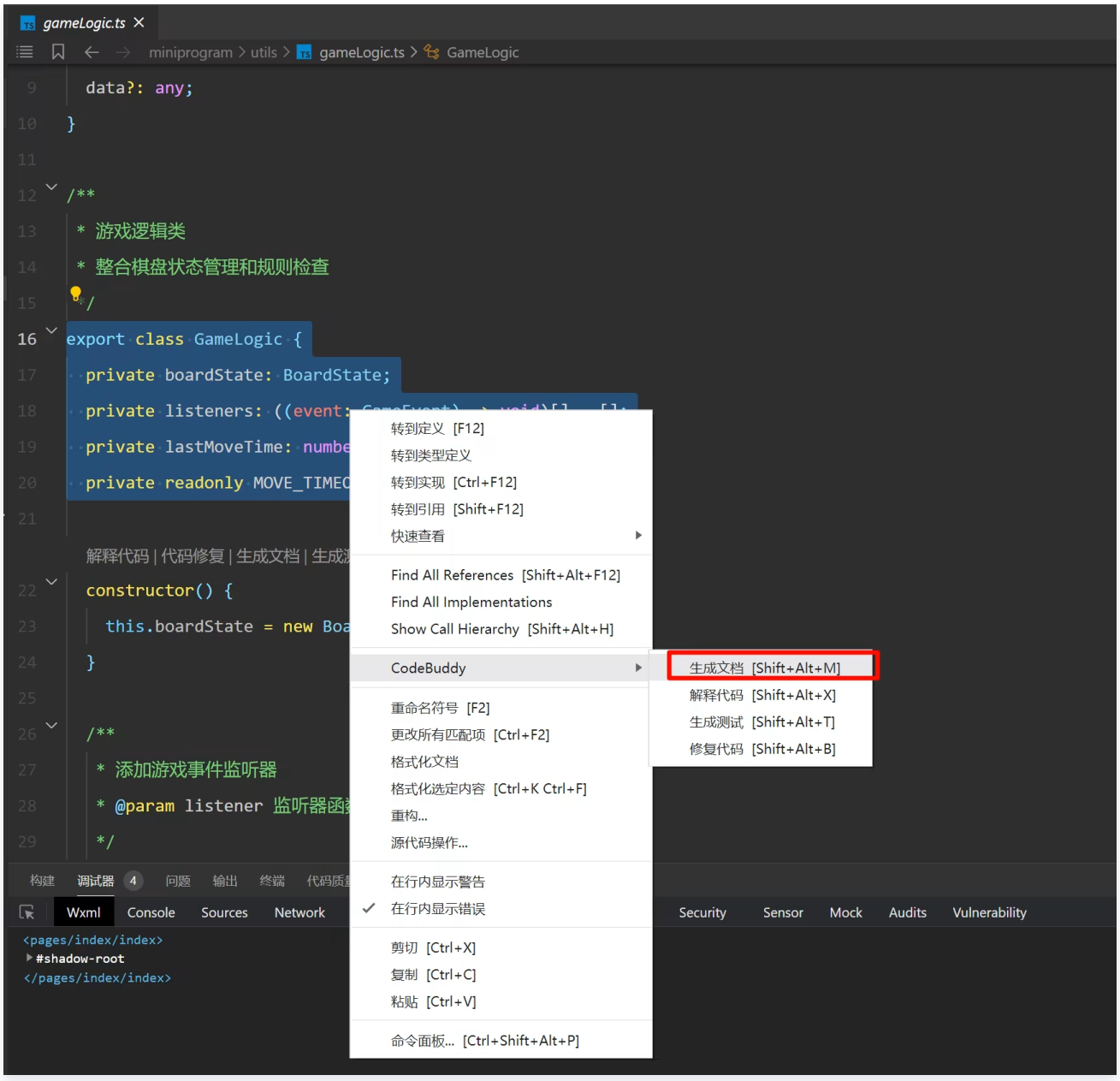
优化代码的艺术

最后，我们来把项目工程优化，然后把核心逻辑解读并重构一下。

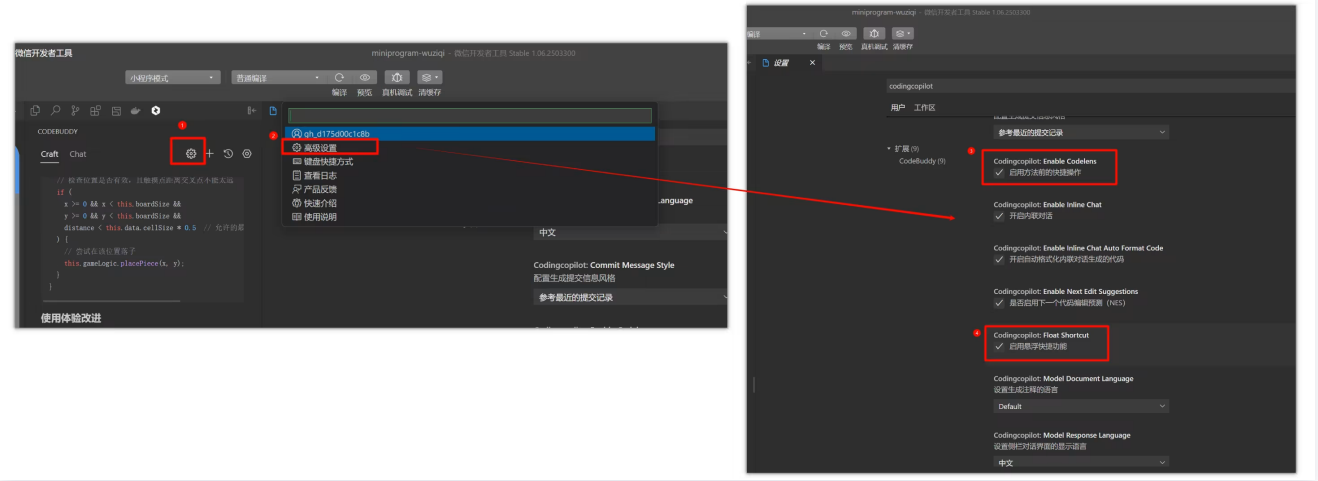
1. 生成文档注释。

   腾讯云代码助手贴心地给出了三种方法生成文档注释：

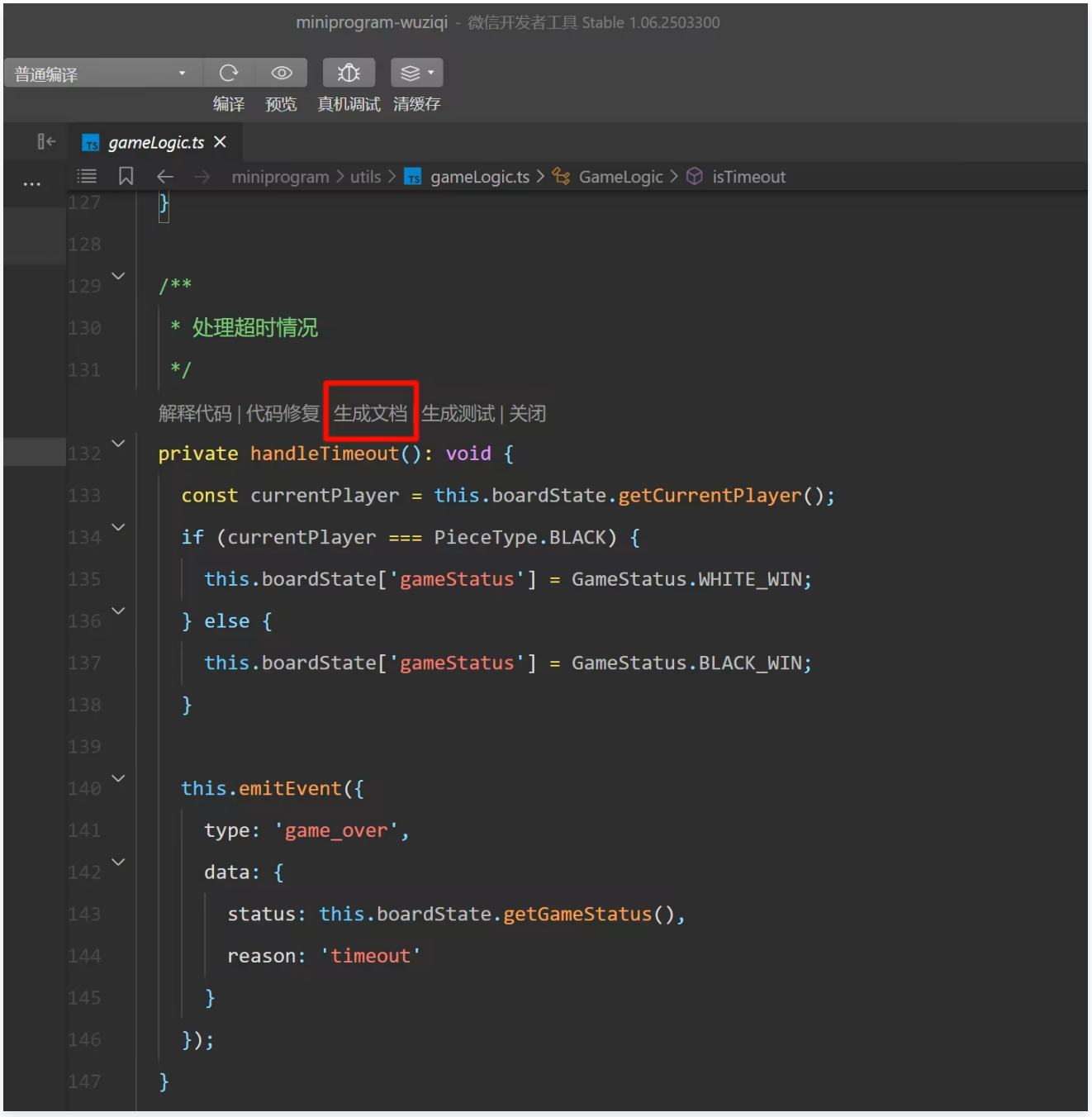
   - 方法一：通过圈选核心代码，右键，选择 **CodeBuddy > 生成文档**，如下：



- **方法二：进入插件的高级设置（下图圈1），开启③和④。**



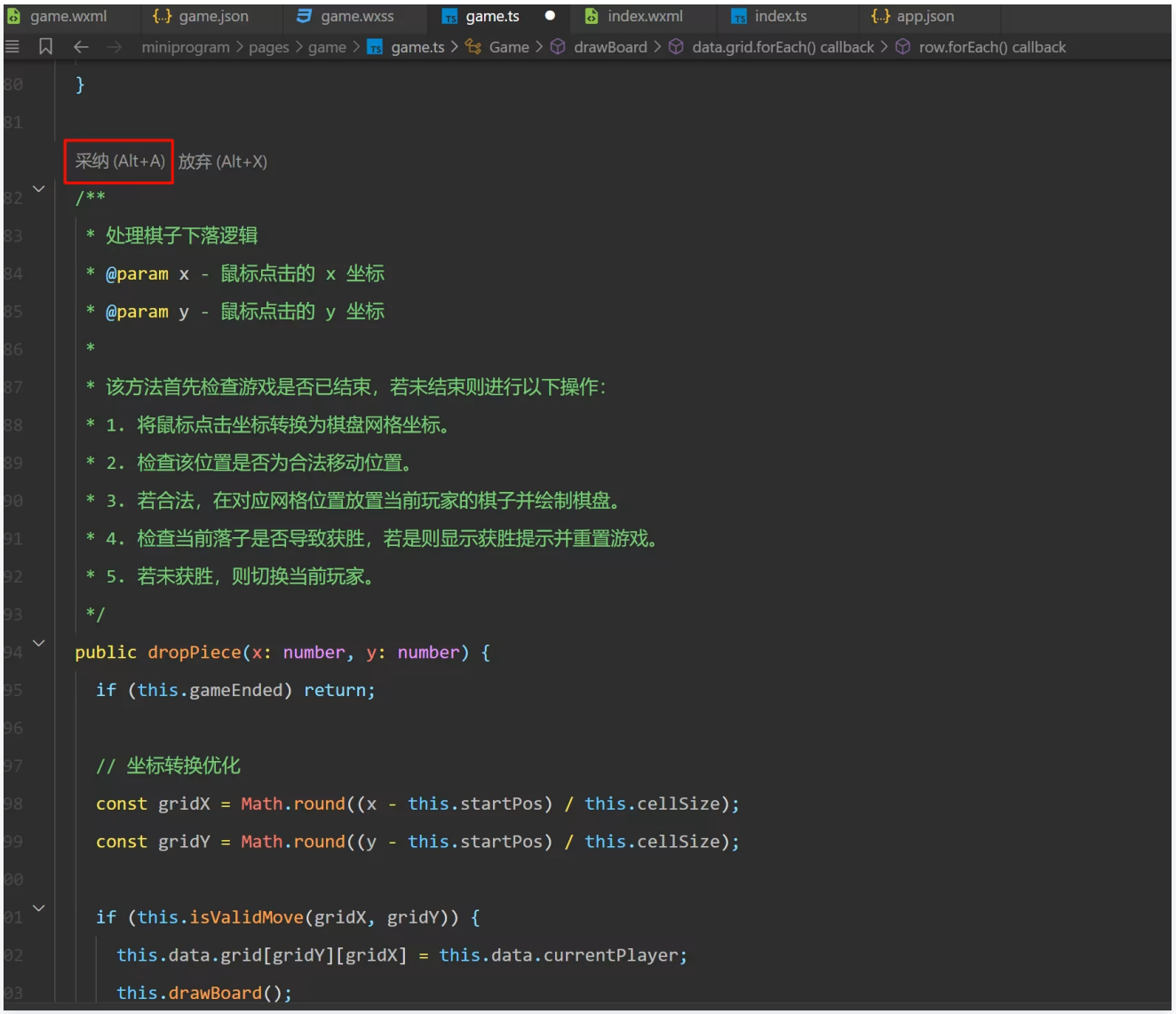
返回代码后，可以在函数头或者鼠标悬浮后出现蓝色提示，单击**生成文档**，如下图：



- 方法三：直接使用快捷键，如下：

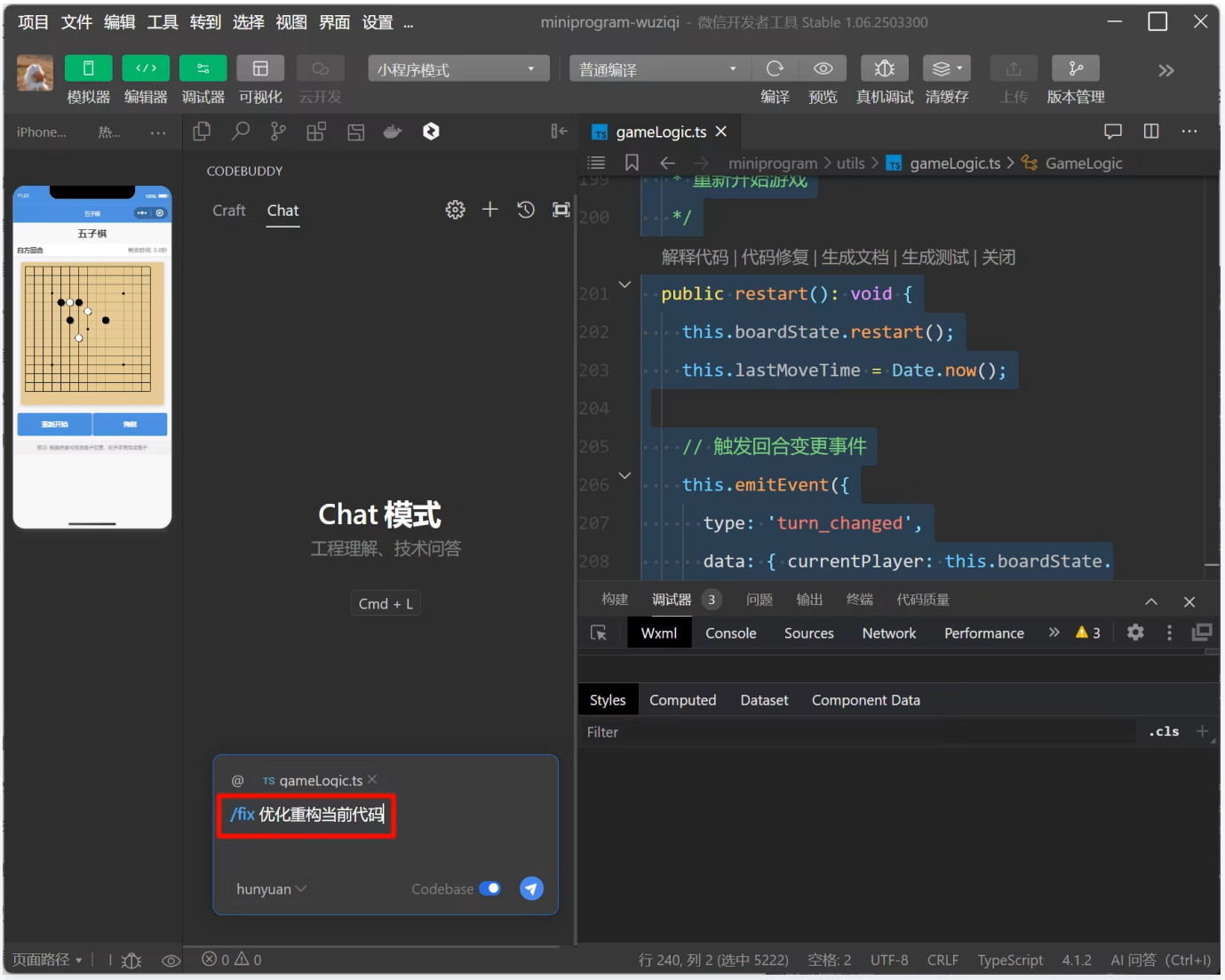
系统	快捷键
Mac	Opt+D
Windows	Shift+Alt+M

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：

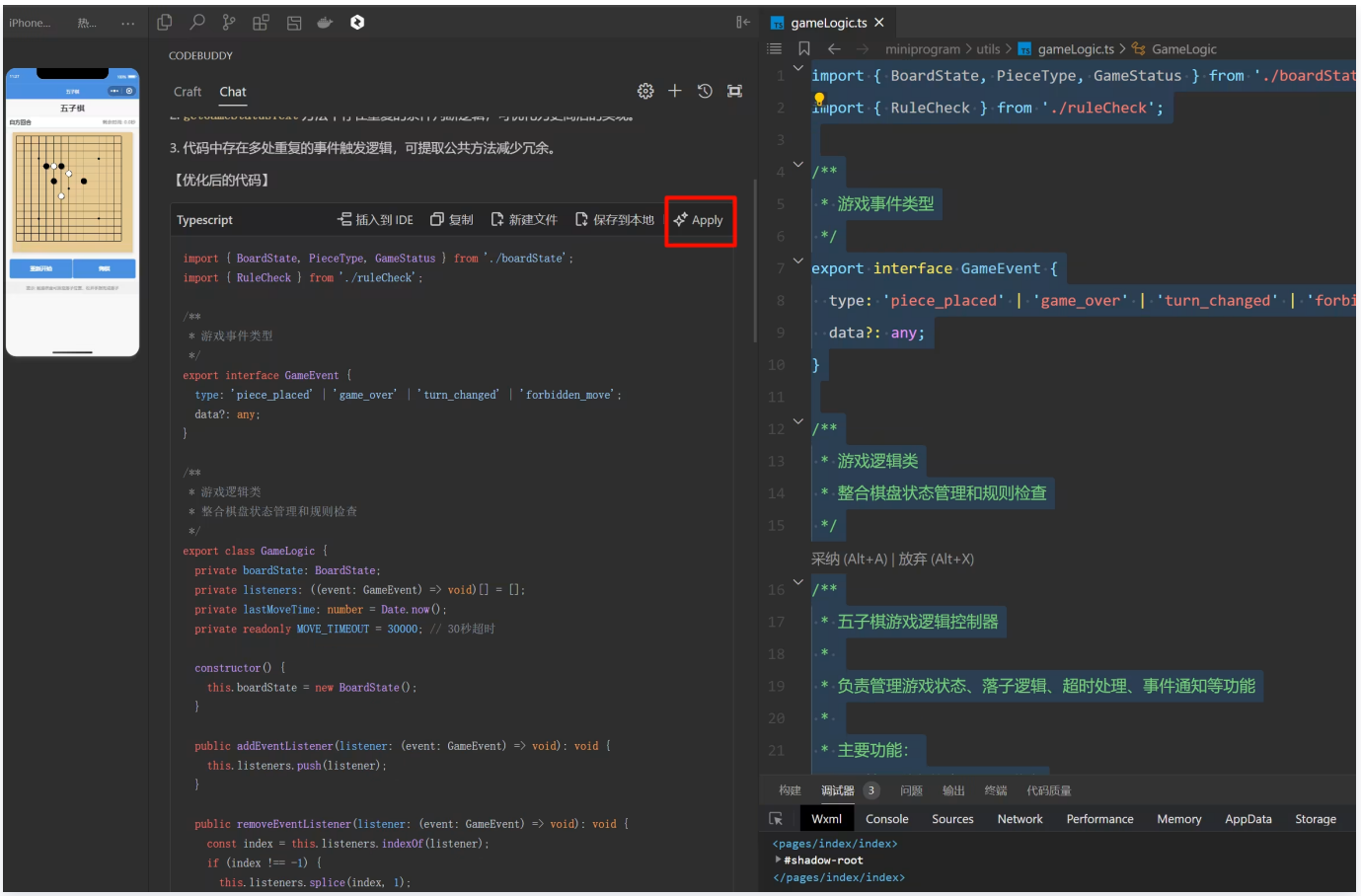


2. 重构代码。

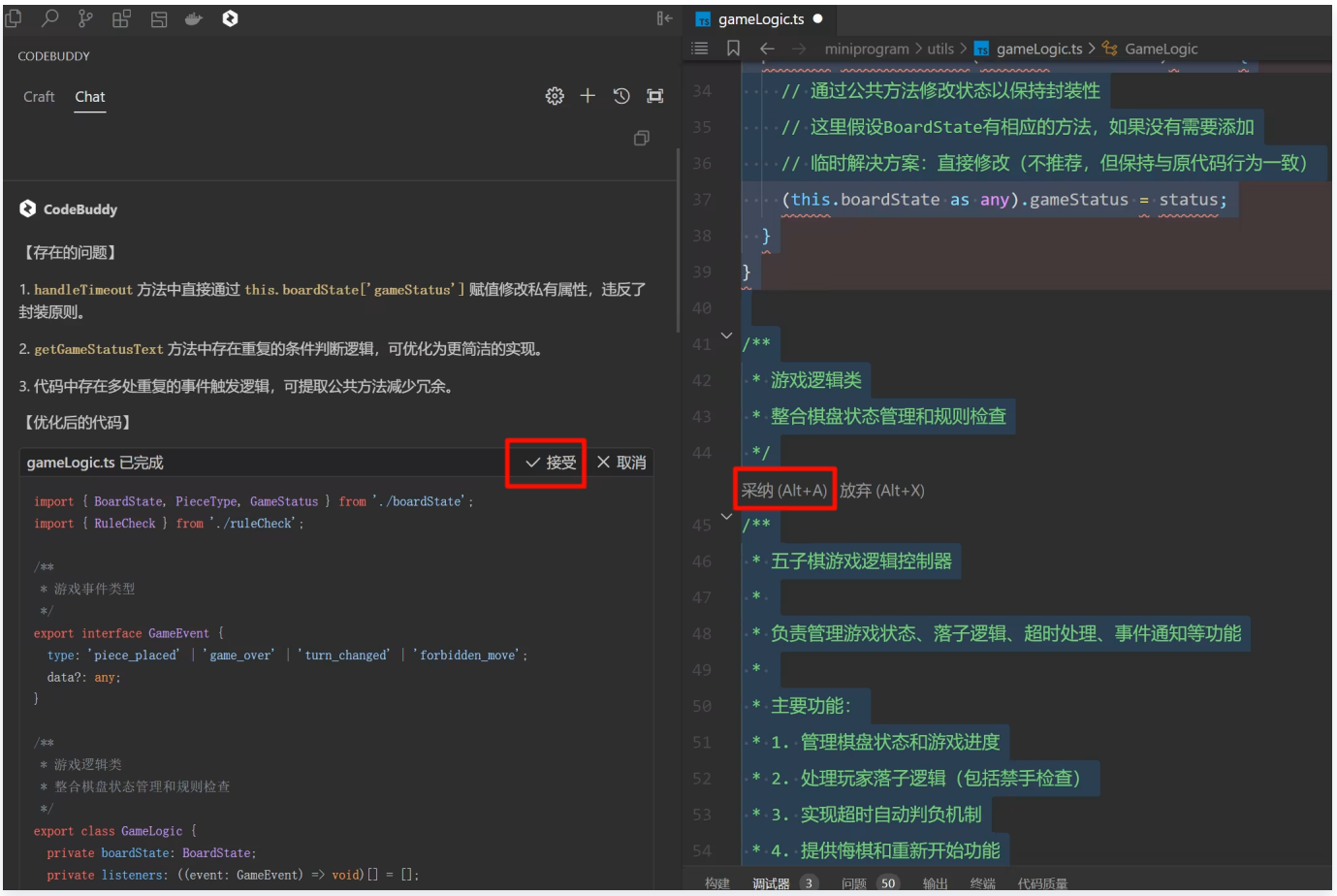
   无论是语法错误，还是逻辑问题，或者是代码需要重构，一个 /fix 就可以帮我们进行 AI 检查。我们可以在对话框中使用 /fix 进行提问。



输出如下，我们可以选择应用代码。

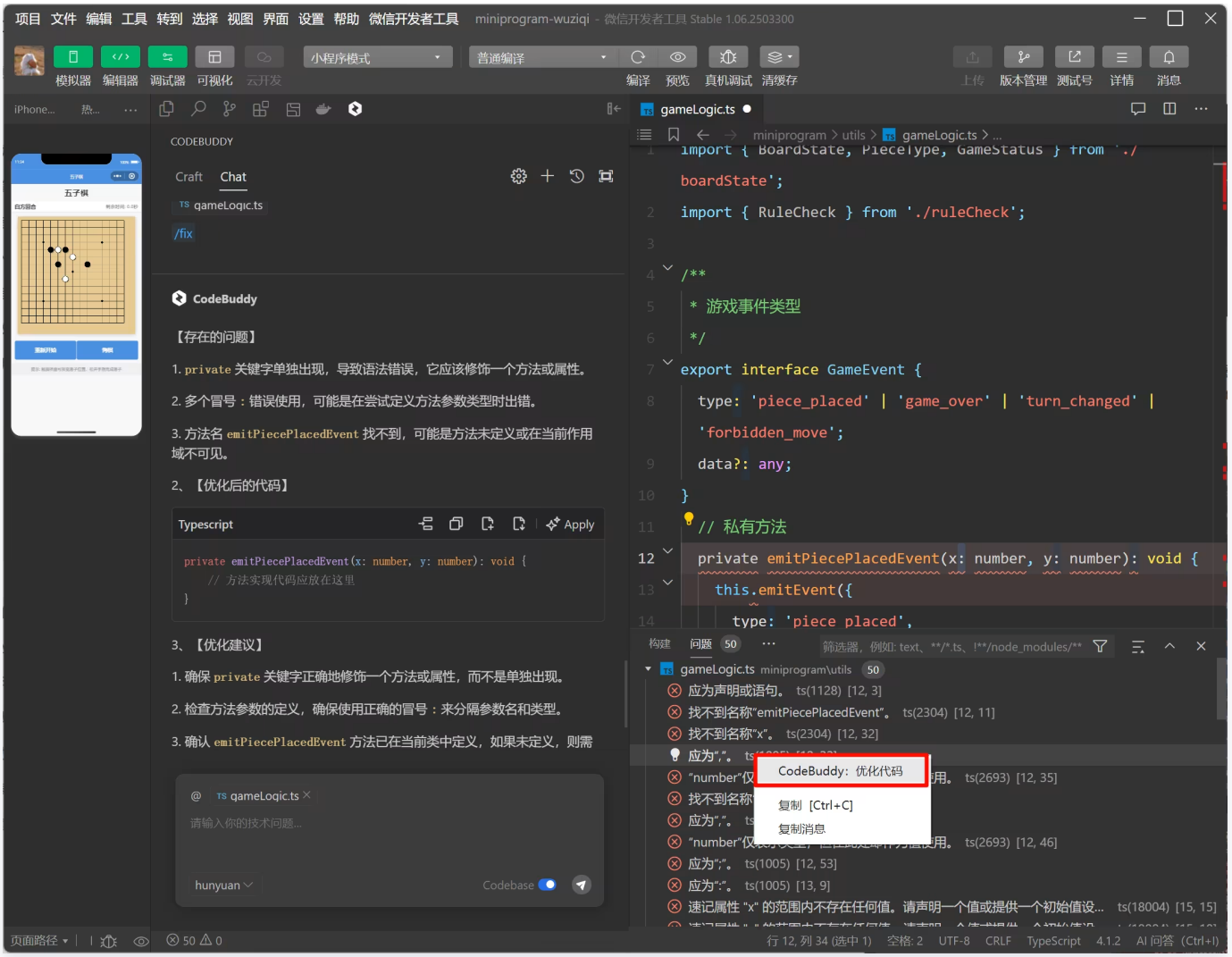


代码助手提供了智能插入的方法，通过大模型为其生成合并的 Diff View 预览效果，如下图，清晰很多了。代码助手在如下视图下可以选择部分采纳或者全部采纳：



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，我们可以鼠标右键单击错误信息，选择 CodeBuddy：优化代码，它能正确地为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，给出优化建议和代码，通过选择对话代码区域的 Apply 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，CodeBuddy 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，让小白也能快速实现自己的个人应用，做自己的产品经理。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 CodeBuddy 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 CodeBuddy 更容易理解需求，起到一个事半功倍的效果。

版权所有：腾讯云计算（北京）有限责任公司

具体功能实践
跨文件能力扩展上下文

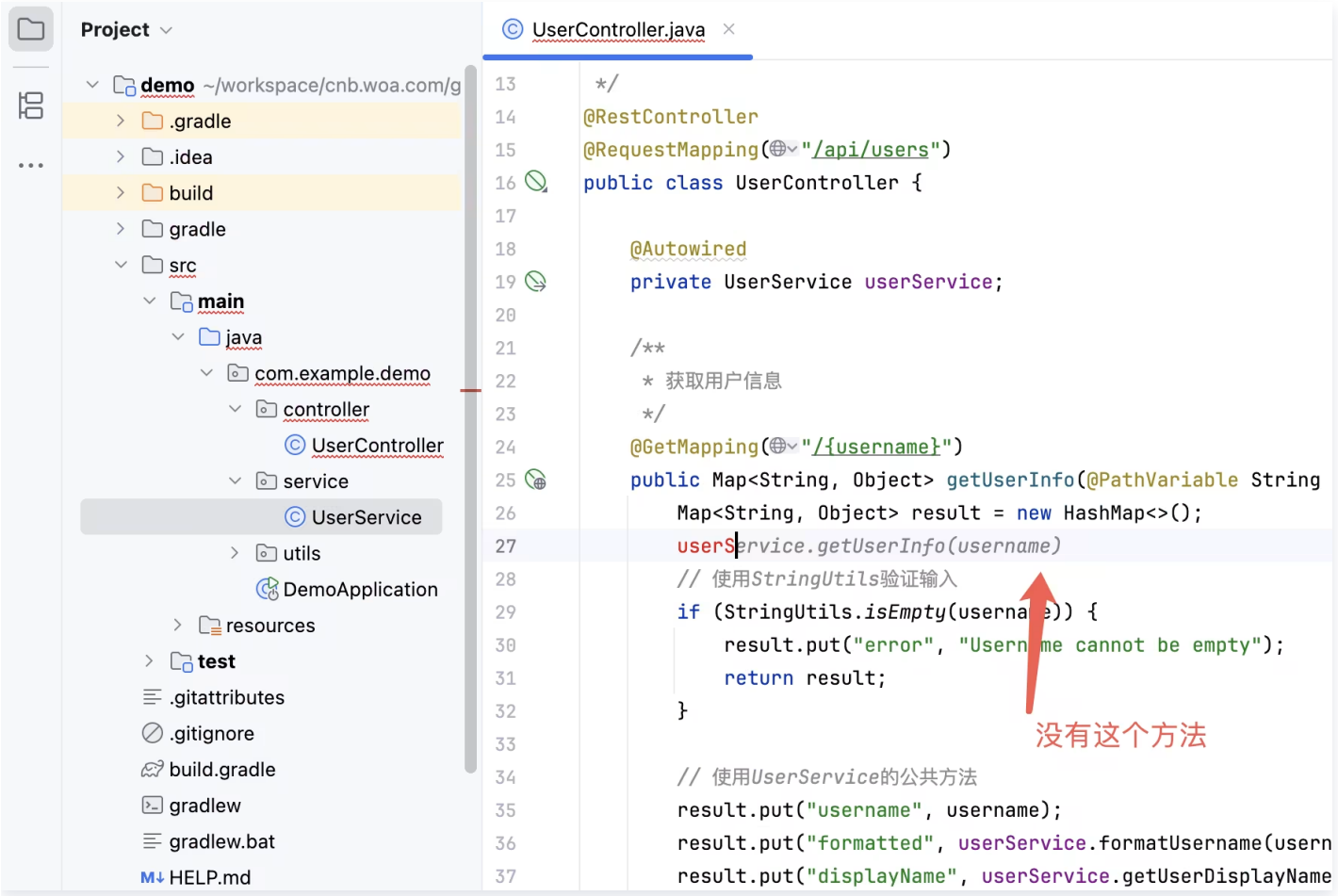
最近更新时间：2026-07-17 17:54:32

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 CodeBuddy 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。

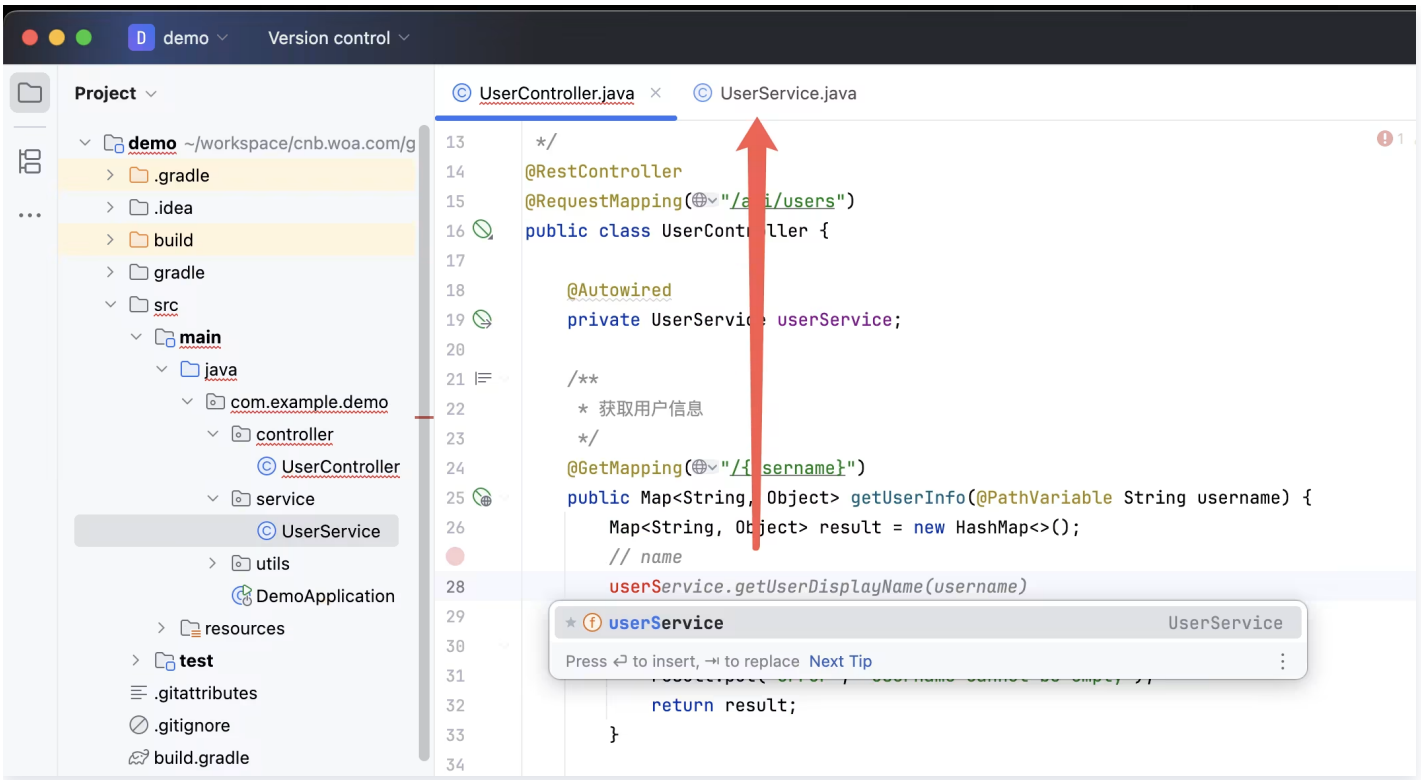
如何让补全感知更多

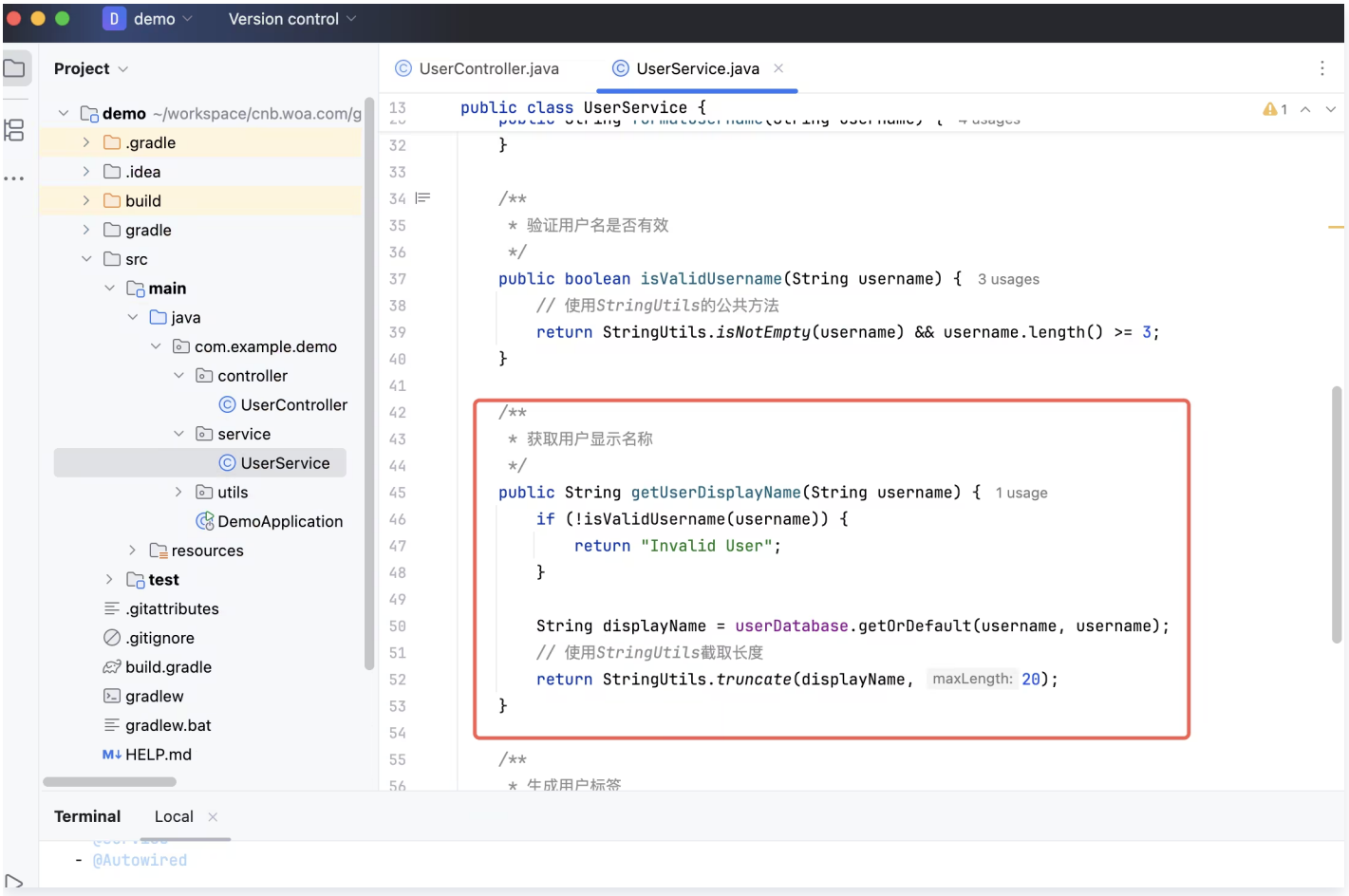
操作指南

如下场景，因为没有把 UserService 加入上下文，所以没有提示有效的内容，出现了 UserService 内没有的方法。



打开文件后可以提示正确的方法：


demo
Version control
Project
UserController.java
UserService.java
demo ~/workspace/cnb.woa.com/g
.gradle
.idea
build
gradle
src
main
java
com.example.demo
controller
UserController
service
UserService
utils
DemoApplication
resources
test
.gitattributes
.gitignore
build.gradle
@RestController
@RequestMapping("/api/users")
public class UserController {
@Autowired
private UserService userService;
/**
* 获取用户信息
*/
@GetMapping("/{username}")
public Map<String, Object> getUserInfo(@PathVariable String username) {
Map<String, Object> result = new HashMap<>();
// name
userService.getUserDisplayName(username)
userService UserService
Press ⏎ to insert, ⇥ to replace Next Tip
return result;
}


demo
Version control
Project
UserController.java
UserService.java
public class UserService {
}
/**
* 验证用户名是否有效
*/
public boolean isValidUsername(String username) { 3 usages
// 使用StringUtils的公共方法
return StringUtils.isNotEmpty(username) && username.length() >= 3;
}
/**
* 获取用户显示名称
*/
public String getUserDisplayName(String username) { 1 usage
if (!isValidUsername(username)) {
return "Invalid User";
}
String displayName = userDatabase.getOrDefault(username, username);
// 使用StringUtils截取长度
return StringUtils.truncate(displayName, maxLength: 20);
}
/**
Terminal
Local
@Autowired

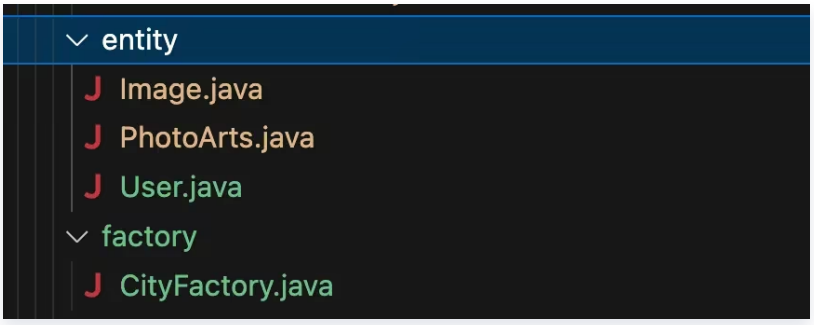
原理介绍

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：

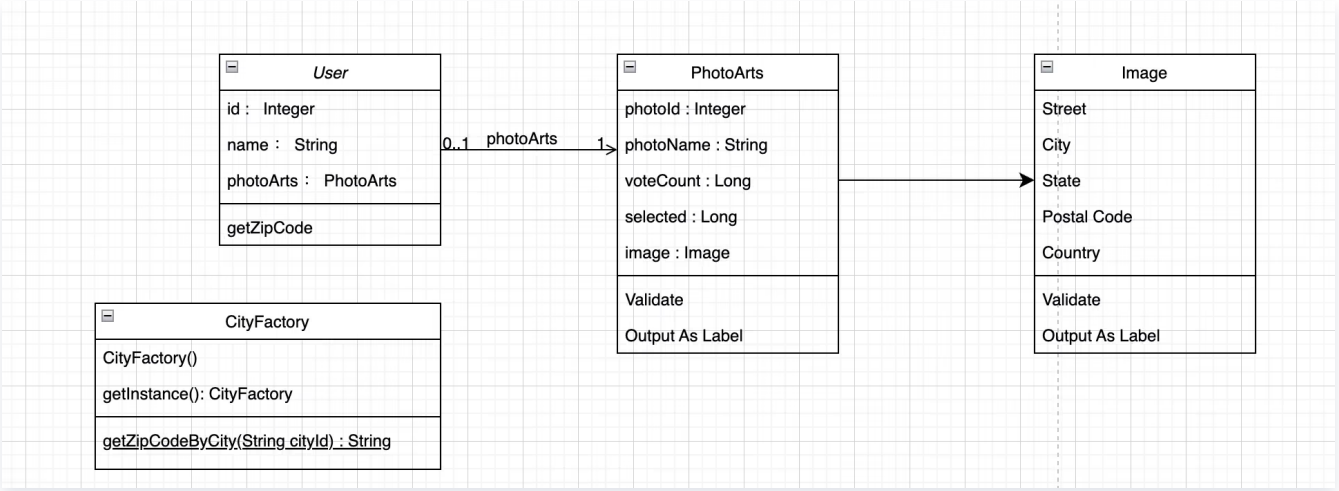
```
<reponame>reponame + <neighbor><filename>file_name<codeblock>code_block
* (n) +
<filename>filename<fim_prefix>code_prefix<fim_suffix>code_suffix<fim_mid
dle>code_middle<endoftext>
```

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：在文件中声明对象关联关系后，验证补全是否能自动引用其他文件中的函数

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： 声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地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。扩展上下文的窗口， 引入更多的相关联的文件，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。

版权所有：腾讯云计算（北京）有限责任公司

代码助手与终端的结合应用实践

最近更新时间：2026-07-20 19:49:32

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，从而帮助程序员在终端运行出现报错的场景下快速查找到问题解决方案。本文旨在介绍 CodeBuddy，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

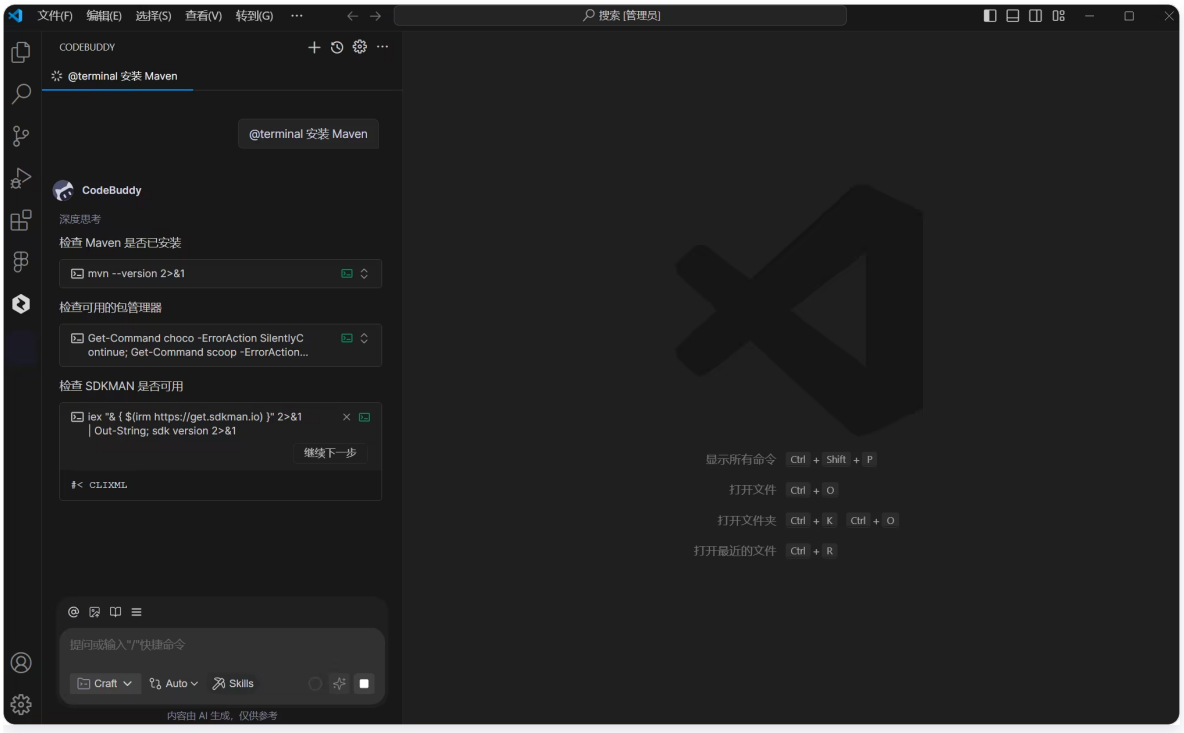
代码助手支持以下 agent 模式：

- @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

直接在对话框中输入 @terminal，例如 @terminal 安装 Maven，如下图，将结合终端环境给出对应的操作指令或解决方案。



单元测试应用实践

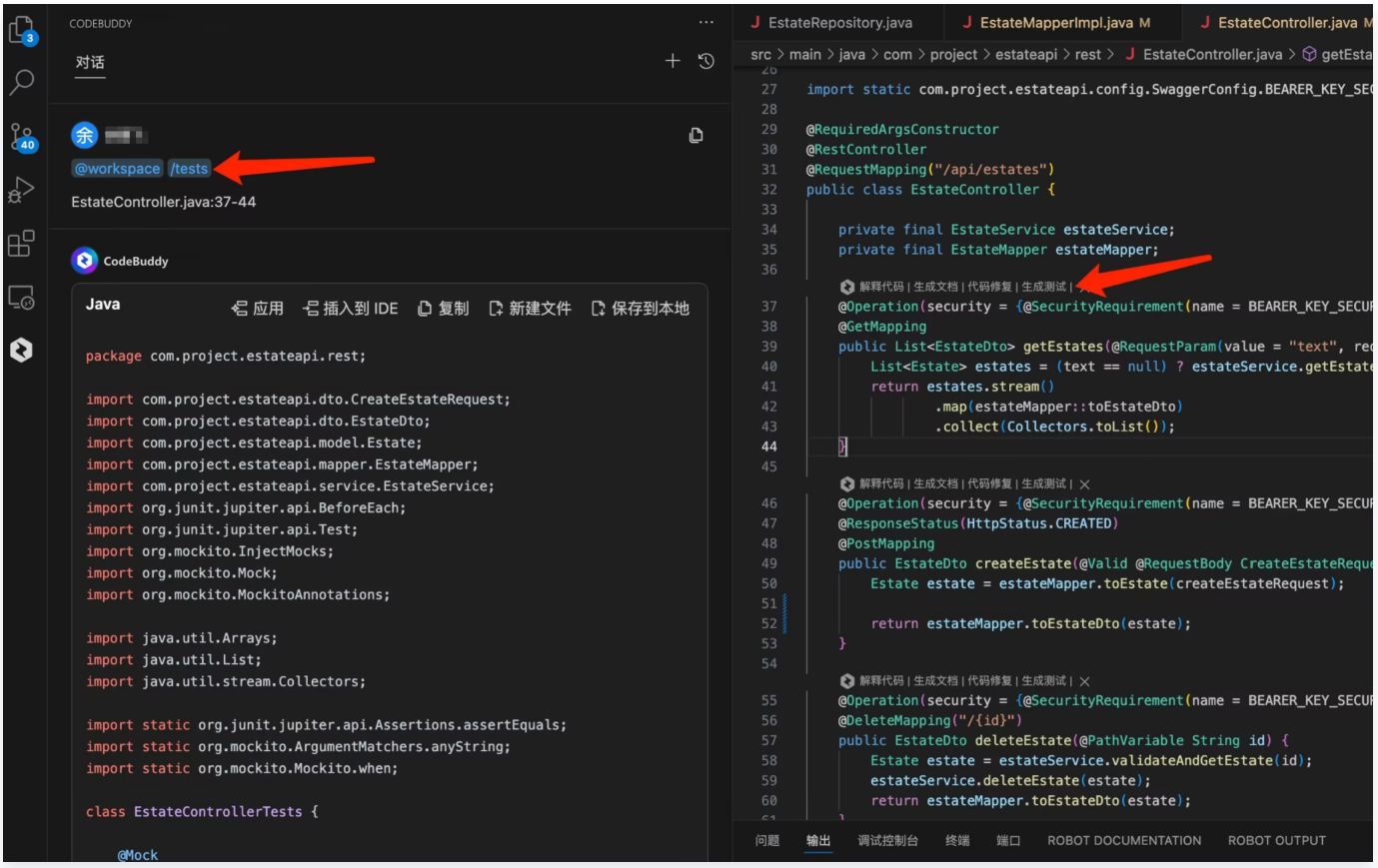
最近更新时间：2026-07-16 21:20:02

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。
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。

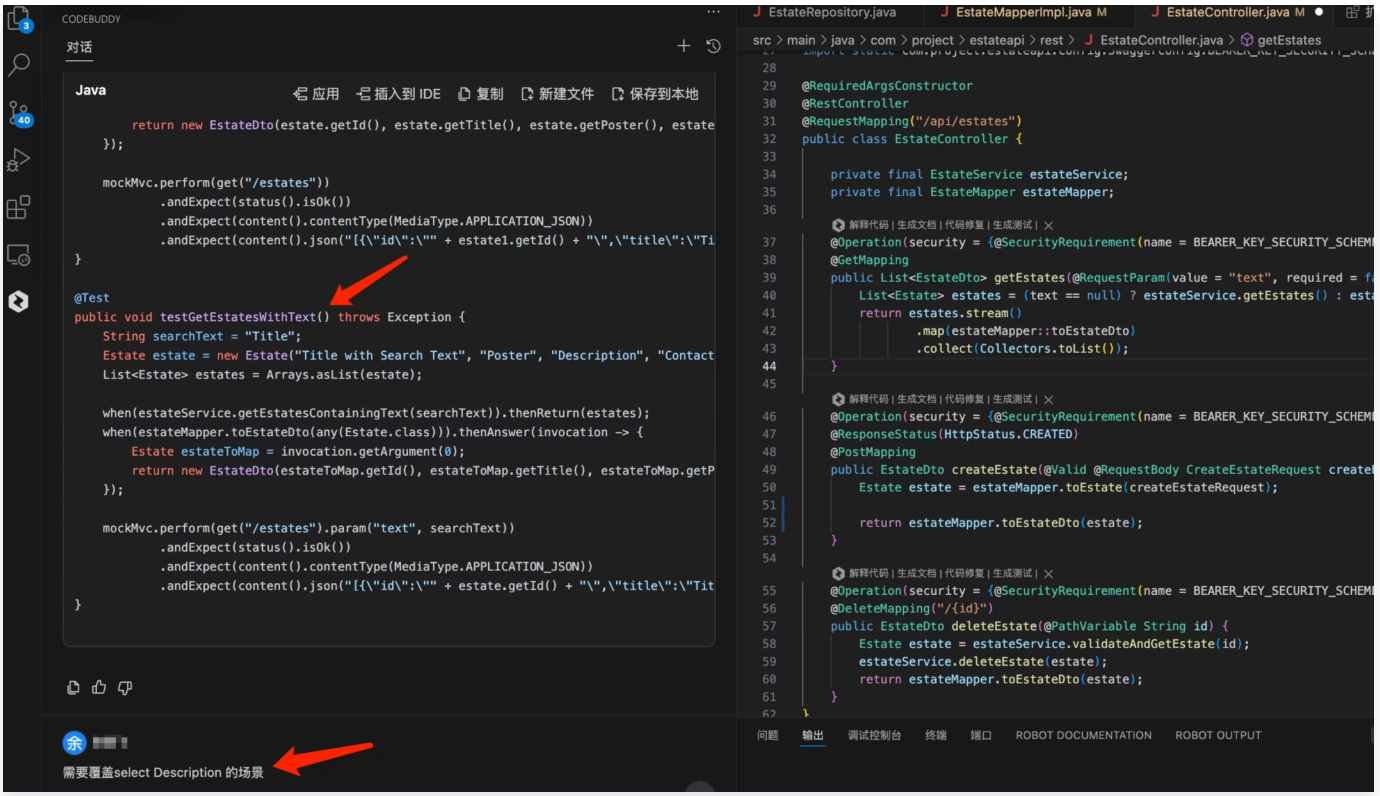
   腾讯云代码助手基于上下文理解，能够自动推荐最可能的代码片段，包括方法调用、变量声明、循环结构等。此外，它还能根据现有代码结构，自动生成对应的单元测试案例，确保代码变更时功能的稳定性。



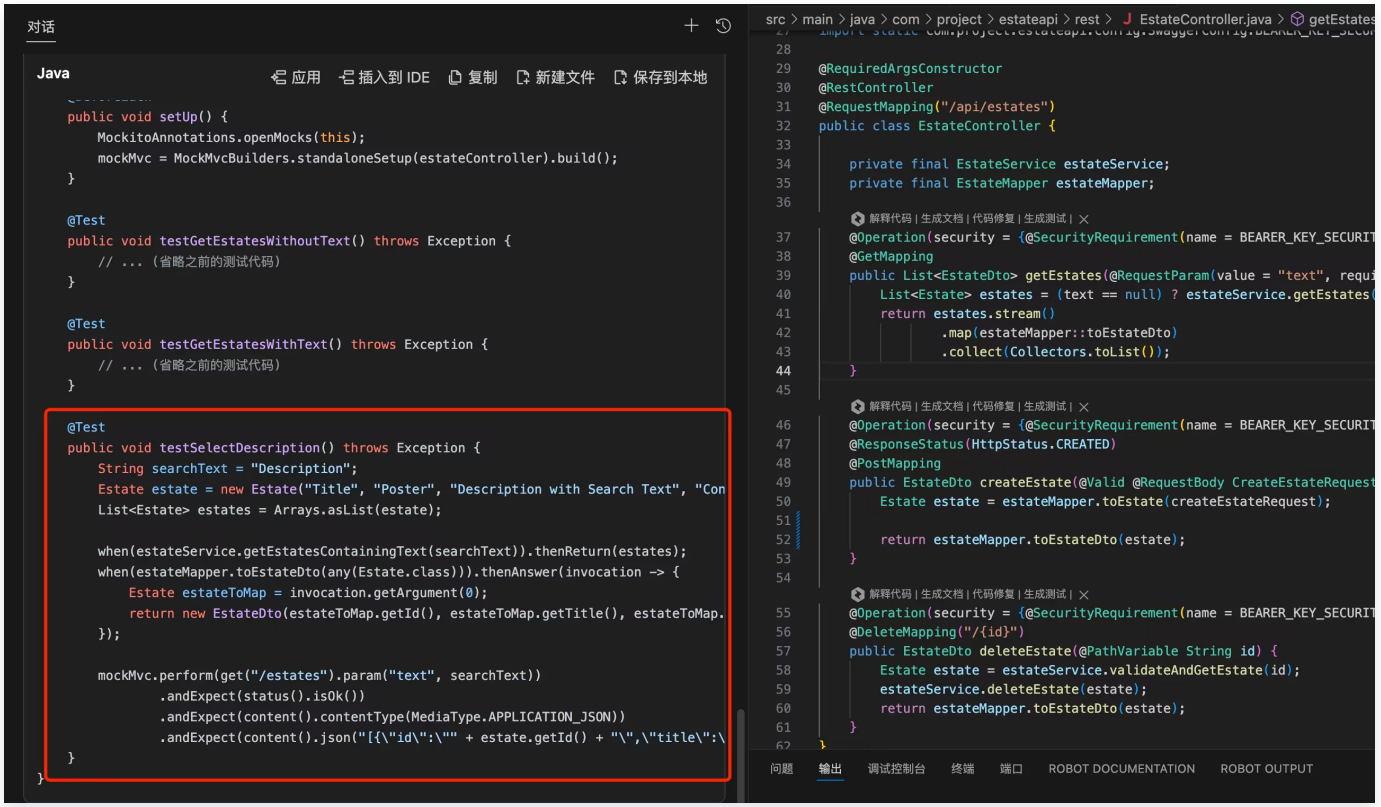
   如上图所示，可以通过对话框中的 /test 生成单元测试，或者在 IDE 编码区域中使用每个功能方法的快捷键**生成测试**来生成该方法的单元测试。

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：

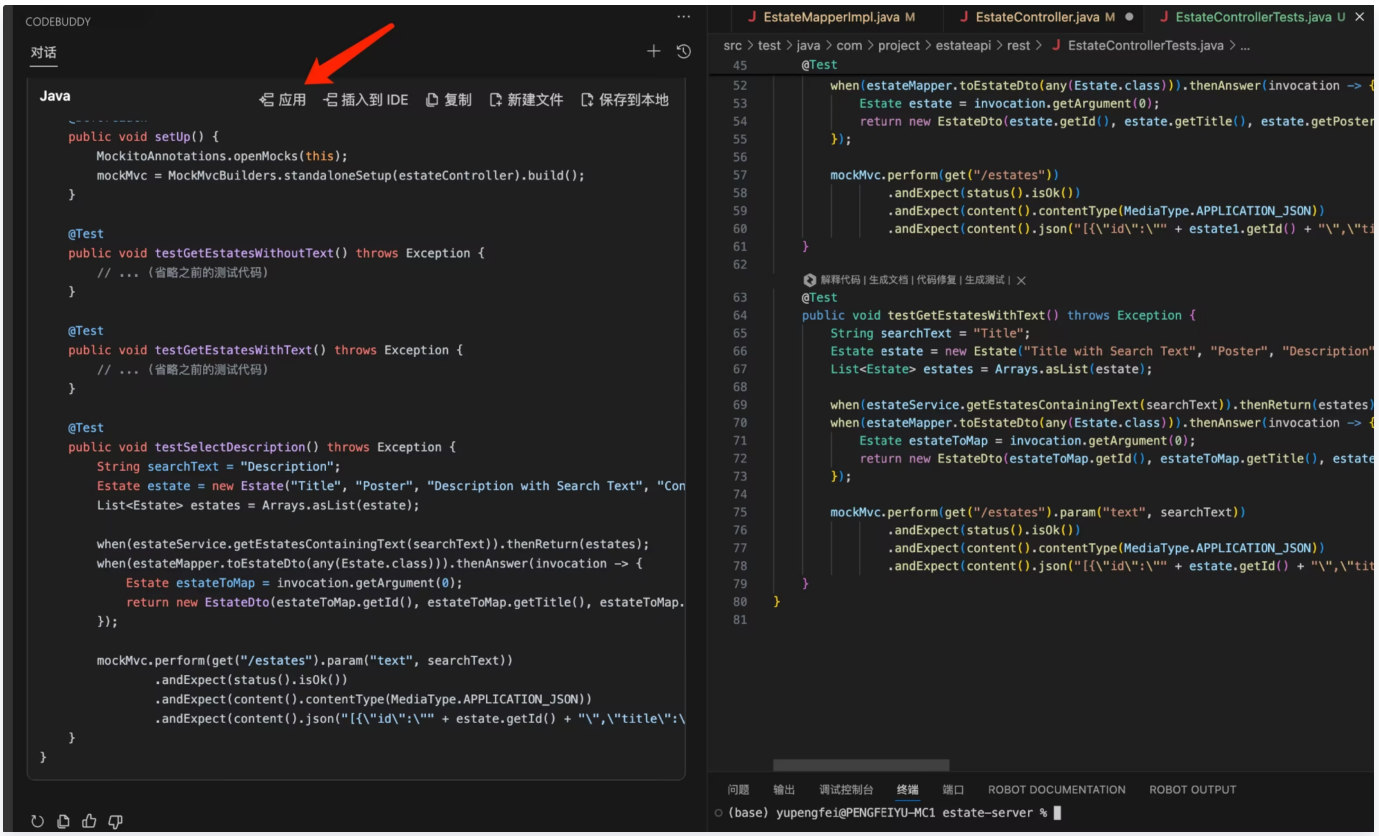


代码助手的对话模型会识别用户意图、结合上下文对话内容优化单元测试结果。如下图所示：

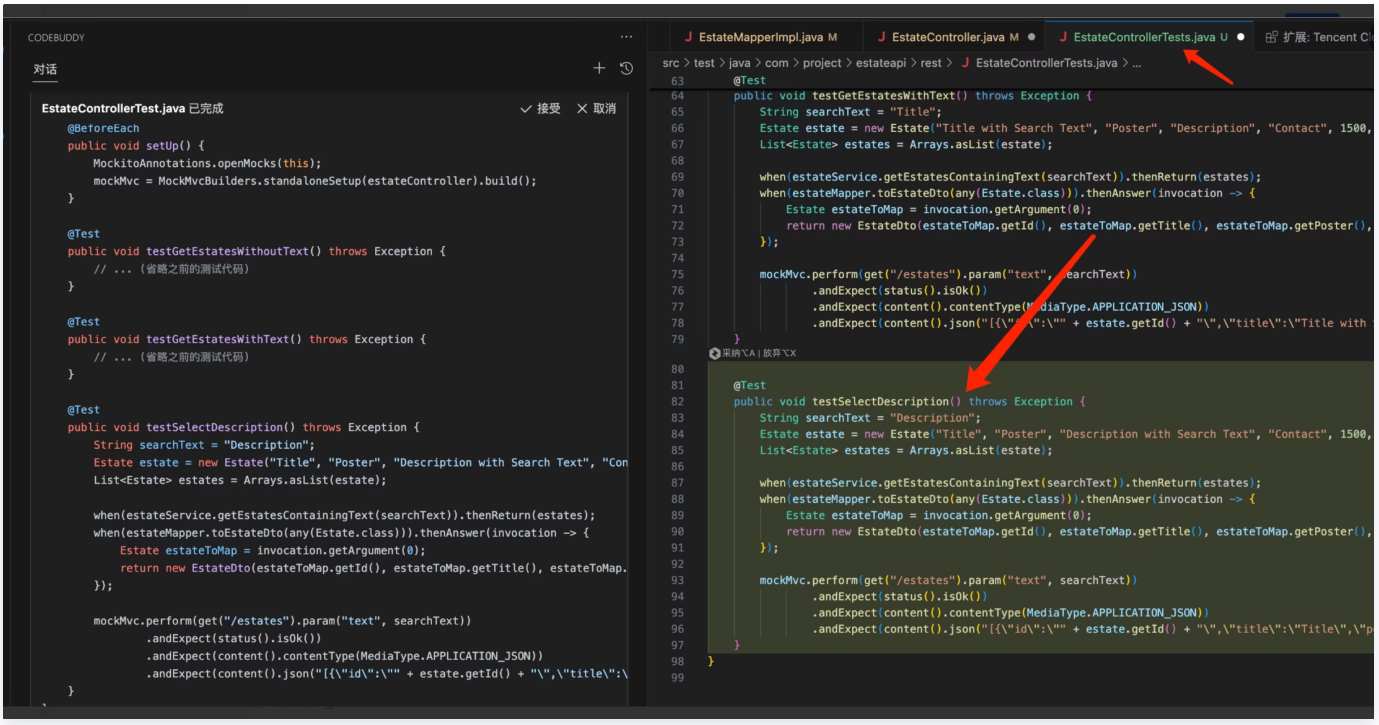


3. 一键应用生成的单元测试内容。

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。如下图所示：

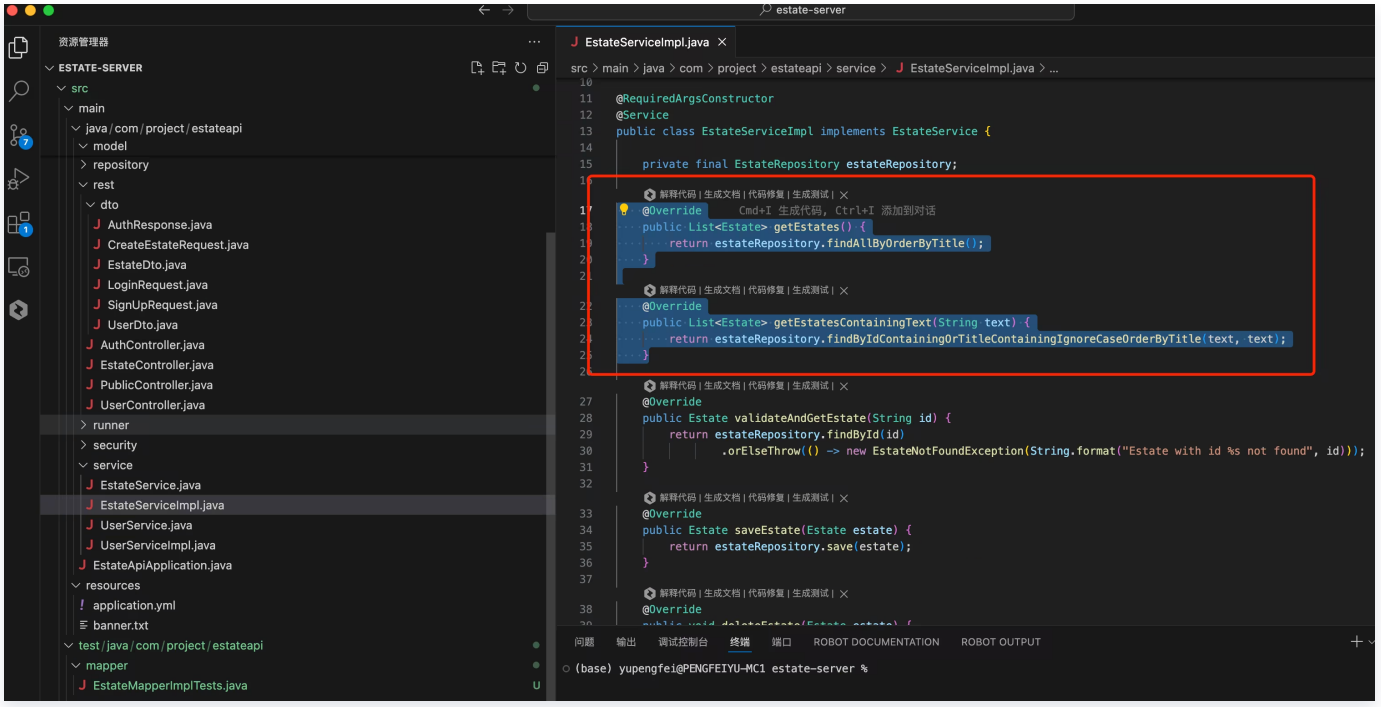


实际编码案例解析

1. 针对业务代码准确生成单元测试。

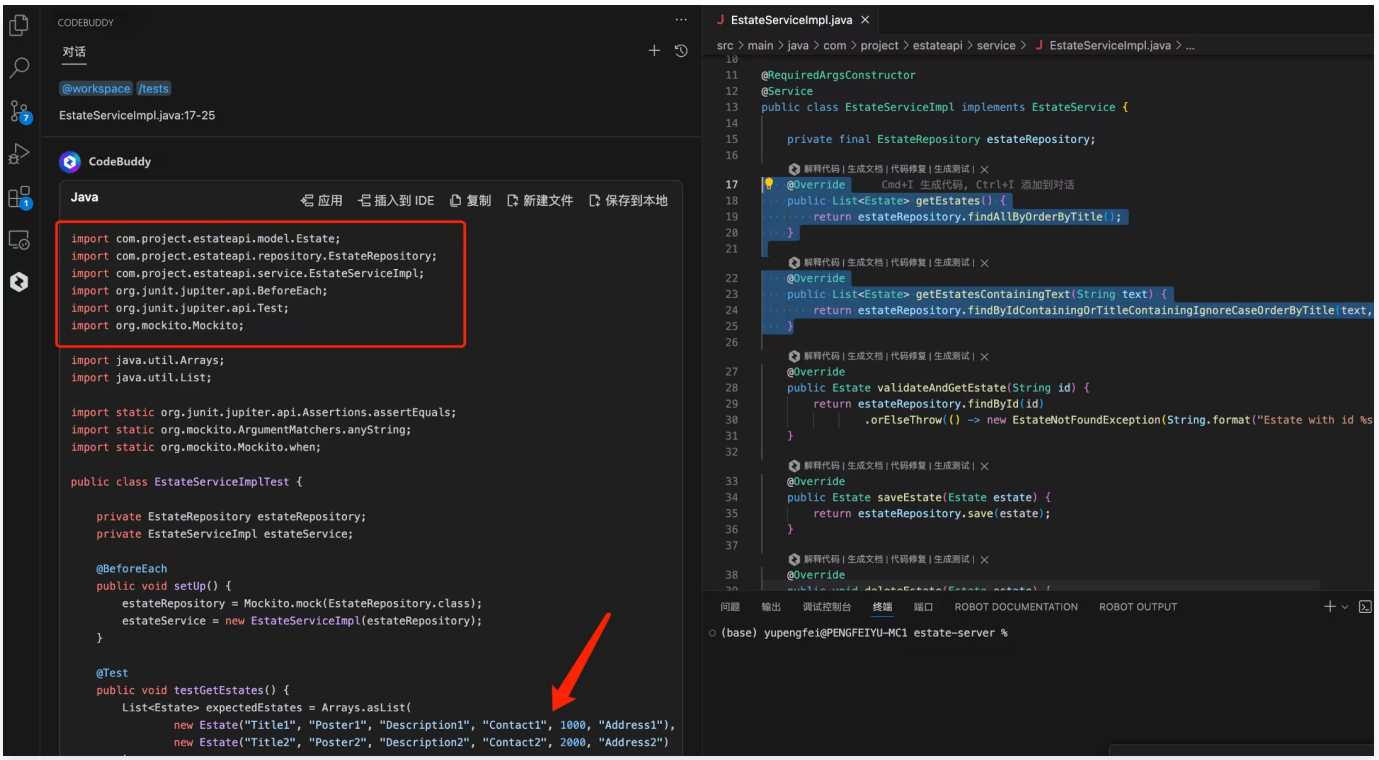
在真实的开发场景中，软件的功能都是异常复杂的，一个完整的软件功能由多个类、方法共同组成。以 Java 开发为例，一个功能的开发，可能由 类的定义、Controller 控制层的方法、Server 接口定义、ServiceImpl 接口实现类方法、Repository 和 DTO 等等部分组成。在生成单元测试的时候，需要考虑到真实业务中的各个组件关联性。

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。

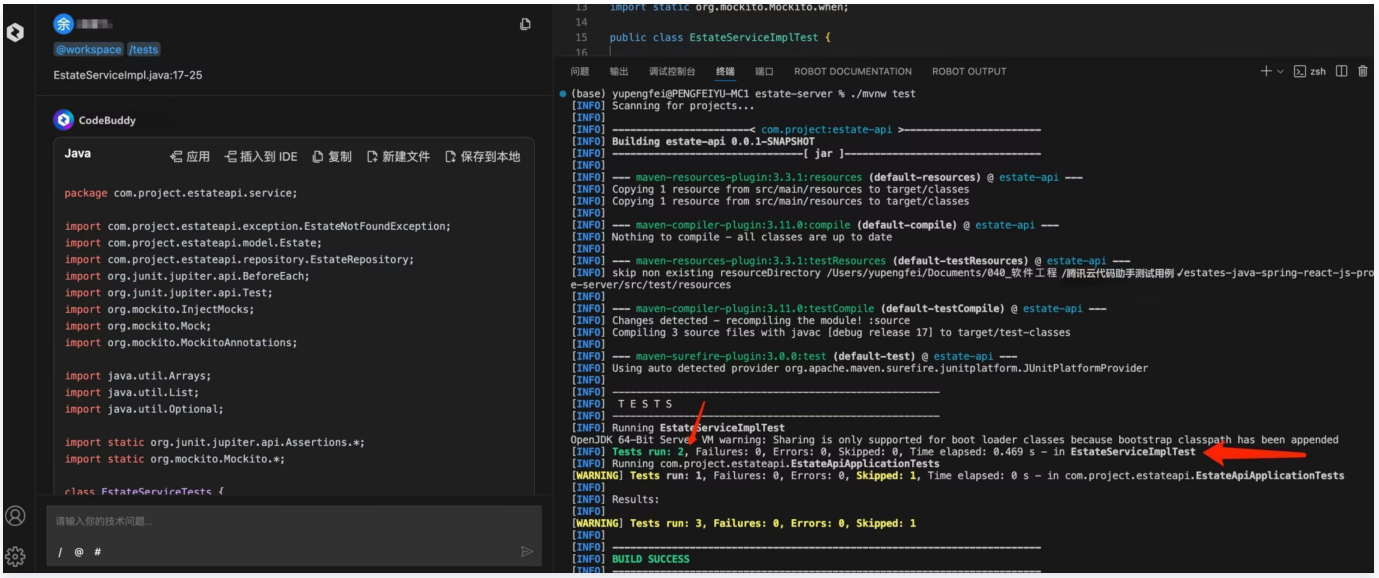


我们选中 getEstates 和 getEstatesContainingText 方法之后，使用 /test 为其生成单元测试代码。在结果中可以看到，腾讯云代码助手在生成单元测试的时候，会分析代码的工程结构，并在生成的时候智能 import 相关依赖；在生成测试代码的时候，会结合其他文件中的类和方法定义生成测试计划，并准确构造测试数据。如下图所示：

版权所有：腾讯云计算（北京）有限责任公司



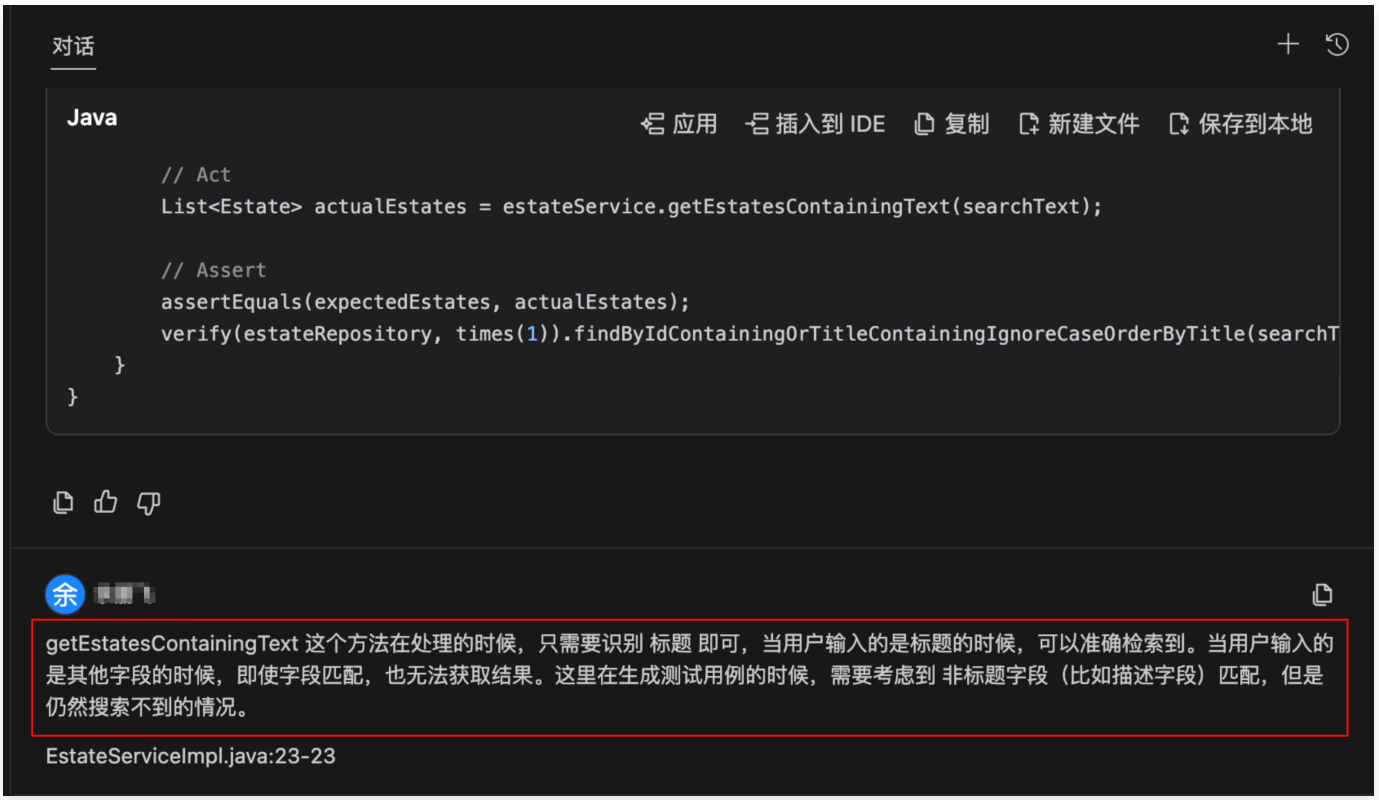
将生成的单元测试内容，插入到指定单元测试文件中，可以直接执行 mvn test 命令进行单元测试验证，如下图所示可以看到这里生成的代码可以编译通过，并完成单元测试功能。



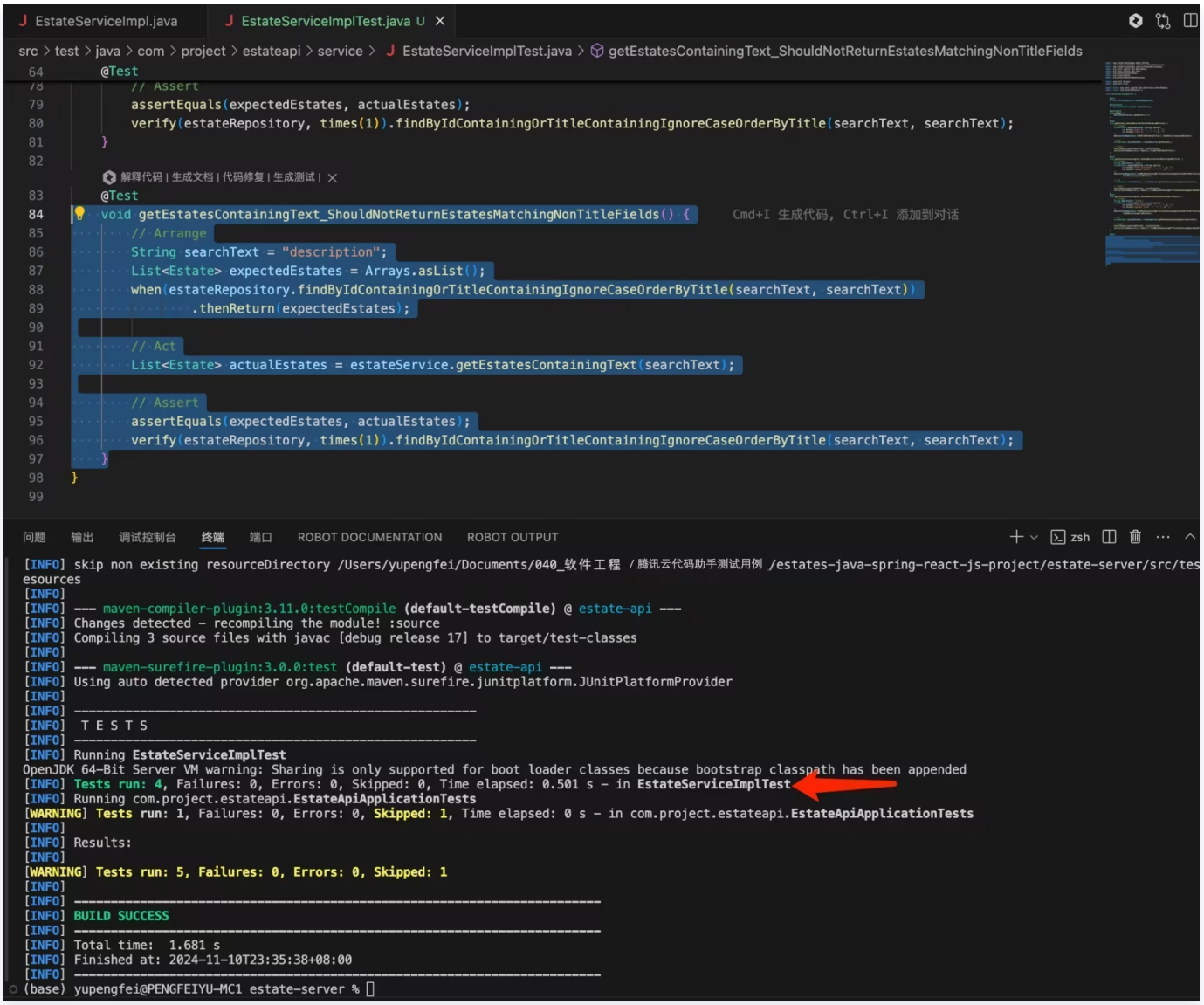
2. 结合业务逻辑逐步完善测试结果。

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：



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



总结

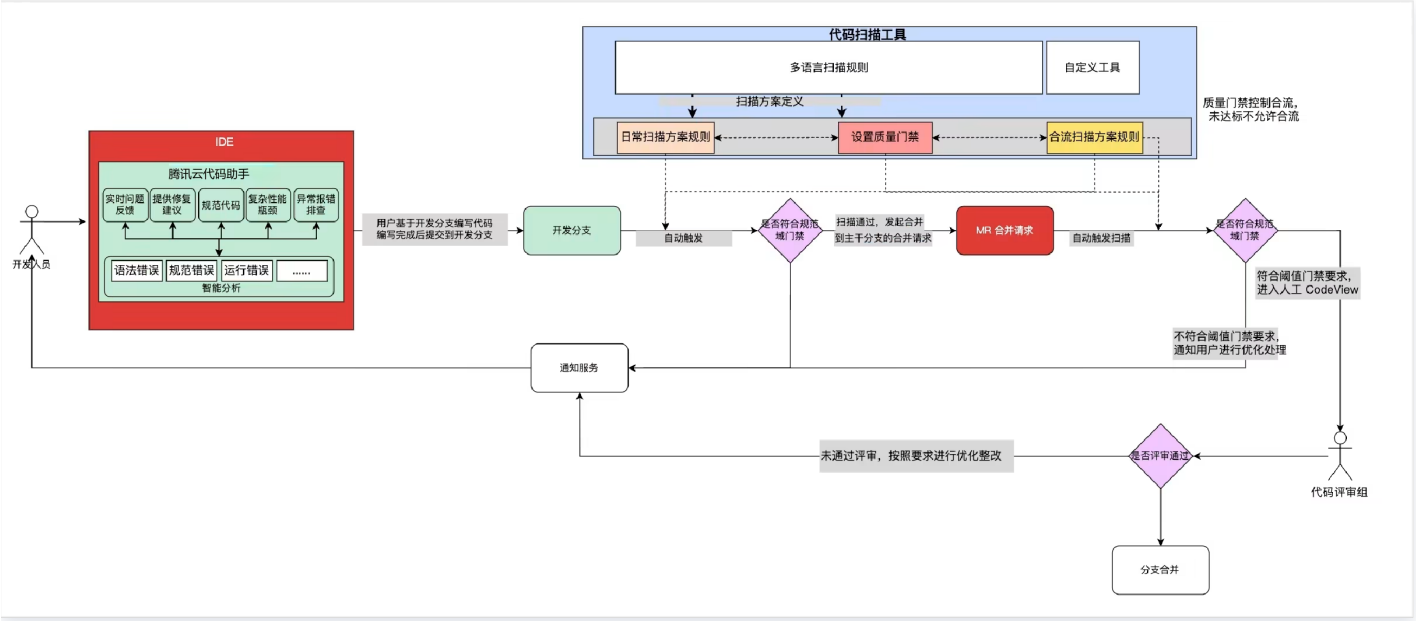
腾讯云代码助手结合腾讯混元大模型的功能，已经在单元测试领域展现出了显著的价值，它不仅提高了开发效率，还帮助确保了软件的质量和稳定性。随着技术的不断进步，在软件工程3.0的时代下，AI 在单元测试中的应用将更加广泛和深入，为软件开发带来更多的便利和创新。

版权所有：腾讯云计算（北京）有限责任公司

代码诊断应用实践

最近更新时间：2026-07-16 21:20:02

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。

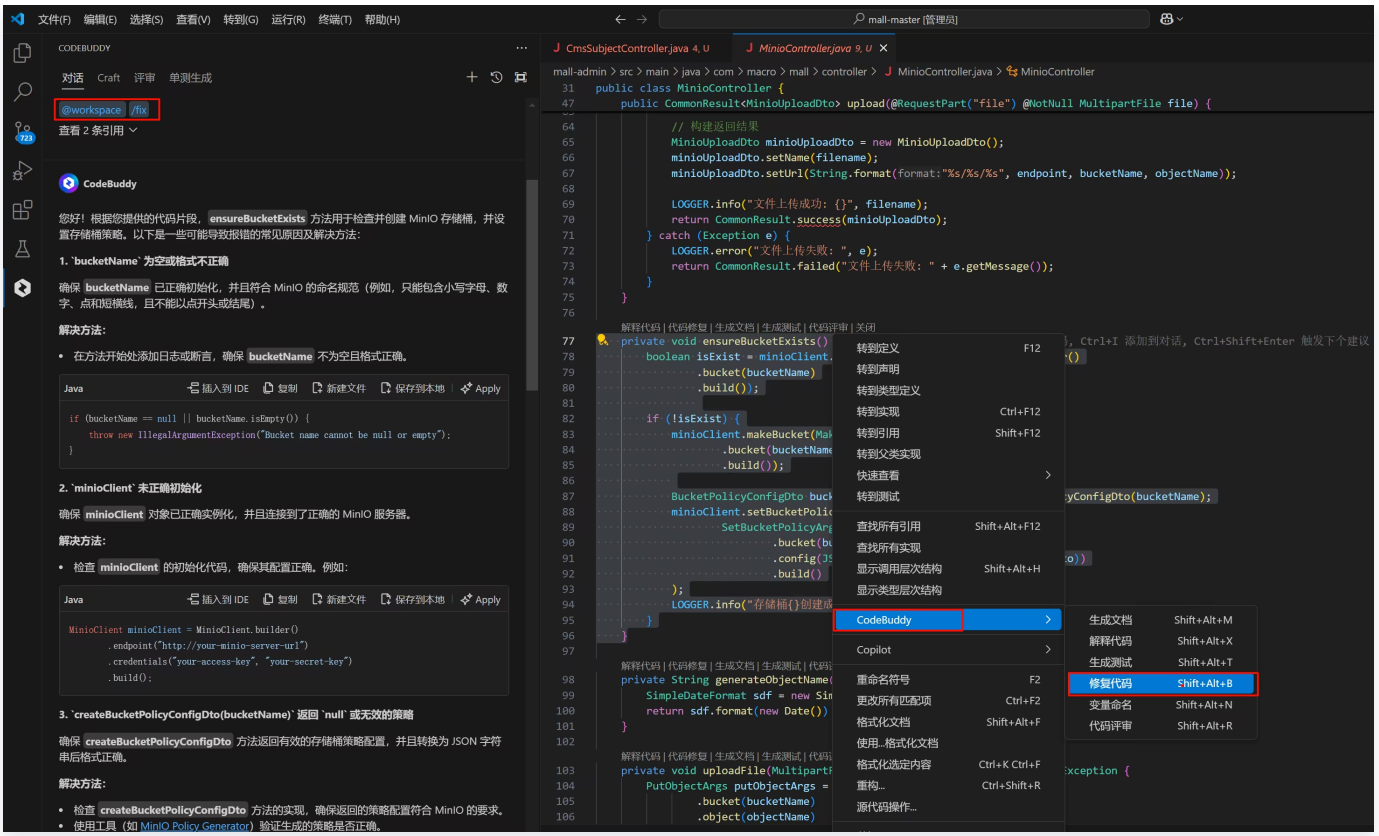


AI 在代码诊断的应用

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。

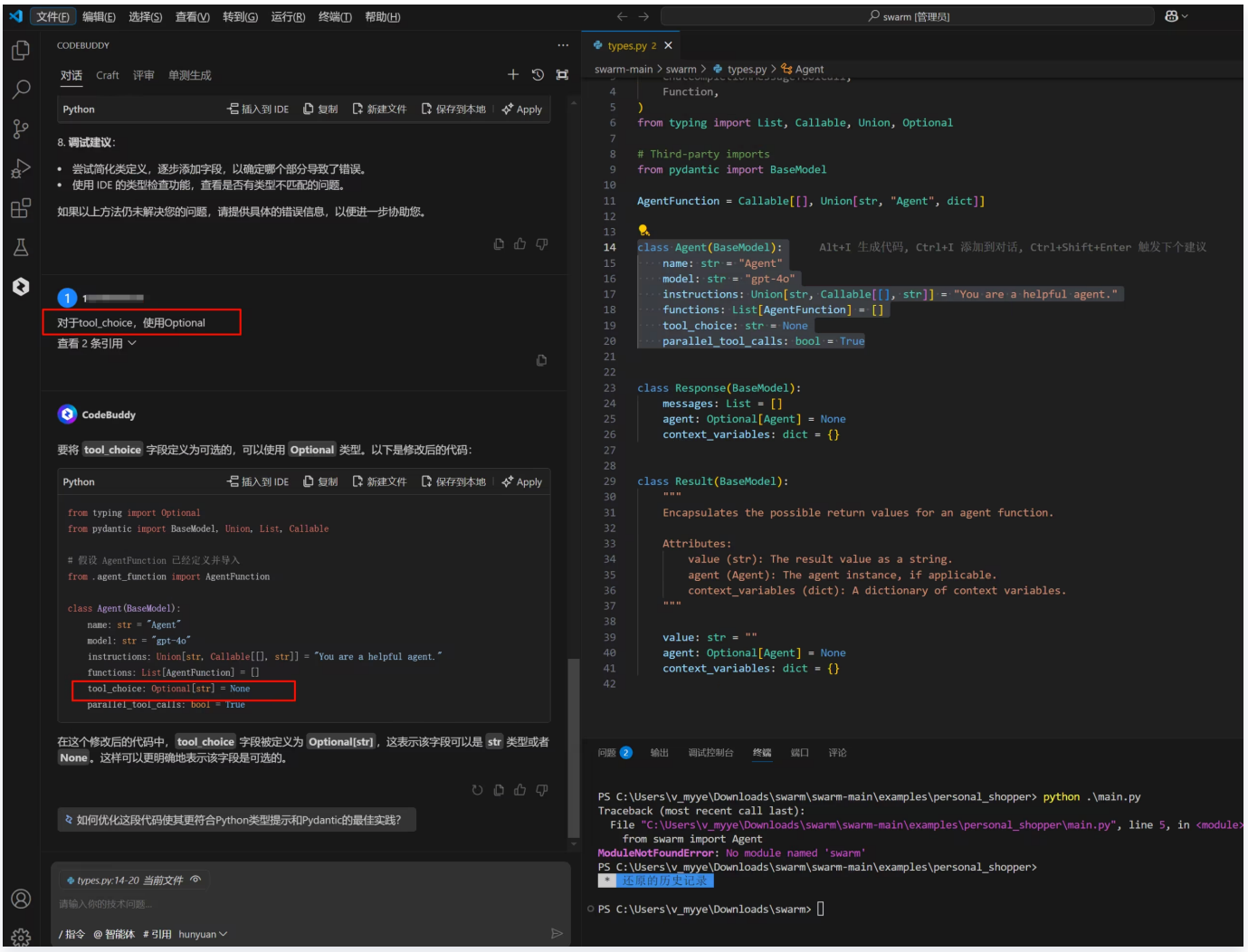
   通过对话框中的 /fix 触发代码诊断，或者使用 IDE 编码区域中使用每个功能方法的快捷键**代码修复**来触发该方法的代码诊断。



2. 多轮对话优化代码诊断。

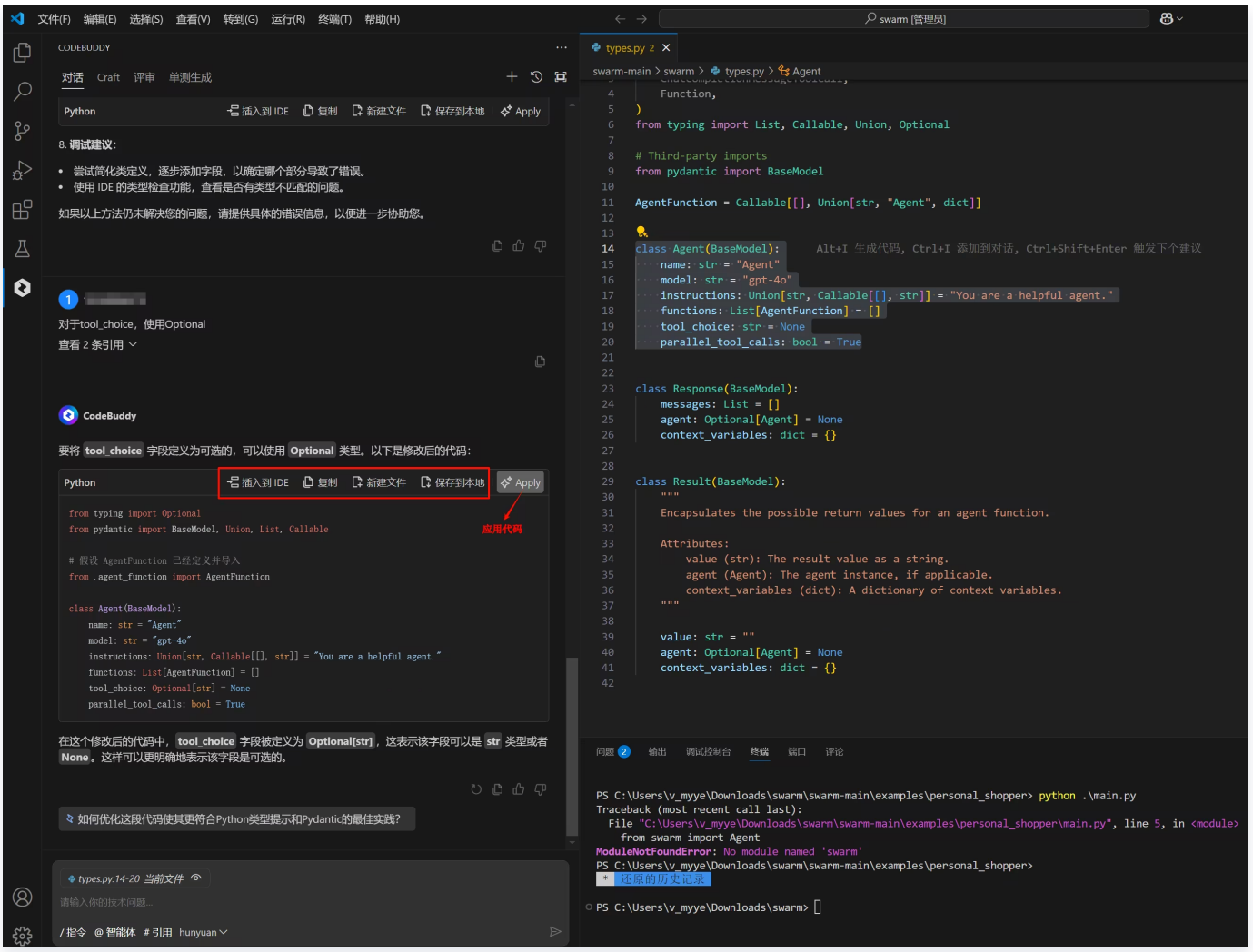
通过多轮对话，告诉 AI 更多的信息，让代码诊断的内容更符合研发人员的预期。例如，可以指定业务边界条件、特殊的异常处理逻辑、数据处理方式等。代码助手的对话模型会识别用户意图结合上下文对话内容优化代码诊断结果。

版权所有：腾讯云计算（北京）有限责任公司

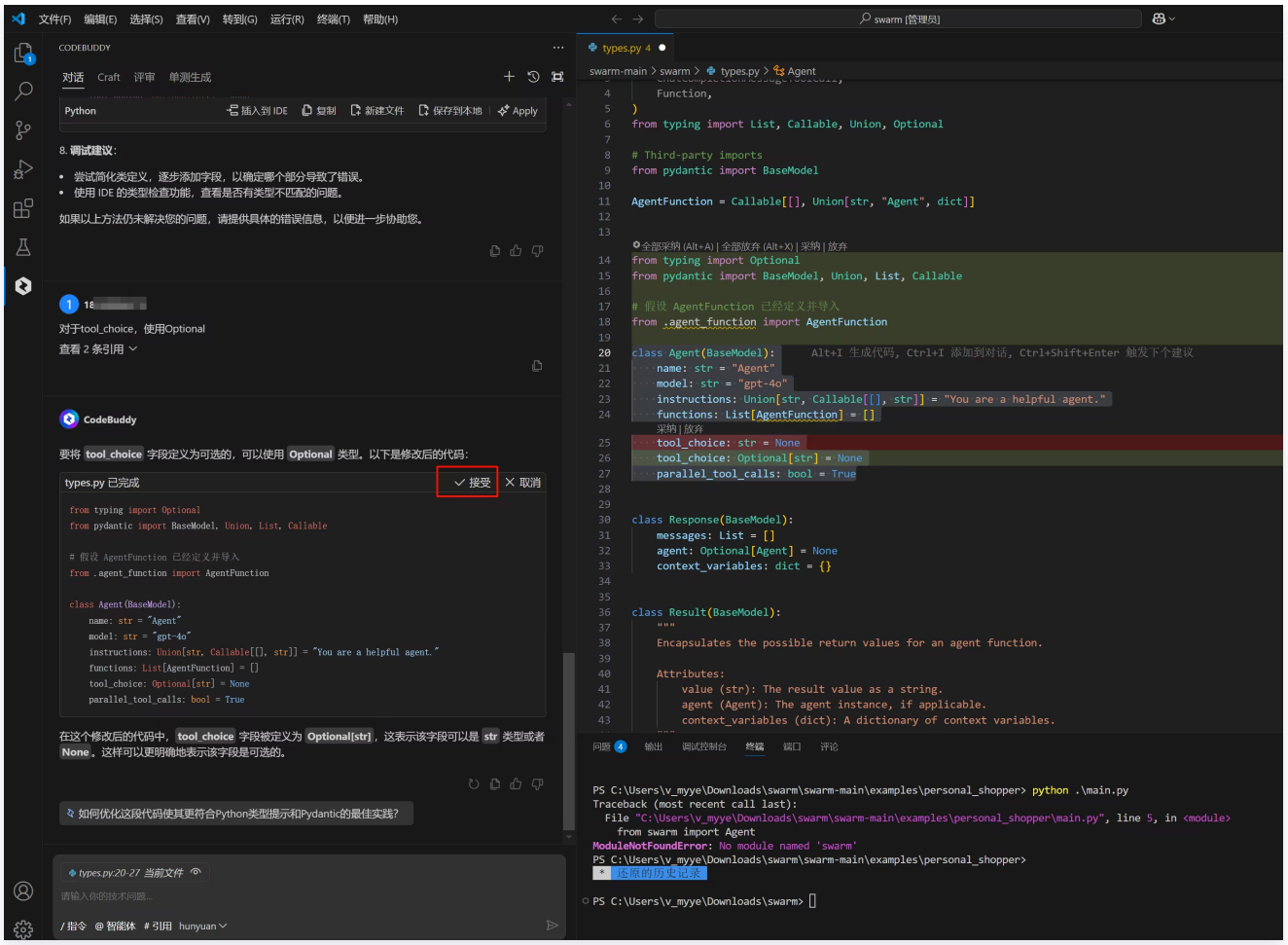


3. 一键应用代码诊断内容。

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。



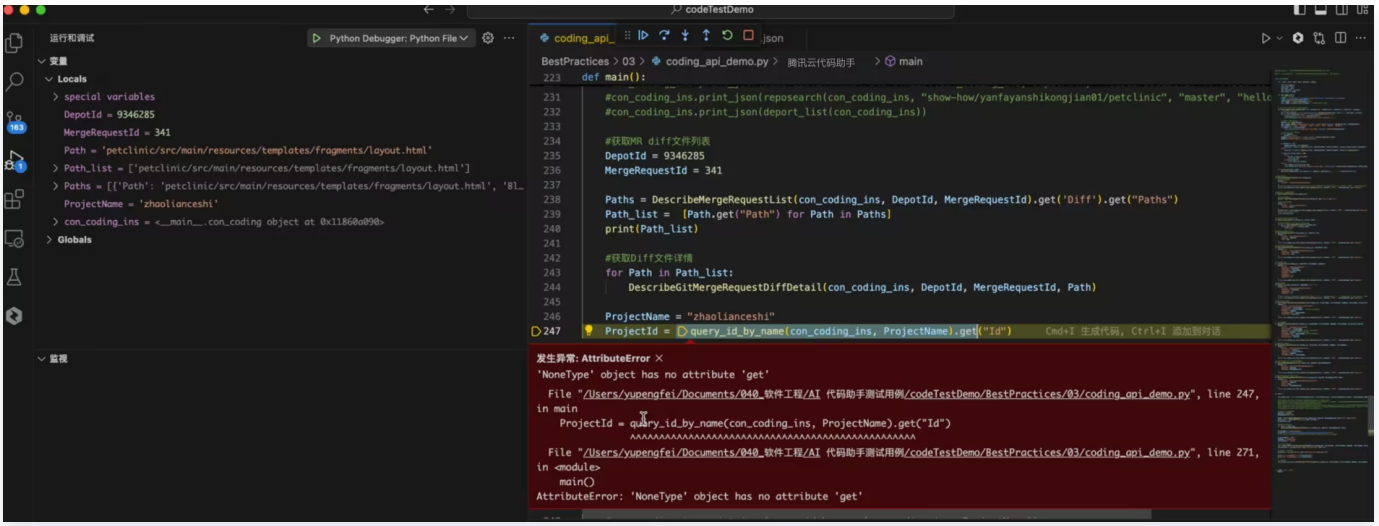
3.2 可以在 IDE 编码区域看到**应用**这部分通过 diff 的能力进行高亮地区分诊断建议代码和当前文件代码的对比修改情况，让研发人员快速识别到改动并判断是否接受。

实际场景演示

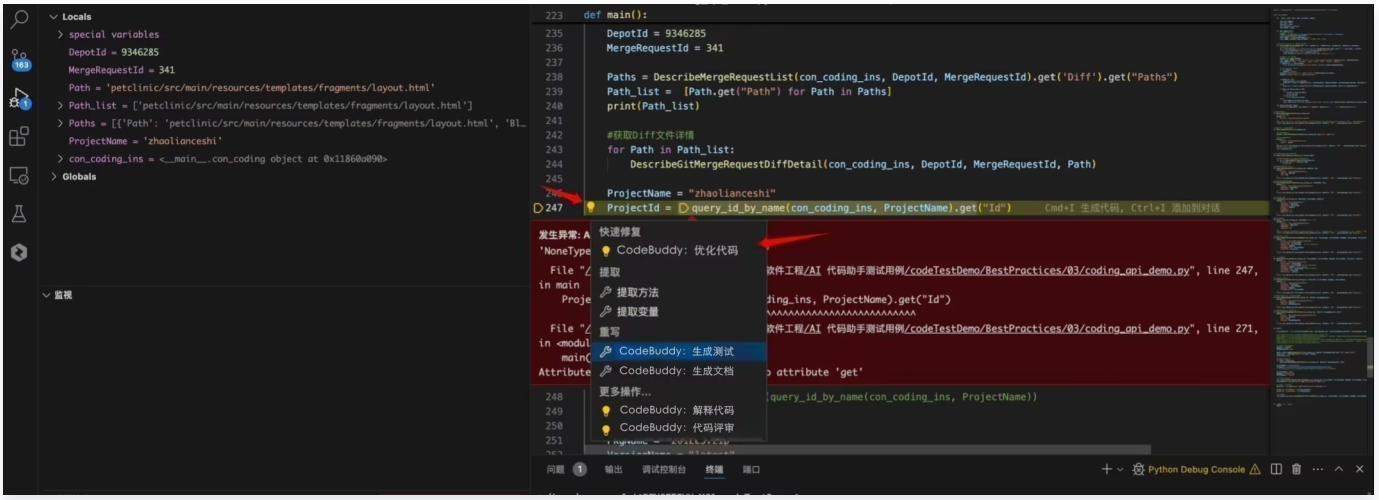
Debug 调试时诊断修复

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。

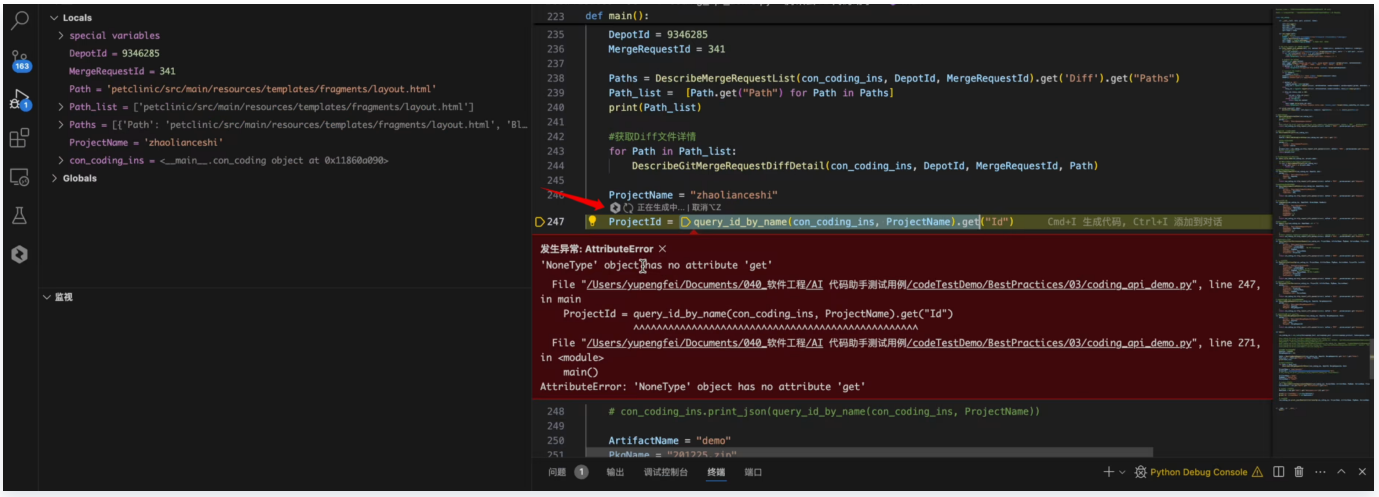
1. 启动 Debug 时出现异常：



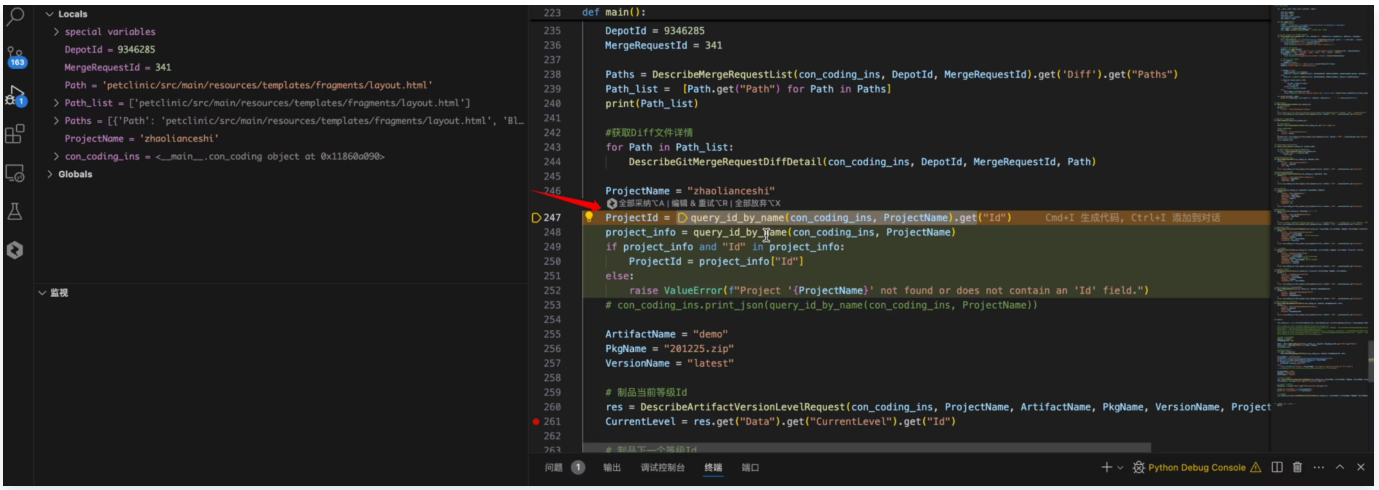
2. 通过 CodeBuddy：优化代码功能获取解决方案：



3. 正在获取解决方案：



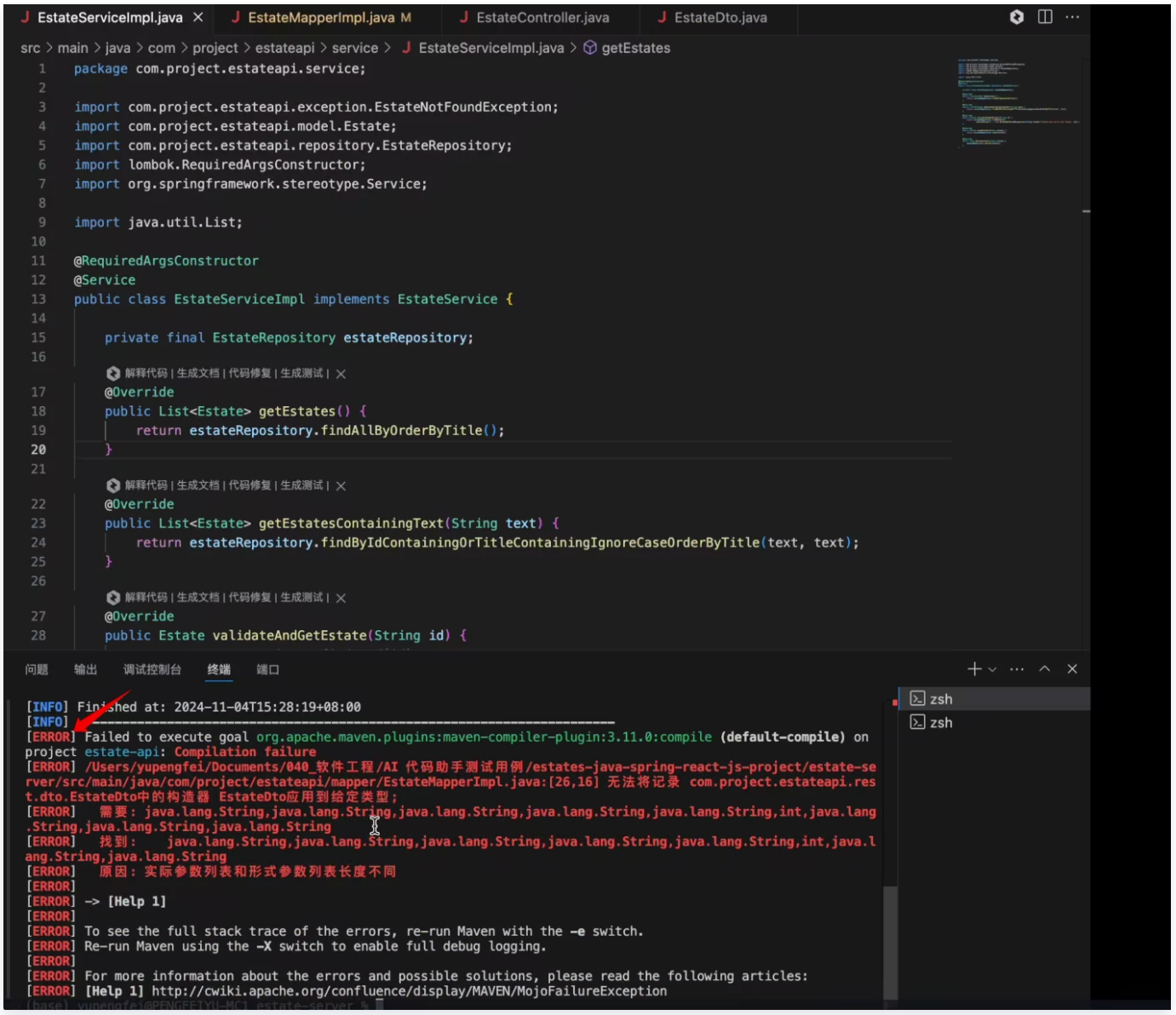
4. 可以选择采纳对代码进行修复。



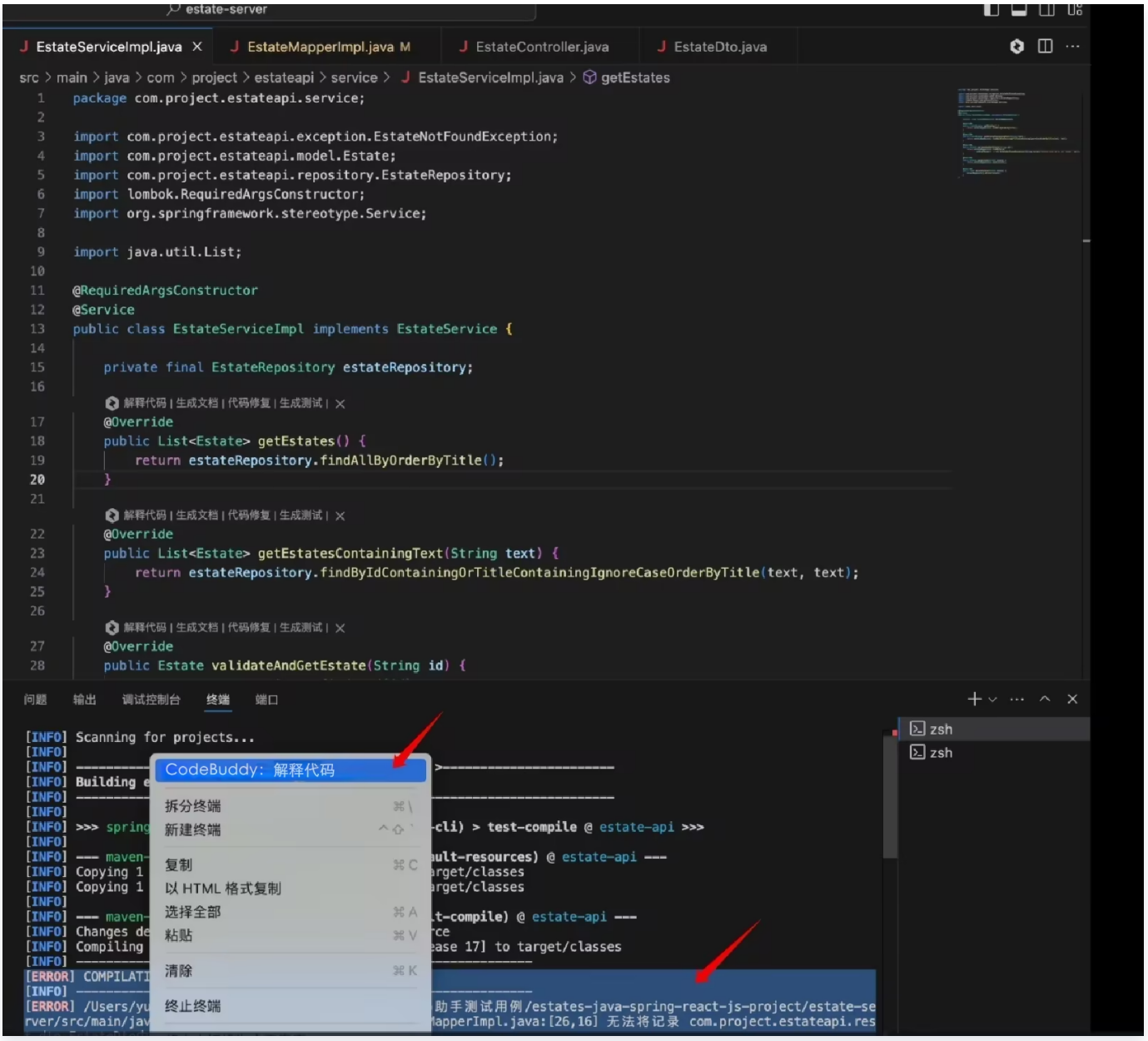
程序运行错误时定位及修复

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。

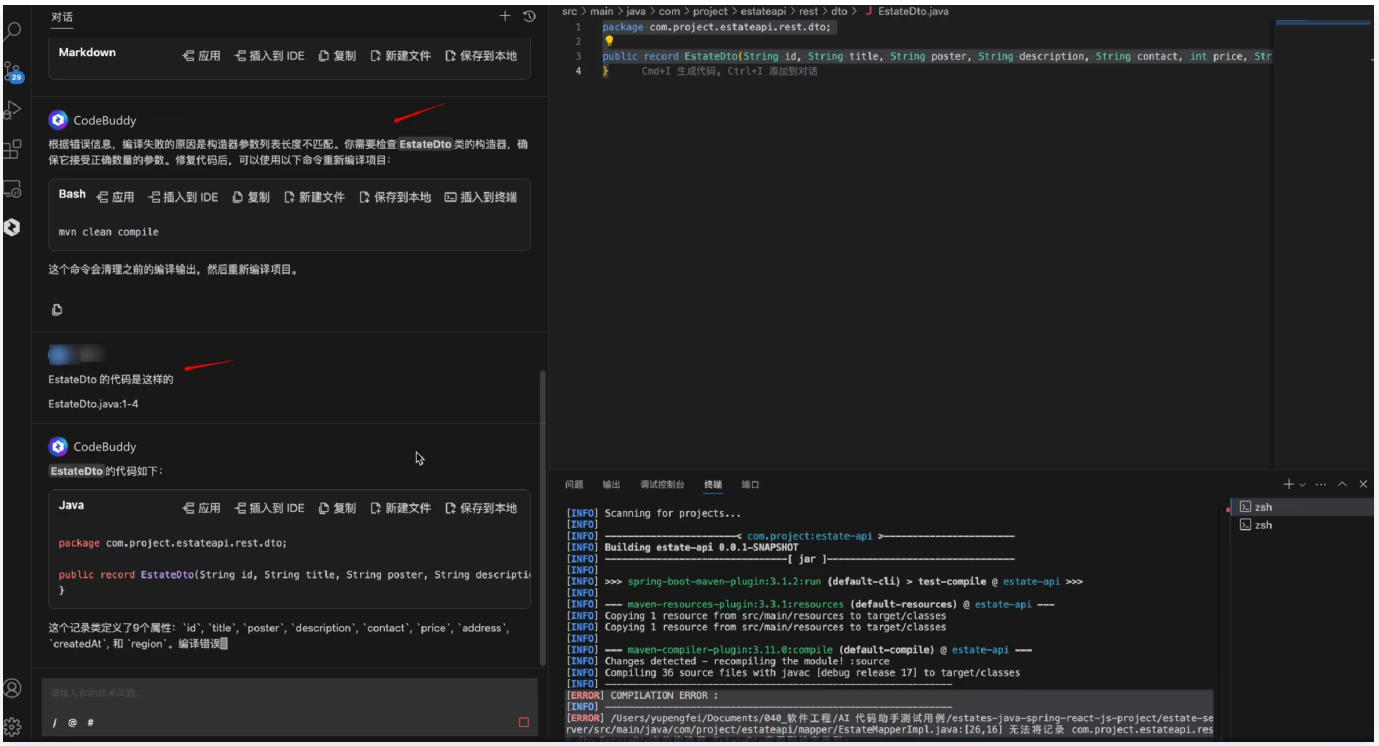
1. 运行错误报错：



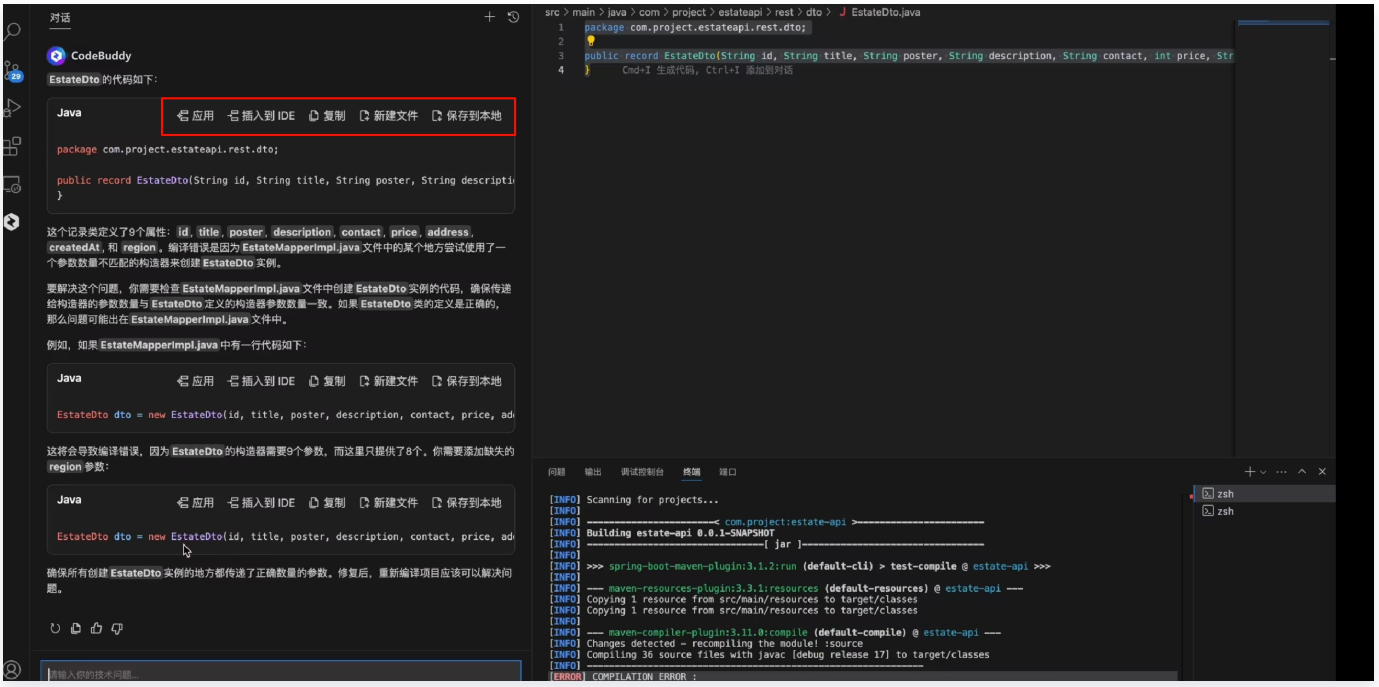
2. 选中异常报错信息通过 **CodeBuddy：解释代码**功能进行问题分析：



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。

版权所有：腾讯云计算（北京）有限责任公司

企业管理实践
构建企业专属知识库

最近更新时间：2026-07-16 21:20:02

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。

版权所有：腾讯云计算（北京）有限责任公司

- 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。

版权所有：腾讯云计算（北京）有限责任公司

- 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。

版权所有：腾讯云计算（北京）有限责任公司

        - **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的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。

版权所有：腾讯云计算（北京）有限责任公司

- 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。

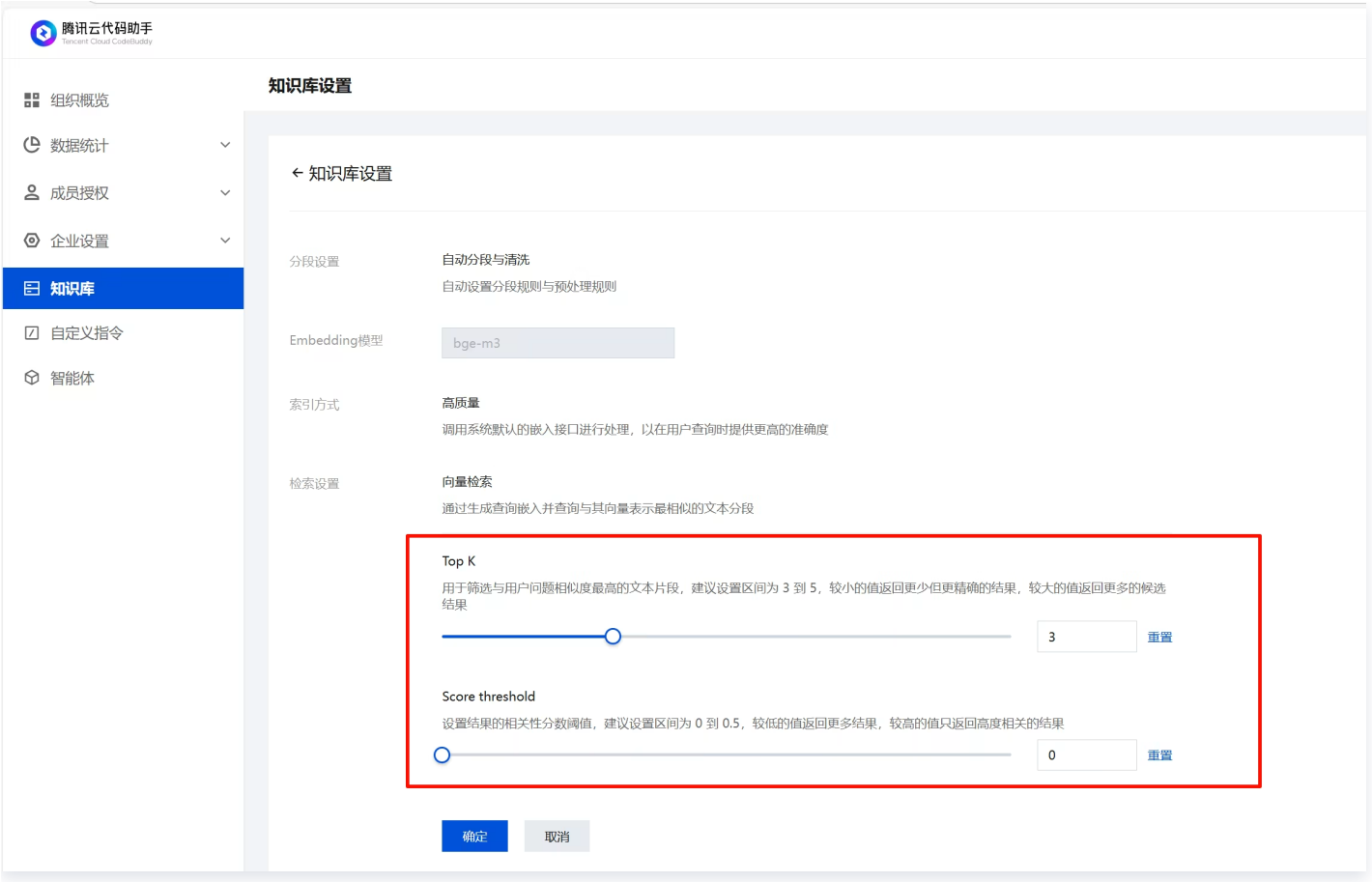
腾讯云代码助手支持将文档与代码文件进行混合压缩后上传，支持常见压缩包格式（.zip、.tar、.tgz、.tar.gz、.gz、.gzip 格式，**每个压缩包不超过300MB，压缩包内文件数量不限**）。



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 到 0.5，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 。

帮助与反馈

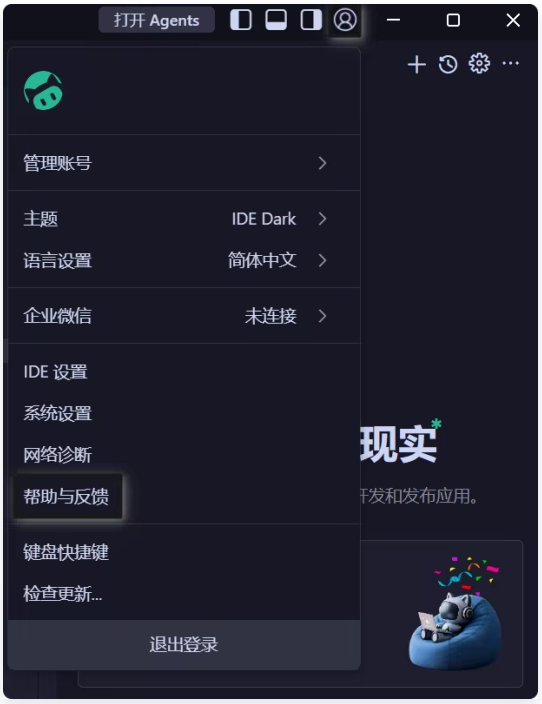
最近更新时间：2026-07-17 17:54:31

如果您希望获得更好的产品体验，可以在 CodeBuddy 内直接对话产品团队，提出宝贵的意见和建议。

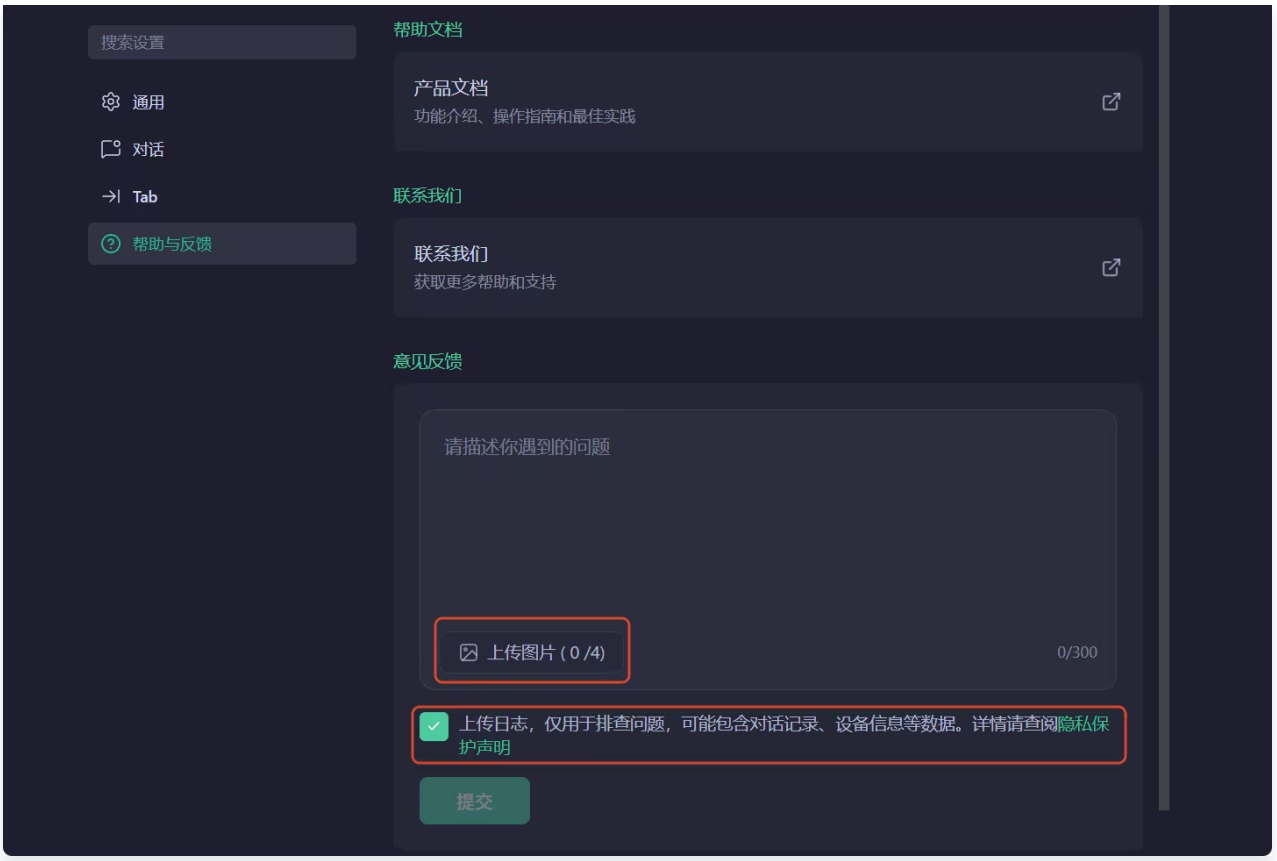
需求反馈

如果您希望 CodeBuddy 提供更好的功能来满足您的需求，请在 CodeBuddy IDE 内反馈给团队。

1. 在 IDE 内单击右上角的头像图标，选择**帮助与反馈**：



2. 在**意见反馈**处描述您的需求，建议上传图片并勾选上传日志，以便团队更准确地理解您的需求和问题：


搜索设置
通用
对话
Tab
帮助与反馈
帮助文档
产品文档
功能介绍、操作指南和最佳实践
联系我们
联系我们
获取更多帮助和支持
意见反馈
请描述你遇到的问题
上传图片（0 /4）
0/300
上传日志，仅用于排查问题，可能包含对话记录、设备信息等数据。详情请查阅隐私保护声明
提交