

# 移动云代码大模型

代码补全

生成代码

解释代码

代码注释

单元测试

智能问答

代码调优

代码翻译

图生代码

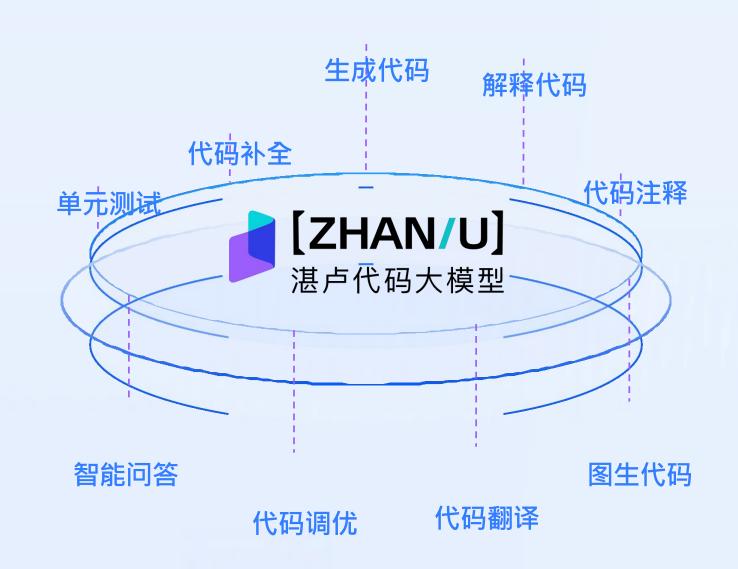
FIFER



## 产品简介

移动云代码大模型,融合海量代码 库、技术文档、研发知识,为研发 人员提供智能、高效的编程支持。

移动云代码大模型能够**准确理解开 发者的意图**,将产品需求、原型图 转化为高效、精确的代码,从而**大 幅提升编码效率,优化代码质量**, 助力企业实现研发管理的数字化转型。



## 语言能力

兼容 Visual Studio Code、JetBrains IDEs 等主流 IDE;

支持 Java、Python、Go、C/C++/C#、JavaScript、TypeScript、PHP、 Ruby、Rust、Scala 等主流编程语言。

## 产品优势

【多语言支持】能够理解和使用100种编程语言,满足多样化的开发需求;

【高效精确】通过智能技术极大提升编码速度和准确性。

【多模态支持】能够解析图片内容,智能生成相应代码。

【持续学习】不断更新和学习新的编程知识和技术,保持行业领先。

# 一天

#### 一、插件安装与登录

- 1.1 VS Code中的湛卢插件安装
- 1.2 JetBrains中的湛卢插件安装

#### 二、使用前配置

- 2.1 VS Code中的插件配置
- 2.2 JetBrains中的插件配置

#### 三、操作指南

#### 3.1 代码补全模式

- 3.1.1 行补全
- 3.1.2 块补全
- 3.1.3 注释生成补全

#### 3.2 Chat模式

- 3.2.1 生成代码(支持多伦对话)
- 3.2.2 解释代码
- 3.2.3 生成代码注释
- 3.2.4 生成单元测试
- 3.2.5 指定代码库智能问答
- 3.2.6 代码调优纠错
- 3.2.7 代码翻译
- 3.2.8 图生代码
- 3.2.9 针对报错提示一键生成修复建议
- 3.2.10 互联网检索问答
- 3.2.11 代码评审
- 3.2.12 Vue升级

#### 3.3 AI程序员

- 3.3.1 项目级单元测试
- 3.4 常见问题解答

## 一、插件安装与登录

## 1.1 VS Code中的湛卢插件安装

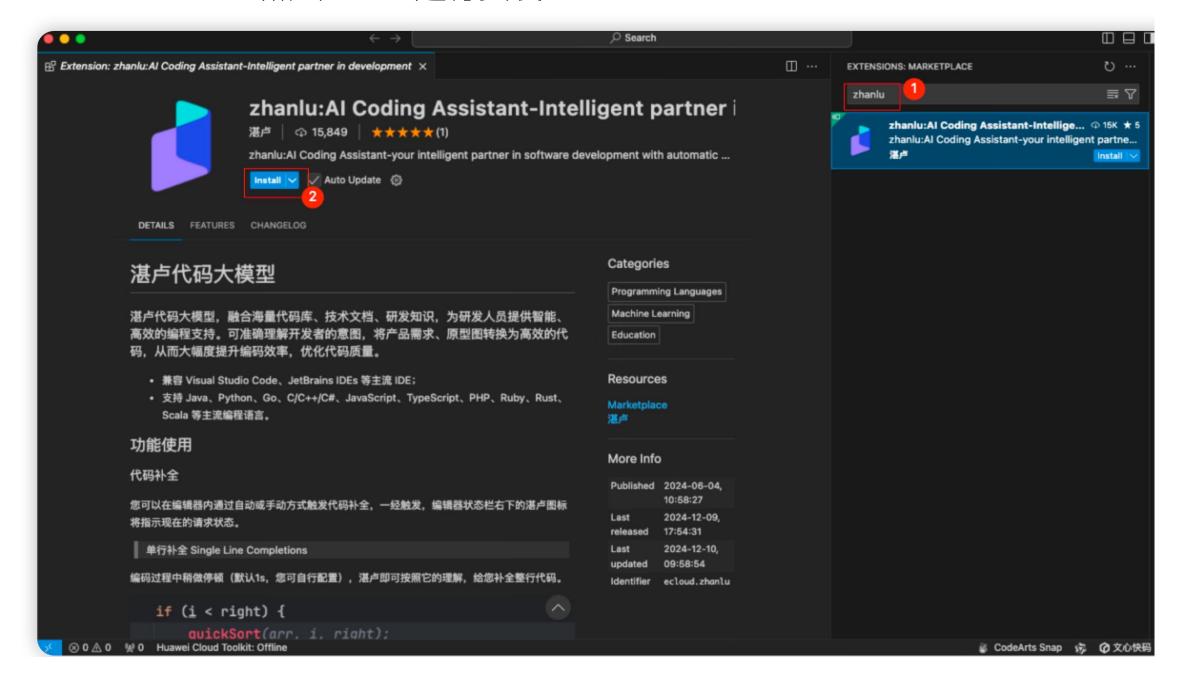
#### 1.1.1 运行环境

适配 VS Code 集成开发环境(**1.86及以上版本**),支持**Windows、macOS**等主流操作系统。

#### 1.1.2 安装步骤

#### ▶ 步骤一:

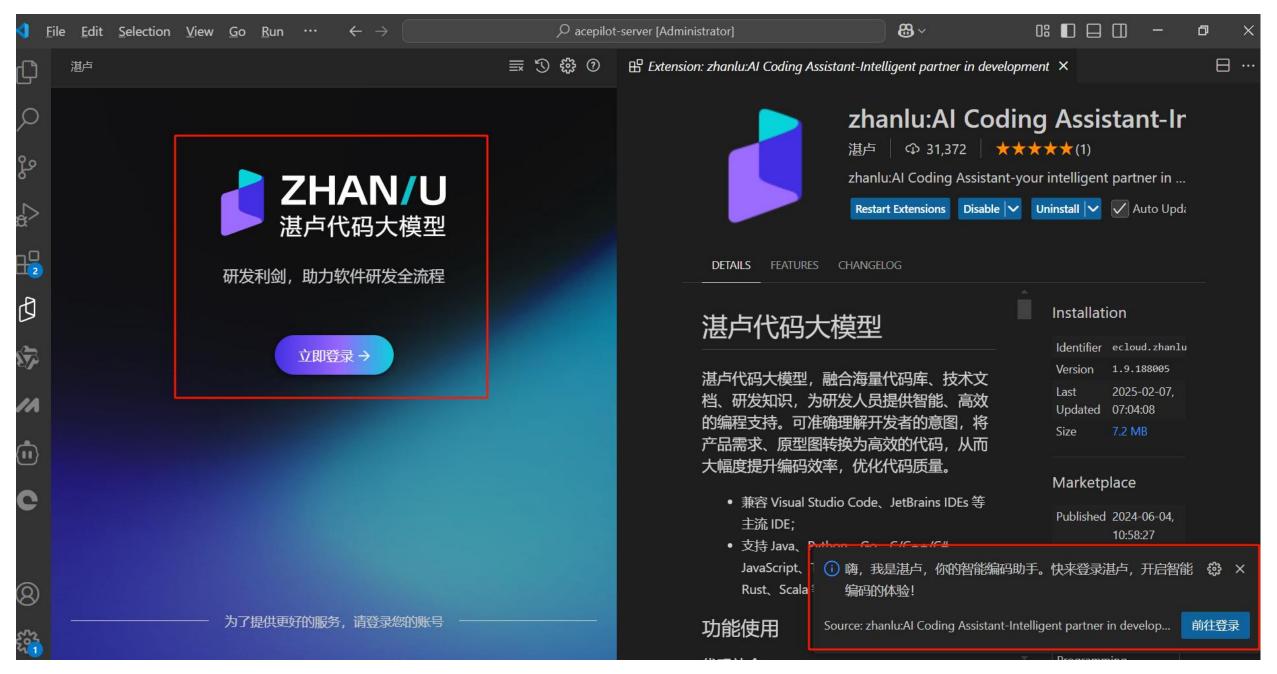
打开VS Code点击左侧扩展(Extensions)按钮,进入插件市场,搜索 "zhanlu",点击install进行安装。



#### ▶ 步骤二:

安装成功后,打开湛卢页面,湛卢提供两种登录方式,可通过**手机验证码登** 录或通过填写AccessKey,SecretKey,License信息登录。未注册过的用户 可通过手机验证码自动完成注册并登录。

若用户未登录, IDE会弹出提示用户登录的弹窗。

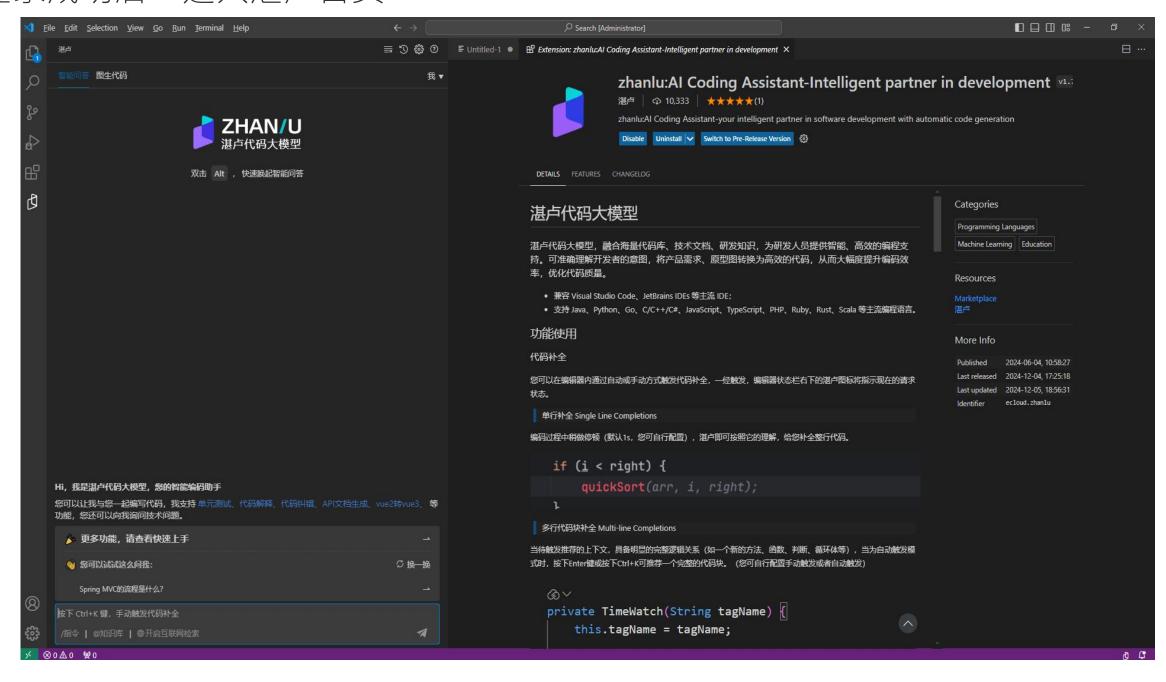


点击前往登录,自动跳转至插件的登录页。



### ▶ 步骤三:

#### 登录成功后,进入湛卢首页。



## 1.2 JetBrains中的湛卢插件安装

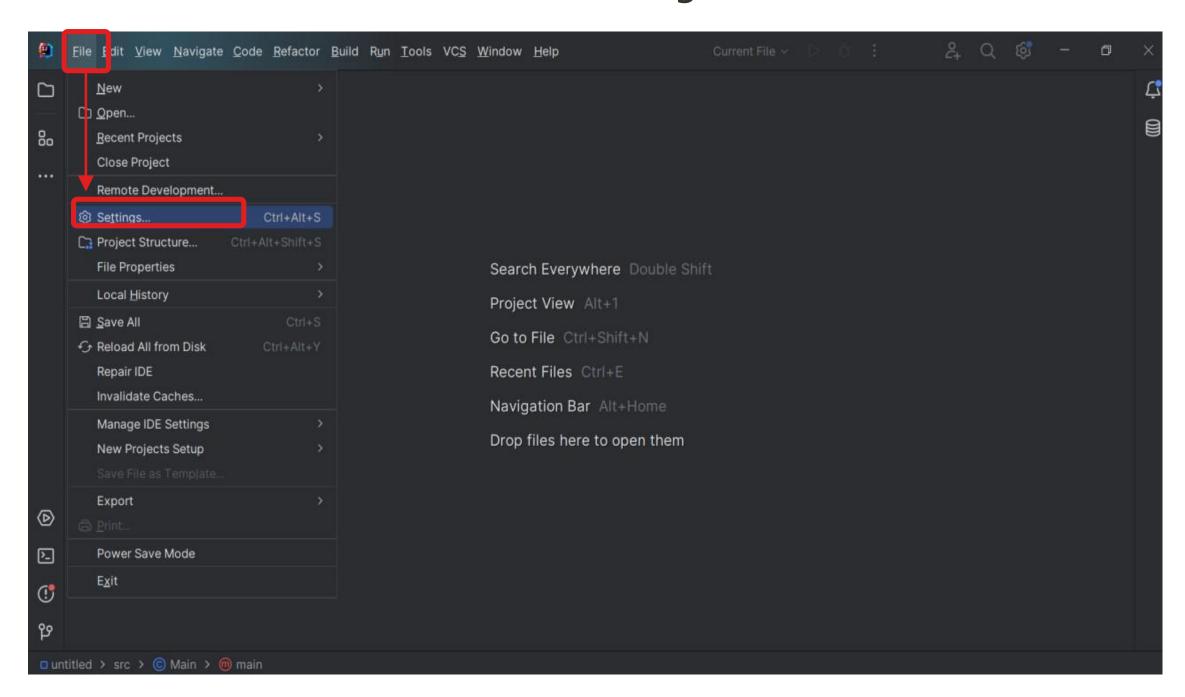
#### 1.2.1 运行环境

适配 **JetBrains** 集成开发环境(支持Ultimate Edition 2020.3-2024.3版本、Community Edition 2020.3-2024.3版本),支持**Windows、macOS**等主流操作系统。

#### 1.2.2 安装步骤

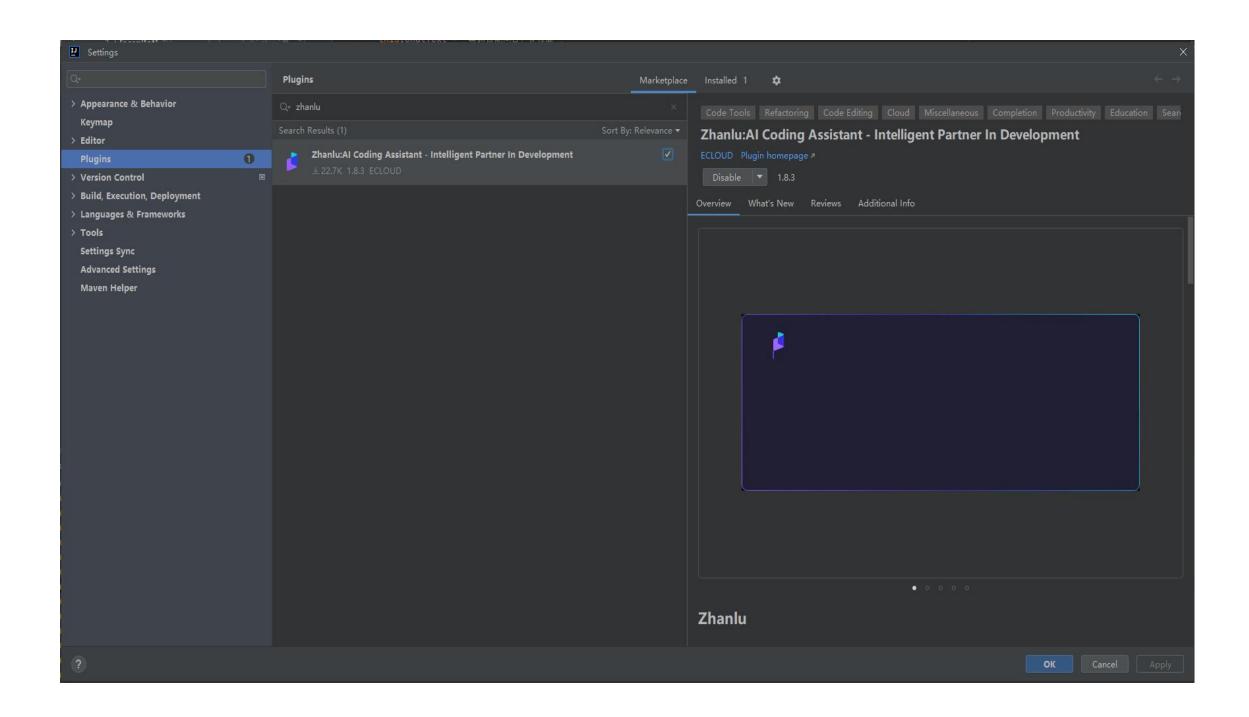
#### ▶ 步骤一:

打开JetBrains,点击顶部"File"-->"Setting"按钮。



#### ▶步骤二:

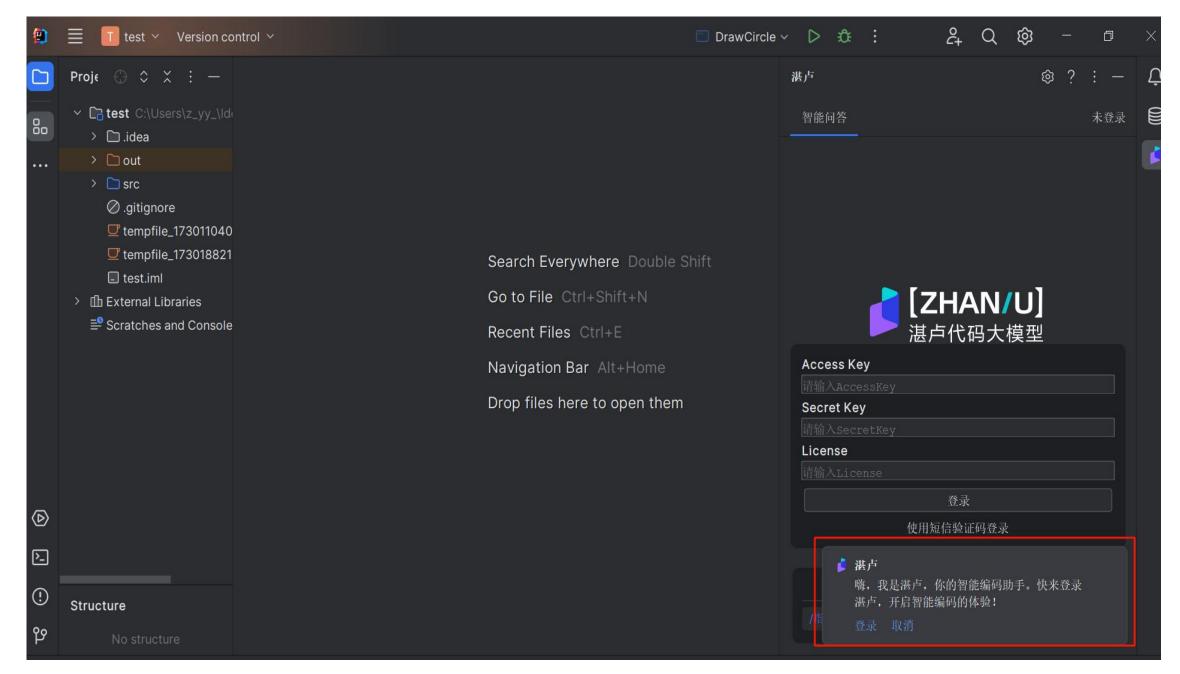
点击**"Plugins"**,在 "Marketplace" 中**搜索"zhanlu"**,点击**"Install"**。



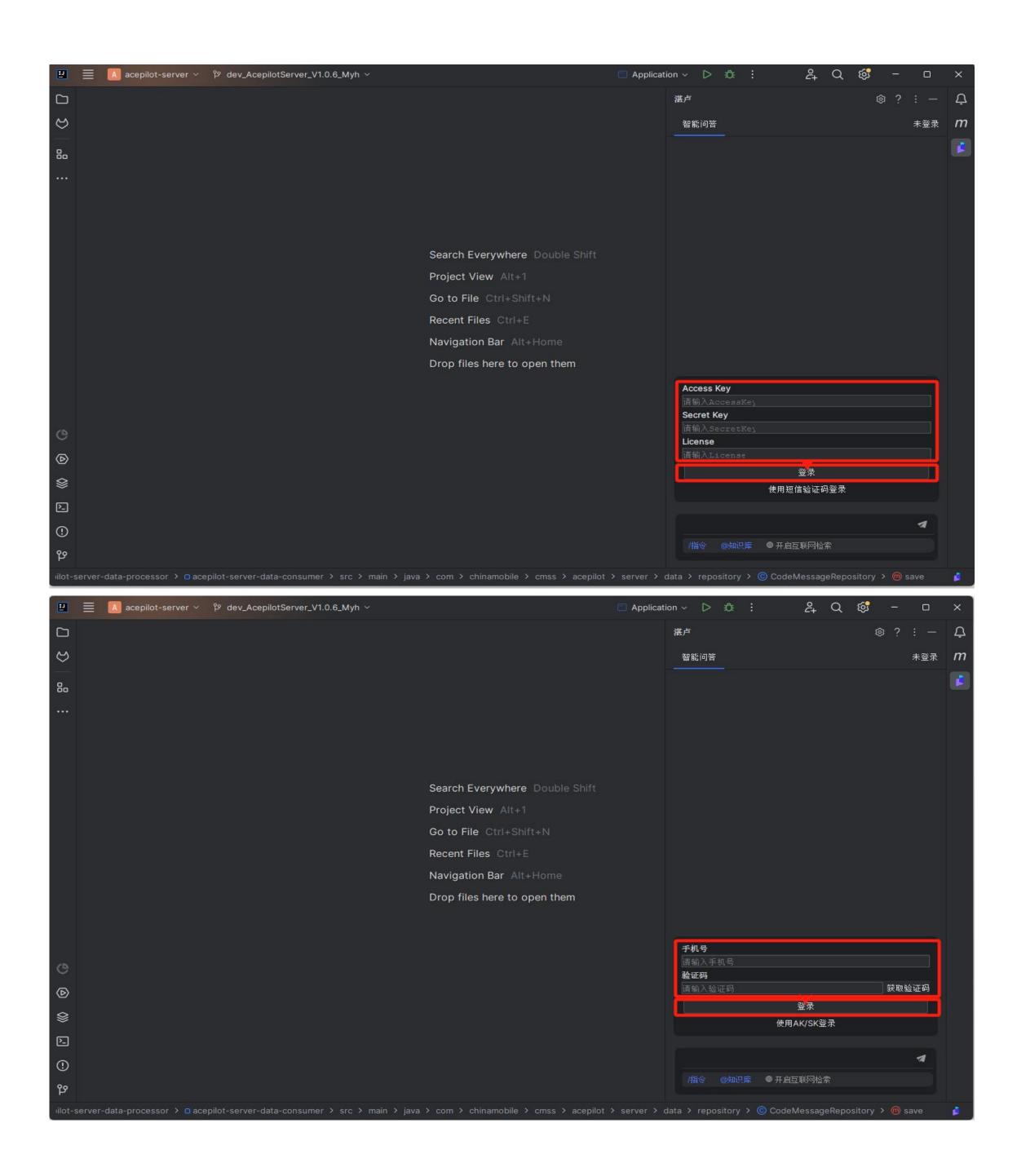
#### ▶步骤三:

安装成功后,打开湛卢页面,湛卢提供两种登录方式,可通过**手机验证码登录**或通过填写AccessKey,SecretKey,License信息登录。未注册过的用户可通过手机验证码登录并自动完成注册。

若用户未登录, IDE会弹出提示用户登录的弹窗。

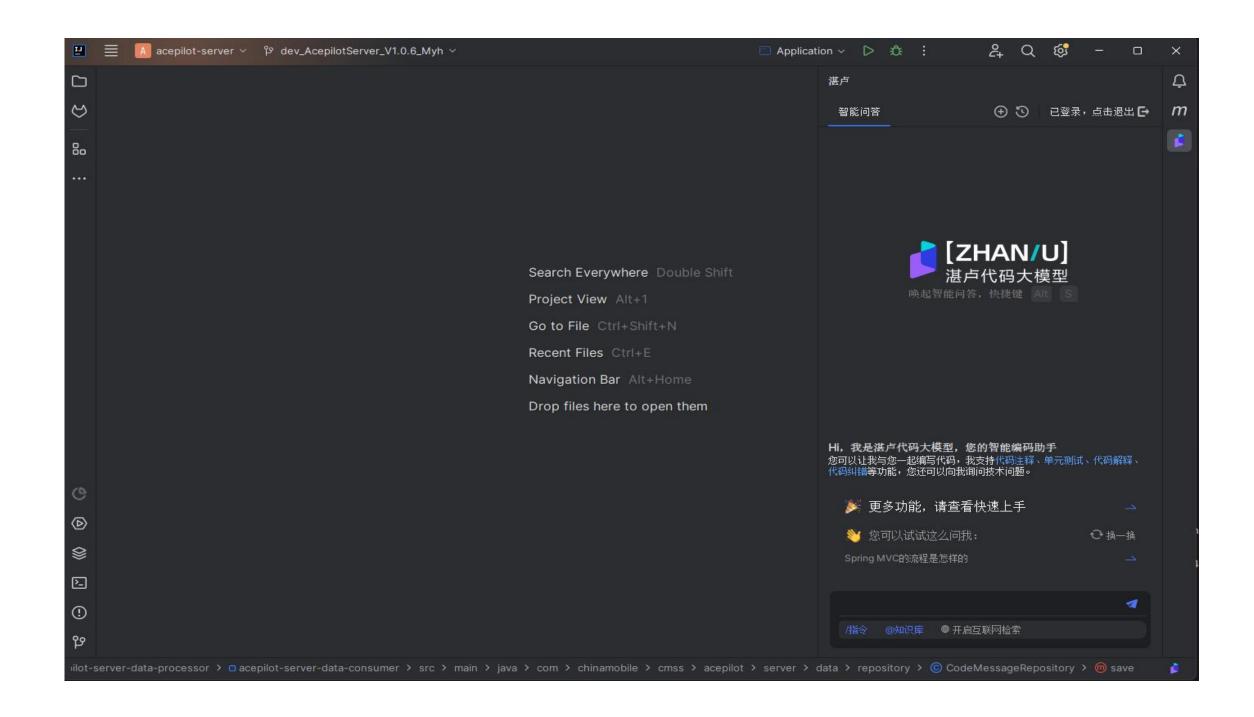


点击前往登录, 自动跳转至插件的登录页。



### 步骤四:

登录成功后,进入湛卢首页。

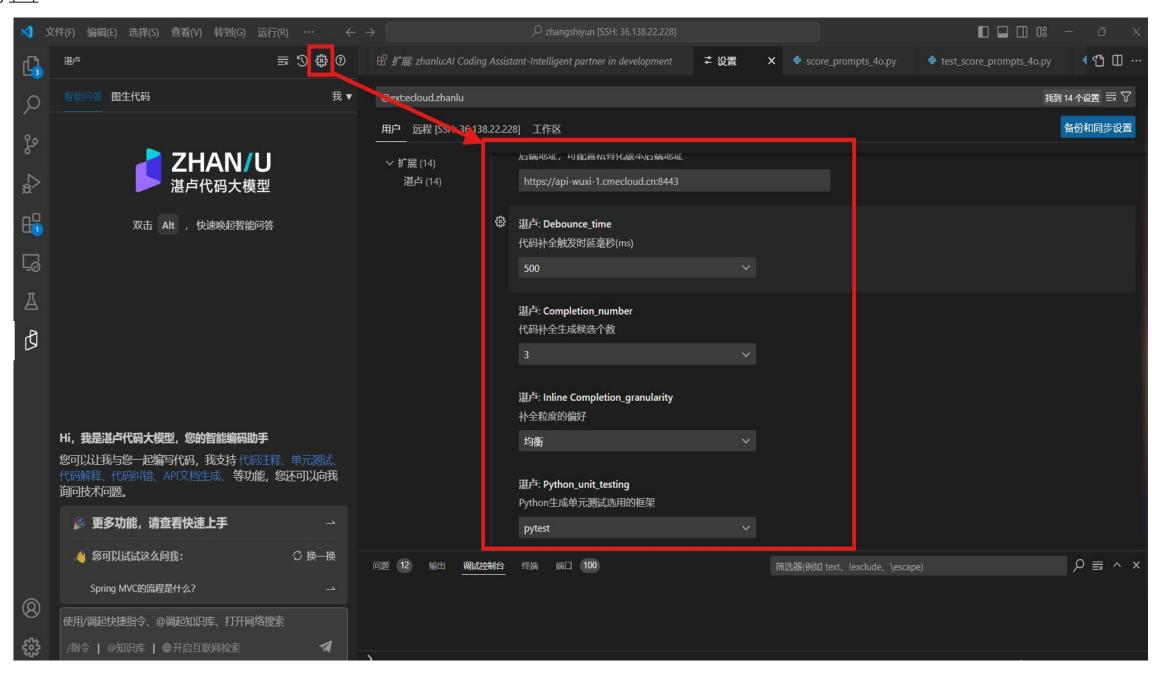


## 二、使用前配置

## 2.1 VS Code中的插件配置

#### ▶ 步骤一:

点击插件上端的配置按钮,进入配置页面,可根据实际需求更改下图的默认配置。



#### ▶ 步骤二:

【**补全配置**】在设置页面中,可以配置编辑器内代码补全的以下四个方面:

(1) **触发延时**:配置从停止输入到触发获取补全建议请求的延时,也可配置为手动,此时仅当按下热键ctrl + k时获取补全建议,无触发时延。

#### (2) 补全偏好:

行级补全:优先保证响应速度,返回单行建议;

平衡:兼顾响应速度和补全建议长度;

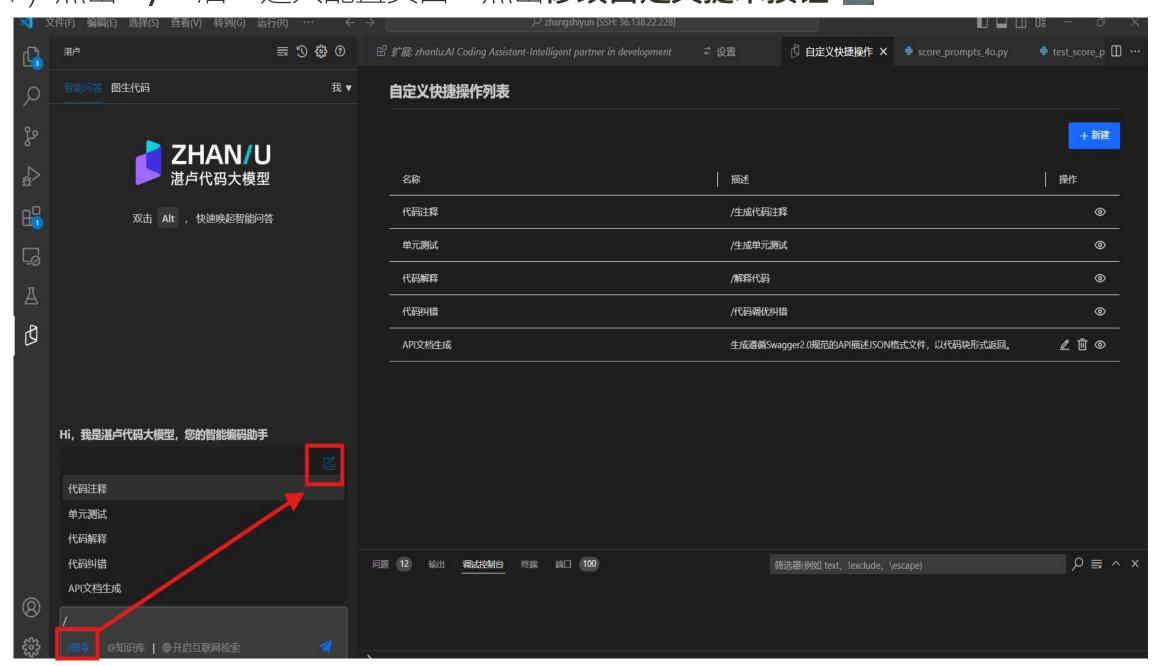
最大长度:依据配置的代码补全的max\_tokens,在最大允许范围内自行确定返回内容。

- (3)补全建议返回的最大token数
- (4)候选建议数量:设置返回候选条目的数量。
- 【其他配置】 还支持配置单元测试框架类型,以及问答的最大token数。
- 可根据需求自行配置,配置完成后,返回 Visual Studio Code 即可体验港 卢的各项功能。

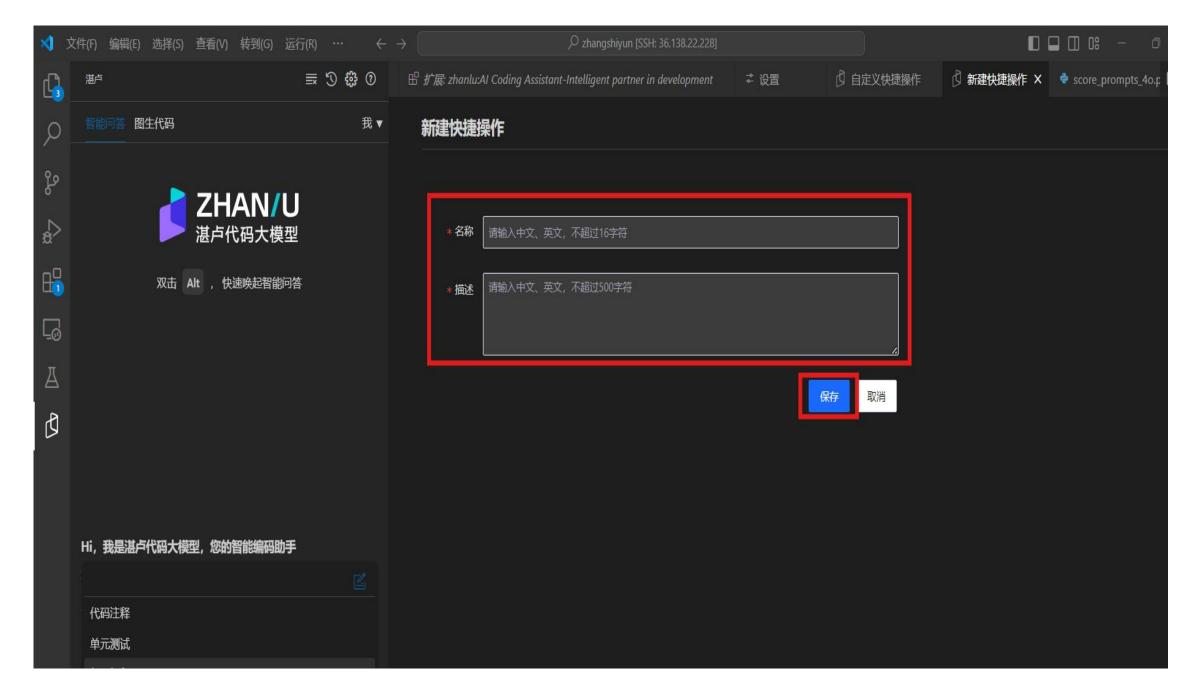
#### ▶步骤三:

【自定义prompt配置】可根据实际需求,自定快捷操作。

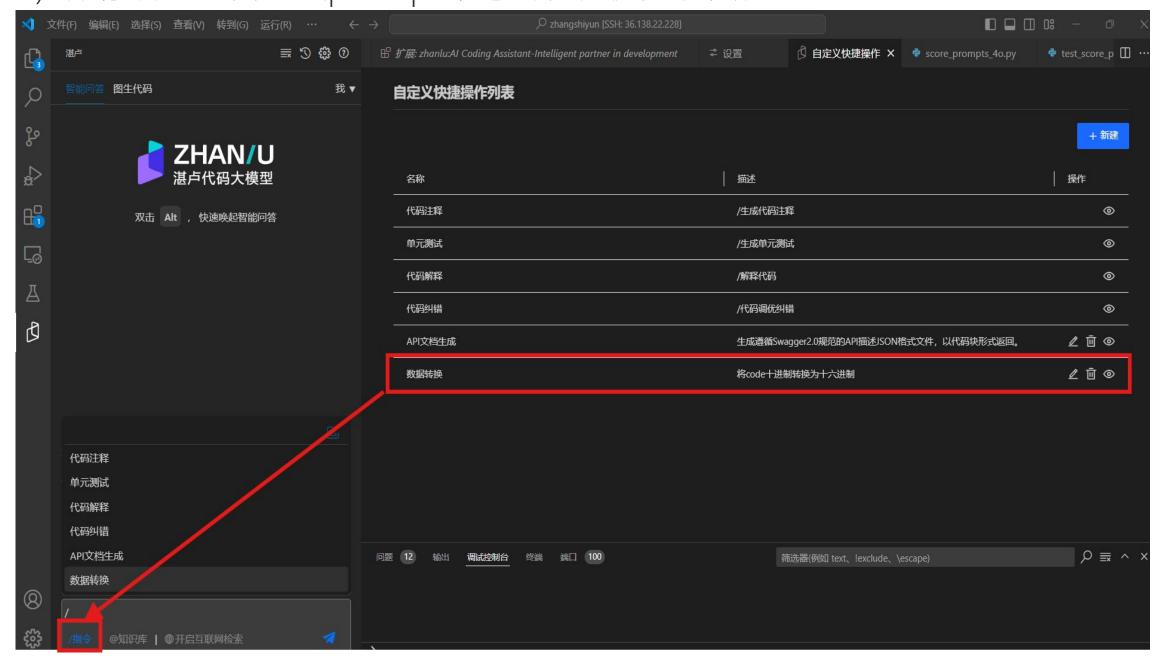
(1)点击"**/**"后,进入配置页面,点击**修改自定义提示按钮 🗾**。



(2)点击"新建",进行快捷操作配置,配置完成后保存。



(3)保存后,此自定义prompt即可出现在快捷触发位置。



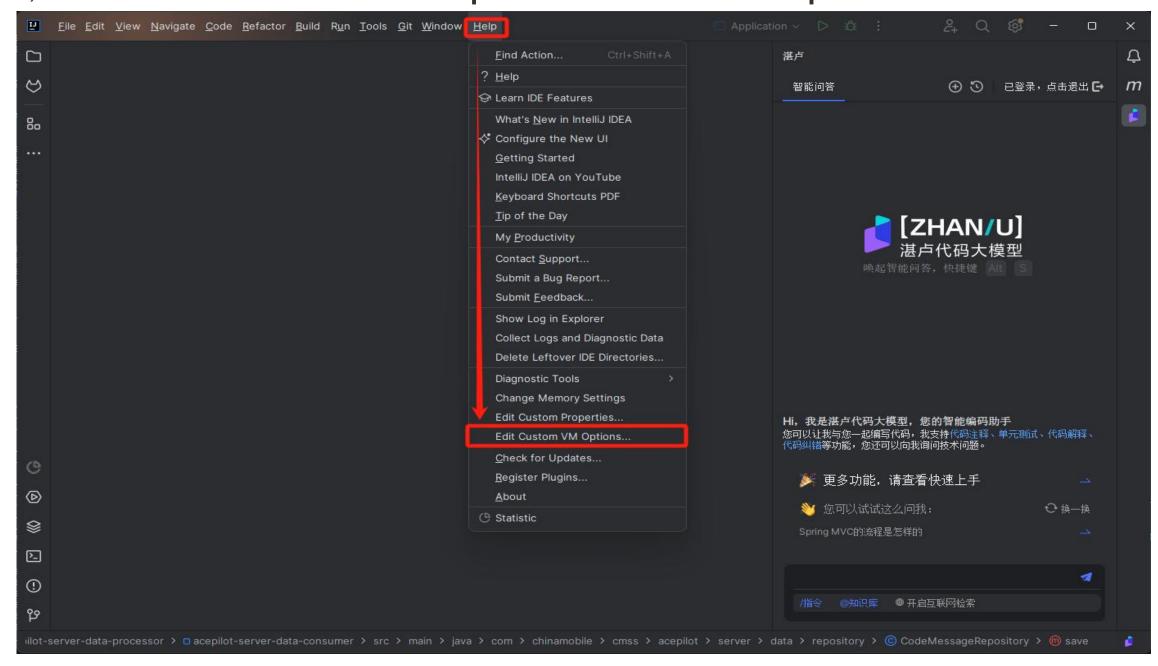
(4)使用时,先选中目标代码,再在问答输入框输入"/",选择对应提示语即可。

## 2.2 JetBrains中的插件配置

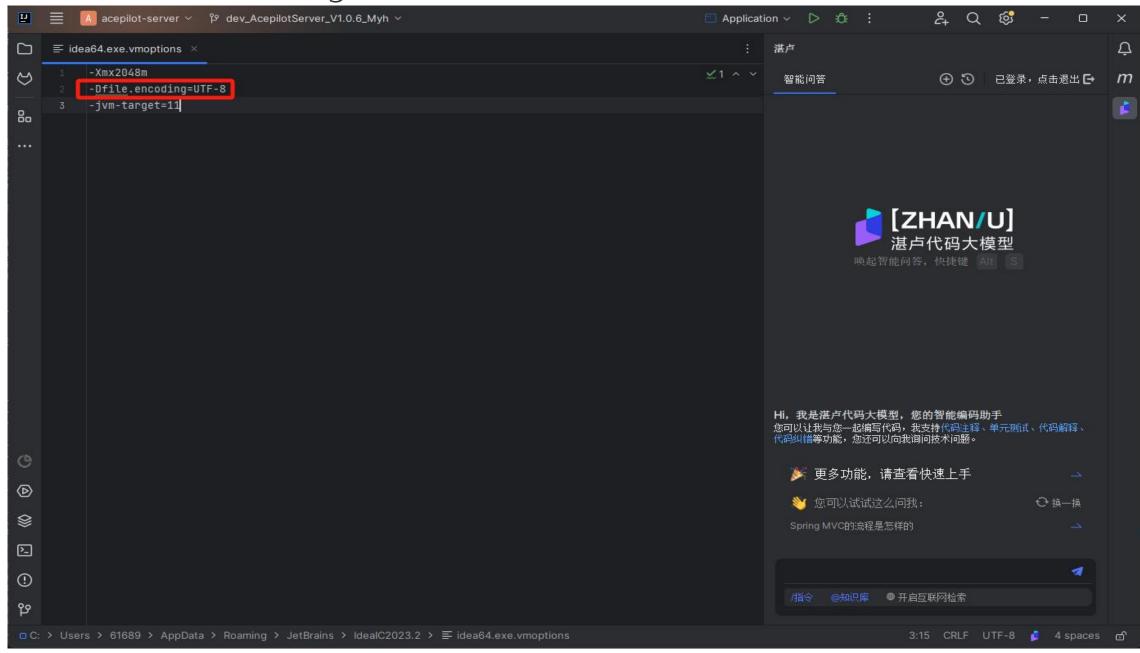
#### ▶ 步骤一:

中文显示问题配置。

(1) IDE**顶部菜单栏**中,选择**Help-Edit Custum VM Options**。



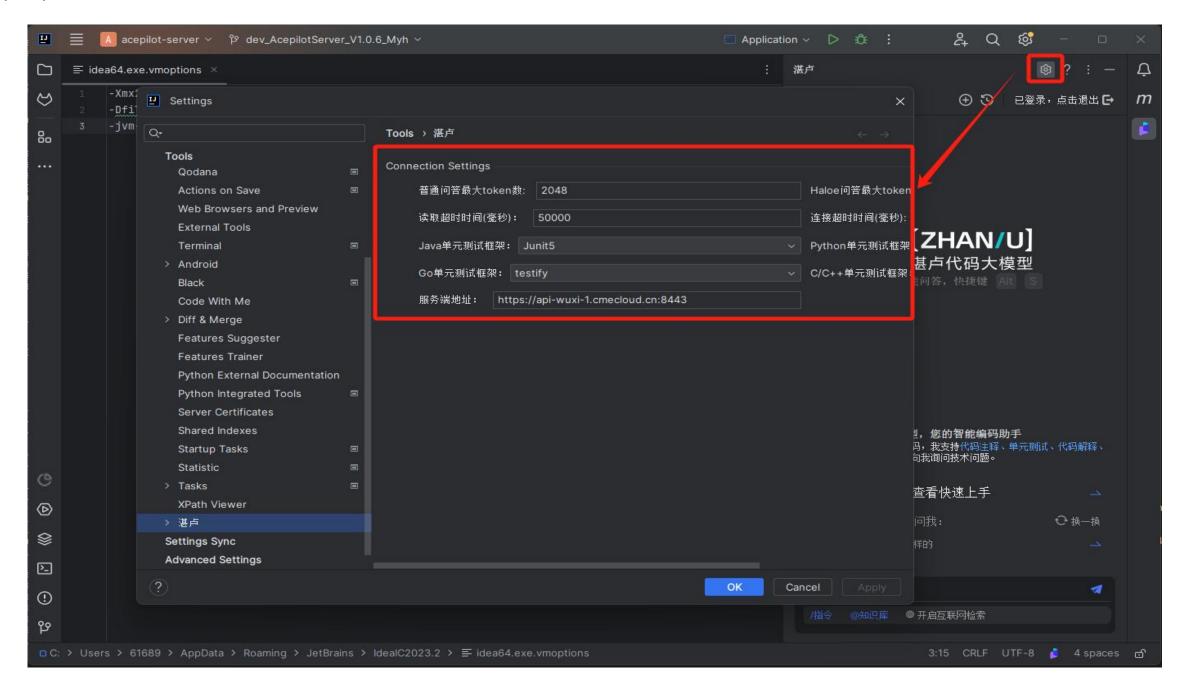
(2)增加-Dfile.encoding=UTF-8 后重启IDE。



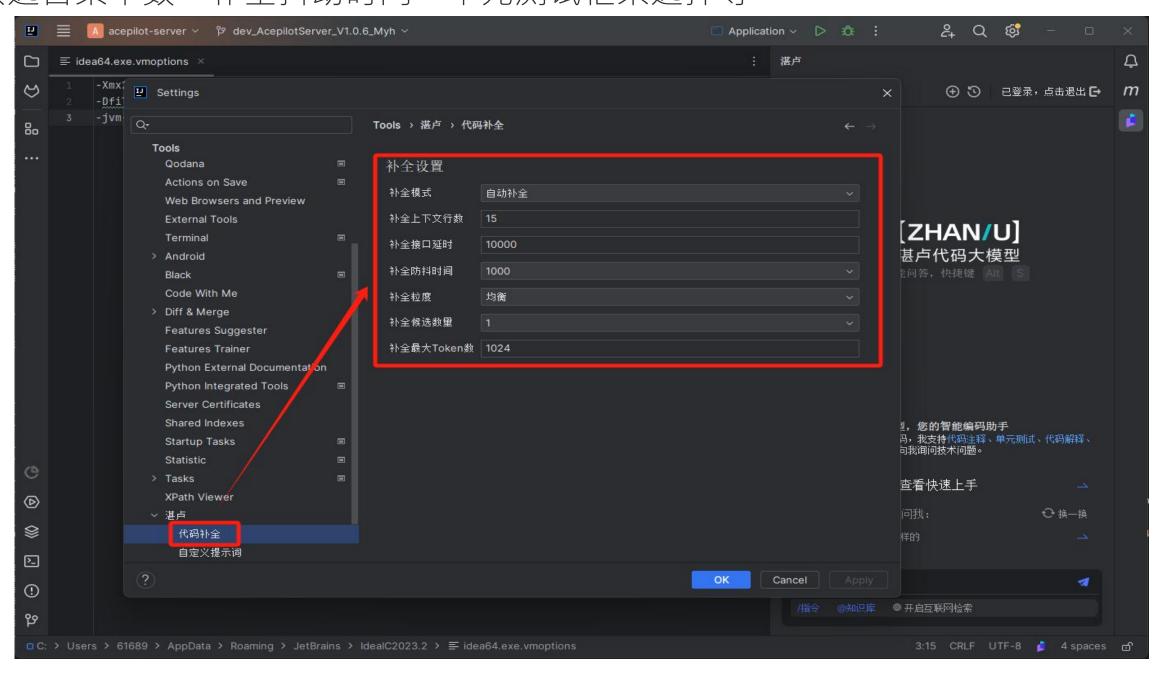
#### ▶步骤二:

#### 通用配置。

(1)点击插件顶端配置按钮 ◎ ,进入配置页面。

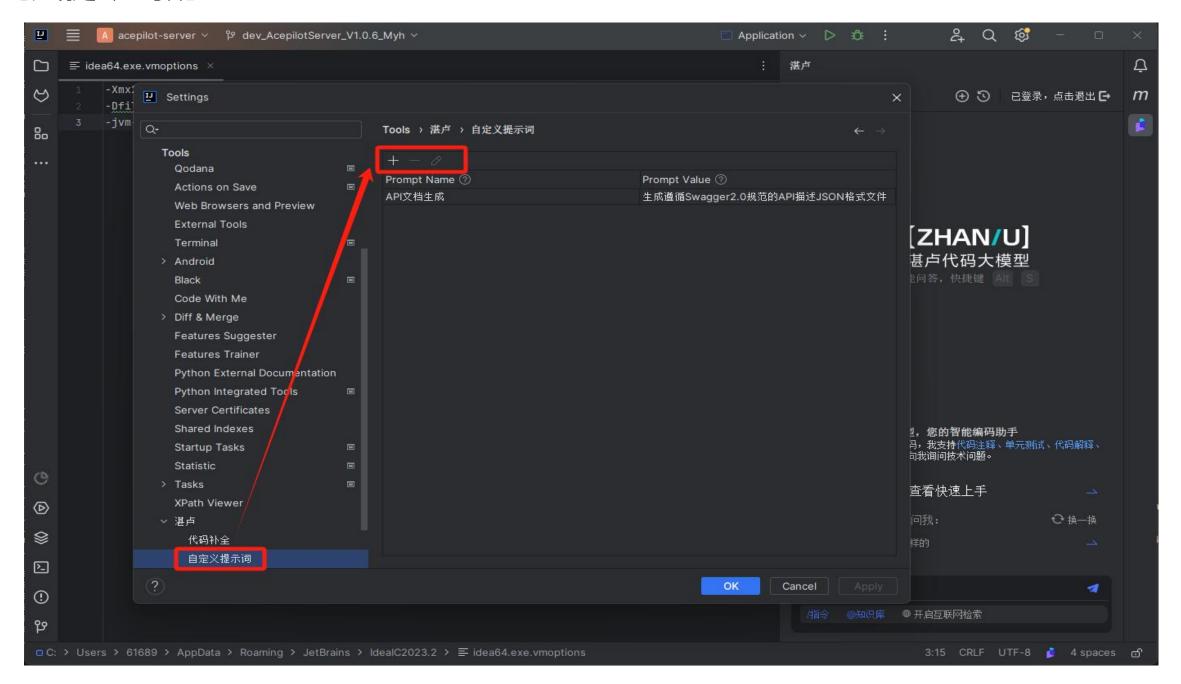


(2)可根据实际需求配置,如单次接受的最大token数,代码补全粒度、补全候选答案个数、补全抖动时间、单元测试框架选择等。



#### ▶步骤三:

自定义提示词配置。



下面详细描述下补全和自定义提示词的配置:

#### 【代码补全】

(1) 触发延时:配置从停止输入到触发获取补全建议请求的延时,也可配置为手动,此时仅当按下热键ctrl + alt+ k时获取补全建议,无触发时延。

#### (2) 补全偏好:

行级补全:优先保证响应速度,返回单行建议;

平衡:兼顾响应速度和补全建议长度;

最大长度:依据配置的代码补全的max\_tokens,在最大允许范围内自行确定返回内容。

- (3) 补全建议返回的最大token数
- (4) 候选建议数量:设置返回候选条目的数量。

#### 【自定义提示词】

#### (1) 如何进行自定义提示?

提示模板中可以包含 {{code}} 来标识该提示需要抓取用户框选的代码,一并提交湛卢处理。提示模板中可以包含用户填空,可以使用 {{input}} 来标识需要填写对应位置的信息后再通过主动点击 发送图标 提问。为了更好地提示用户 ,您可以为当前填空编写占位符,在内容为空时显示提示。如 {{input:language}} 将会显示可编辑文本框,并以 language 作为提示信息占位显示。以下是几个示例:

编写一个 Dockerfile 模板 将{{input:十进制数}}转换为十六进制 提取以下代码中的所有函数签名

{{code}}

将以下代码翻译为{{input:目标语言}}语言代码 {{code}}

#### (2) 如何使用?

用户可根据实际需求,自定义提示模板,以更好地适用于您的使用场景。在输入框点击"**/快捷指令**",插件将在上方生成对应的操作按钮。

#### ▶ 步骤四:

配置完成后重启IDE,即可体验湛卢智能编码助手的各项功能。

## 三、操作指南

#### > 快捷键介绍:

首先介绍下的快捷键,使用Tab键采纳推荐,使用Esc键取消推荐,当触发模式为自动触发时,使用Enter键或Ctrl+K触发多行补全;当为手动触发模式,仅 Ctrl+K触发多行补全。候选数量大于 1,可以使用 Alt+[ 以及 Alt+]进行翻页浏览,确定要接受的建议项后,使用 Tab 键插入编辑器。您也可以部分接受结果,使用Ctrl+↓ 接受单行建议。VS Code编辑器在触发补全事件后,如遇用户的其他选择、移动、输入行为,则会取消补全操作。VS Code和 Jetbrains 具体使用方式将在下面介绍。

#### (1) VS Code

快捷功能	快捷键 (Windows)	快捷键 (Mac)
召出问答框	Alt+Alt	cmd+cmd
接纳补全建议	Tab	Tab
取消补全建议	Esc	Esc
多个补全候选答案 生成时,前后选择	Alt +[(Alt+] )	cmd+[(cmd+]
补全单行接受	Ctrl+↓	Ctrl+↓
手动触发	Ctrl+K	Ctrl+K

#### (2) Jetbrains

快捷功能	快捷键 (Windows)	快捷键 (Mac)
召出问答框	Alt+S	cmd+S
接纳补全建议	Tab	Tab
取消补全建议	Esc	Esc
多个补全候选答案生 成时,前后选择	Alt +[(Alt+])	cmd+[(cmd+]
补全单行接受	Ctrl+Alt+↓	Ctrl+cmd+↓
手动触发	Ctrl+Alt+K	Ctrl+cmd+K

## 3.1 代码补全模式

湛卢代码大模型为多种语言和各种框架提供了建议,但特别适用于 Python, JavaScript, Java, Go, C++。

以下以Java语言为例,湛卢代码大模型将自动显示灰色文本建议(即自动代码补全或代码续写)。

#### 3.1.1 行补全

用户在IDE编写某行代码时,根据改行前序内容,编码过程中稍做停顿(默认1s,您可自行配置),自动补全该行内容,补全内容灰色显示,一经触发,编辑器状态栏右下的湛卢( ) 图标将指示现在的请求状态。按下**Tab**键接受补全自动将代码补充,按下**Esc**键或鼠标焦点移开则取消此次补全。

```
acepilot-server > p dev_AcepilotServer_V1.0.6_Myh >
                                                                                                      Application V D 🌣 :
m pom.xml (api-fegin-client)

    DataConsumer.java

                                                                                                  application.yml
8
                 public void save(UserModel userModel) {
                    String createTimeStr = userModel.getCreateTime();
                        Date createTime = DateTimeUtils.inputDateFormat.parse(createTimeStr);
                        String formattedDate = DateTimeUtils.outputDateFormat.format(createTime);
                        String index = Objects.equals(userModel.getSource(), b "source-c")
                                ? IndexName.USER_LOG_PRE.getIndex() + formattedDate
                                : getUserLogIndex();
                        this.bulkProcessor.add(new IndexRequest(index)
                                .source(JsonUtils.objectToJson(userModel), XContentType.JSON)
                    } catch (ParseException e) {
                        log.error("Error parsing date: {}", e.getMessage());
                    String searchDay = DateTimeUtils.timestamp(LocalDateTime.now(), DateTimeUtils.dateIndexFormatter);
①
୧୨
data-consumer > src > main > java > com > chinamobile > cmss > acepilot > server > data > repository > 🕲 UserRepository > 🎟 getUserLogIndex 186:9 CRLF UTF-8 🔆 👂 4 spaces 🗗
```

#### 3.1.2 块补全

用户在IDE编写代码时,当待触发推荐的上下文,具备明显的完整逻辑关系(如一个新的方法、函数、判断、循环体等),当为自动触发模式时,按下Enter键或按下Ctrl+K可推荐一个完整的代码块(您可自行配置手动触发或者自动触发)。一经触发,编辑器状态栏右下的 湛卢( ) 图标将指示现在的请求状态。

根据用户编写的上下文代码情况,如果能触发后端代码补全,则出现补全内容,如果不能触发代码补全,则不会显示补全内容。按下**Tab**键接受补全自动将代码补充,按下**Esc**键或鼠标焦点移开则取消此次补全。

```
Application ∨ ▷ ☼ :
                                                                                                                유 Q � - ㅁ ×
er.java
                                                 © SnowflakeldGenerator.java × © CodegenRepository.java © UserRepository.java © CodeMessageRep ∨ :
                                                                                                                       ●1 ▲11 × 1 ∧ ∨
\mathfrak{S}
               lastTimestamp = timestamp;
               return ((timestamp - twepoch) << timestampLeftShift) |</pre>
            protected long tilNextMillis(long lastTimestamp) {
             long timestamp = timeGen();
            protected long timeGen() {
               return System.currentTimeMillis();
①
થ
```

#### 3.1.3 注释生成补全

用户在IDE编写代码时,在注释中例如写"//交换两个元素",按下Enter键或 Ctrl+K,触发代码补全,在IDE中显示补全内容。按下Tab键接受补全自动将代码补充,按下Esc键或鼠标焦点移开则取消此次补全。

```
Application ∨ ▷ 🌣 :
                                                                                                                             24 Q ®
er.java
                                                        © SnowflakeldGenerator.java 

○ CodegenRepository.java 
○ UserRepository.java
                                                                                                                              © CodeMessageRep ∨ :
                        timestamp = tilNextMillis(lastTimestamp);
                     sequence = 0L;
                 lastTimestamp = timestamp;
                return ((timestamp - twepoch) << timestampLeftShift) |
                        (datacenterId << datacenterIdShift) |
                        (workerId << workerIdShift) |
             protected long tilNextMillis(long lastTimestamp) {
                 return System.currentTimeMillis();
$
2
(1)
જ
 cessor > 🛘 acepilot-server-data-consumer > src > main > java > com > chinamobile > cmss > acepilot > server > data > utils > 🕲 SnowflakeldGenerator 56:5 CRLF UTF-8 🚦 4 spaces 🗗
```

## 3.2 Chat模式

湛卢通过连续对话的方式为您解决问题。输入问题描述,湛卢为您生成代码。

#### (1) 支持多轮对话

(2) 支持指定开源知识的问答,当前已具备halo2组件库、haloe3组件库、网络安全规范、研发管理规范、代码规范、经分系统操作规范、BCOP操作手册、 MOP对接规范、全栈专属云ECSO规范、OpenAPI、资源编排知识库。

#### (4)快速召唤代码助手

方式1:随时通过**连按两次 Alt键 (JetBrains为Alt+S)**,召唤侧边栏代码助手。

方式2:在编辑器内单击**右键选择湛卢--**>功能按键,转到侧边栏代码助手提问或

使用预设提示提问。

您可以在**提问框中输入问题**,或通过**键入 / 选择预设提示**(部分操作可能需要用 户补全提示,完成补全后可以点击发送图标提交内容)开始对话。如在编辑器 内有活动的代码文件内容被选中,则会随问题一起发送;

对话结果为流式输出模式,即回答将在产生过程中逐步显示,以便干您提早审 阅结果,在这种模式下,您可以随时点击**取消生成**按钮来终止此次回答。

对于结果中可识别的代码内容,插件将以代码框形式显示,并提供生成到 **新文件、复制到剪贴板和插入到当前文件位置**的按钮,便干您的查看和操作。



#### 关于上下文

在提问过程中,湛卢会按照算法 token 长度要求,携带合适数量的上下文信息 一起发送。

或者您可以点击湛卢侧边栏顶部标题栏上的 + 按钮来新建对话,可以清空当前上 下文内容。



#### 历史记录

您可以通过 ↑/↓ 方向键调取当前对话窗的历史提问,通过 Enter 键直接定 位到历史对话处。

为了提供更加连续的开发体验,对于侧边栏代码助手页面,湛卢会将历史会话信 息本地缓存,这些信息会随着您的问答,操作(删除、清空)等同步更新内容。



#### 芝 互联网检索

点击对话框内"互联网问答"按钮, 并输入相关关键词,可以发起互联网检索,湛 卢会展示检索结果,使您不离开 IDE 就能浏览需要的信息。再次点击"互联网问 答"按钮,可关闭互联网搜索,基于大模型问答。



#### 知识库问答

击对话框内@知识库按钮,可展示所有知识库,点击选择与您问题相关的 知识库,输入关键信息,点击发送即可,知识库中@湛卢,支持在问答框询问 湛卢相关使用问题。

## **评**价回答结果

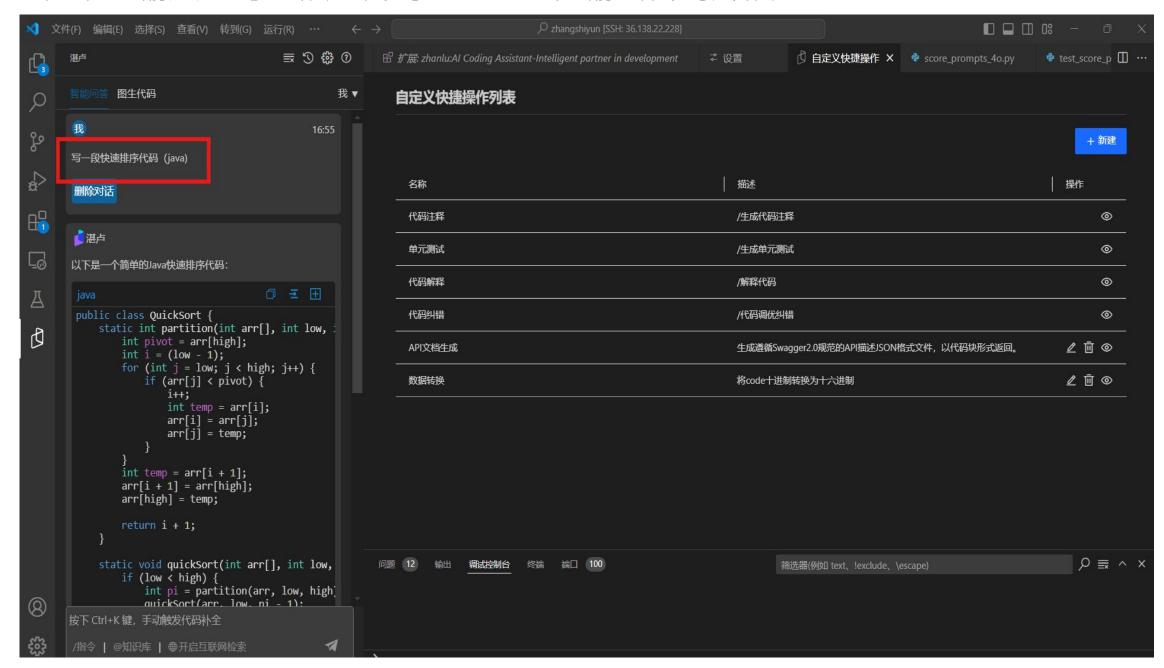
在回答结束或回答被终止后,答案底部将显示评价按钮,如果您接受了湛卢的隐私协议,当您点击对应按钮时,我们将收集你的评价反馈,这对我们改进算法和产品设计很有帮助,十分感谢您的支持。

下面对各个功能进行详细介绍:

### 3.2.1 生成代码(支持多轮对话)

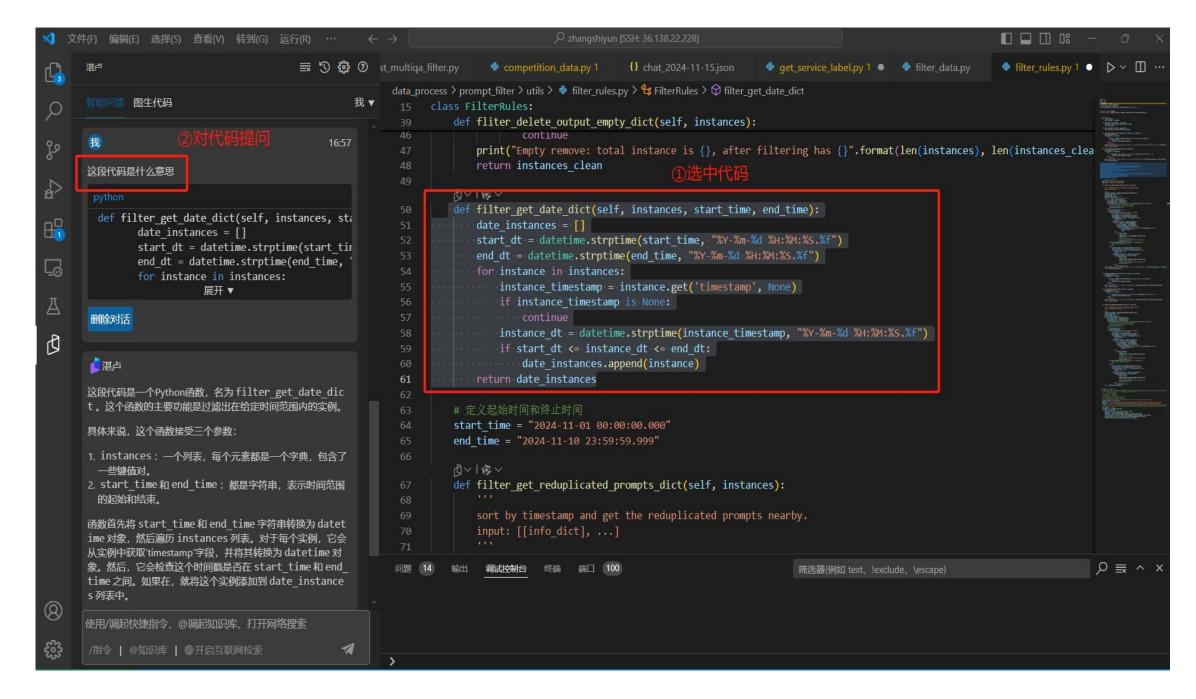
#### (1) 直接提问

在chat框中,输入"写一段xx代码",chat框输出代码片段.



#### (2) 结合代码提问

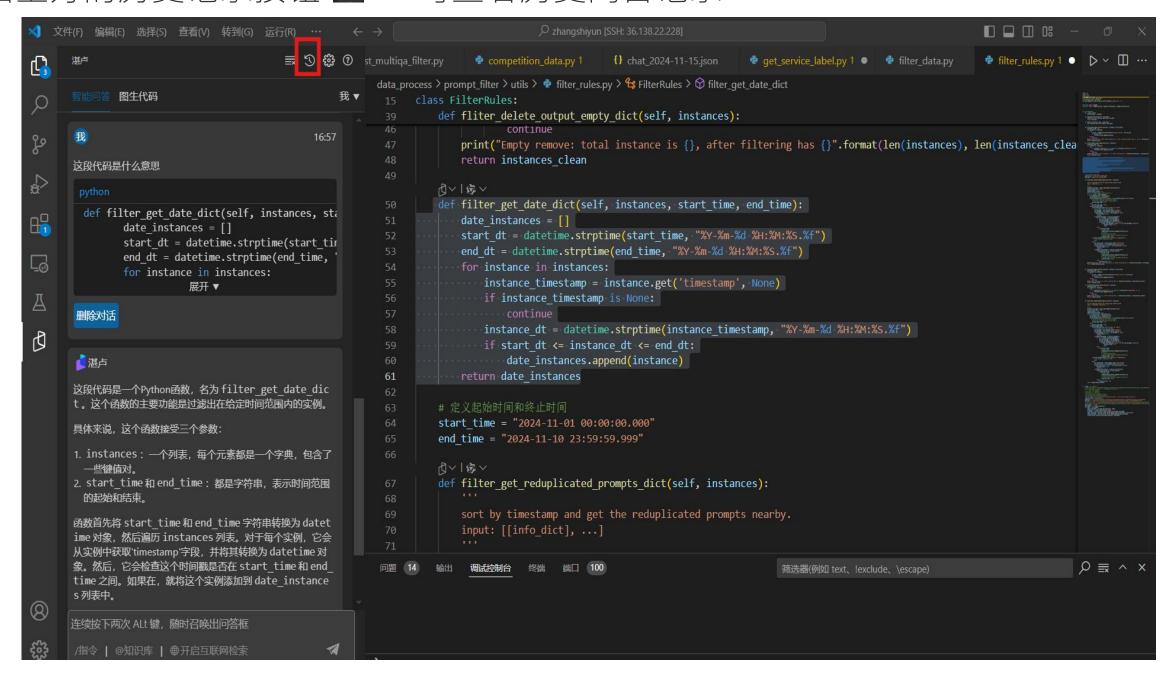
用户可以在编辑器中选中代码,再输入款加上问题描述,点击发送。



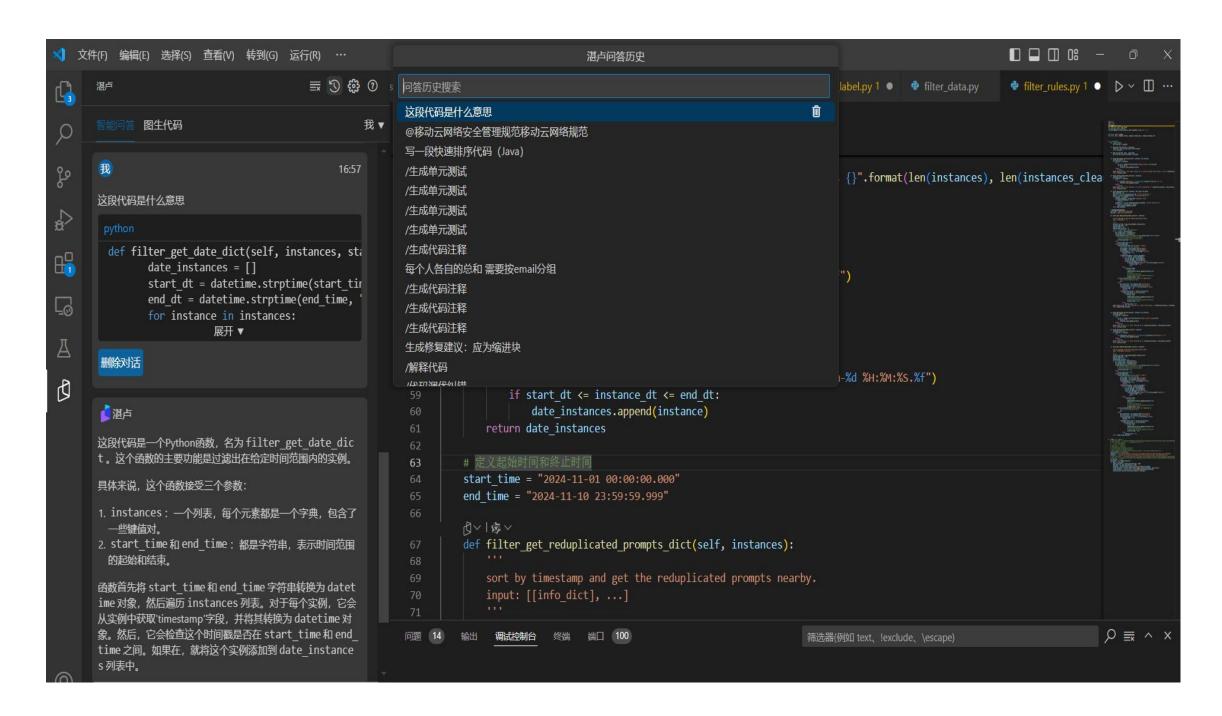
用户对生成的答案可进行插入、复制、生成到新文件、点赞和踩,意见反馈。

#### (3) 历史记录查看

点击上方的历史记录按钮 💿 ,可查看历史问答记录



点击对应的记录,可直接点定位到相应问答位置。

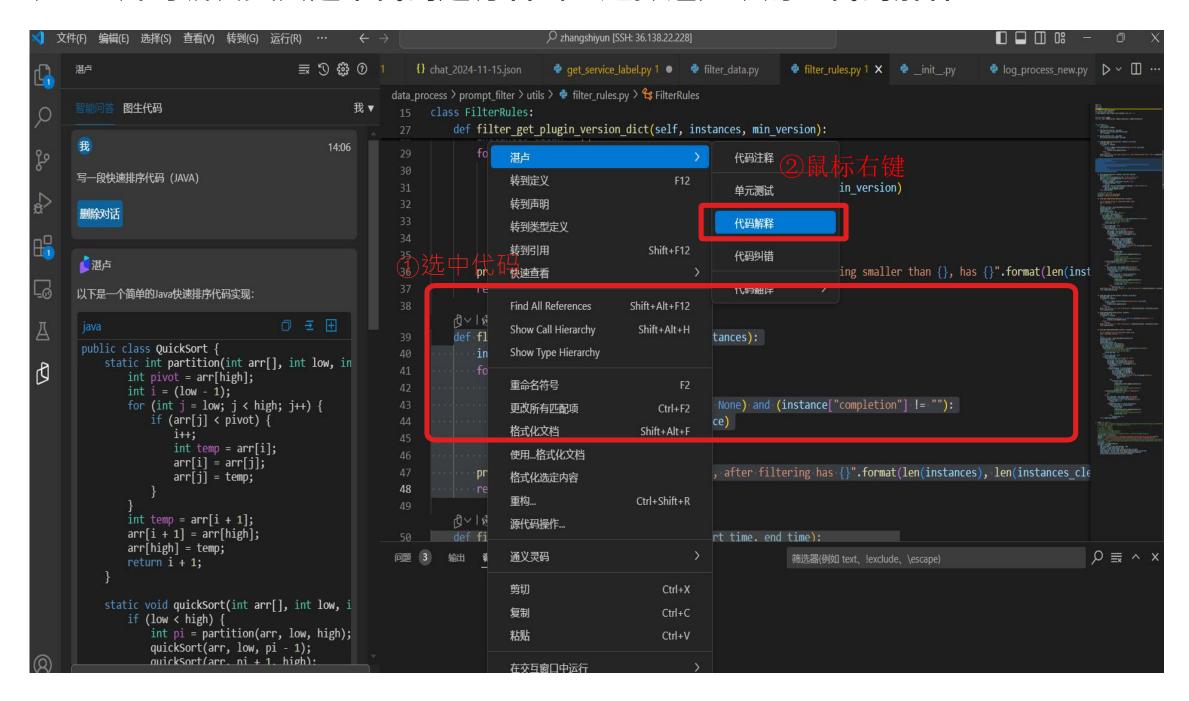


#### 3.2.2 解释代码

支持对用户代码进行解释,触发方式支持选中代码**右击选择按钮触发**及选中代码后在**对话框中输入命令触发**。

#### 触发方式1:右击触发

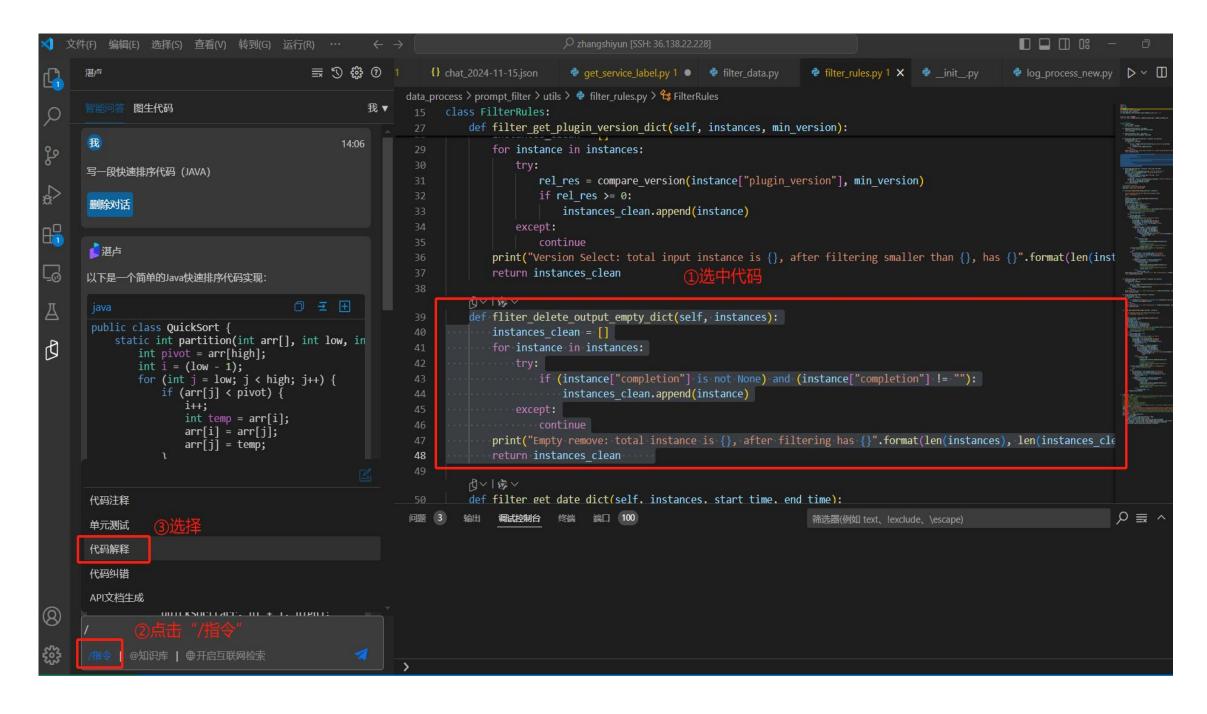
用户在IDE代码编辑页面**选中代码**进行**右击**,选择**湛卢**下的"**代码解释**"。



智能助手在对话框中对该代码进行解释,可对回答进行可以进行插入及复制、生成到新文件、差异对比、点赞和踩,意见反馈,操作同上。

#### ■触发方式2:选中后通过命令触发

用户在代码编辑页面选中代码,在对话框中**输入命令"解释代码"**或直接**点击"/代码解释"**触发快捷操作选择解释代码,对话框中自动将选中代码带入对话中。



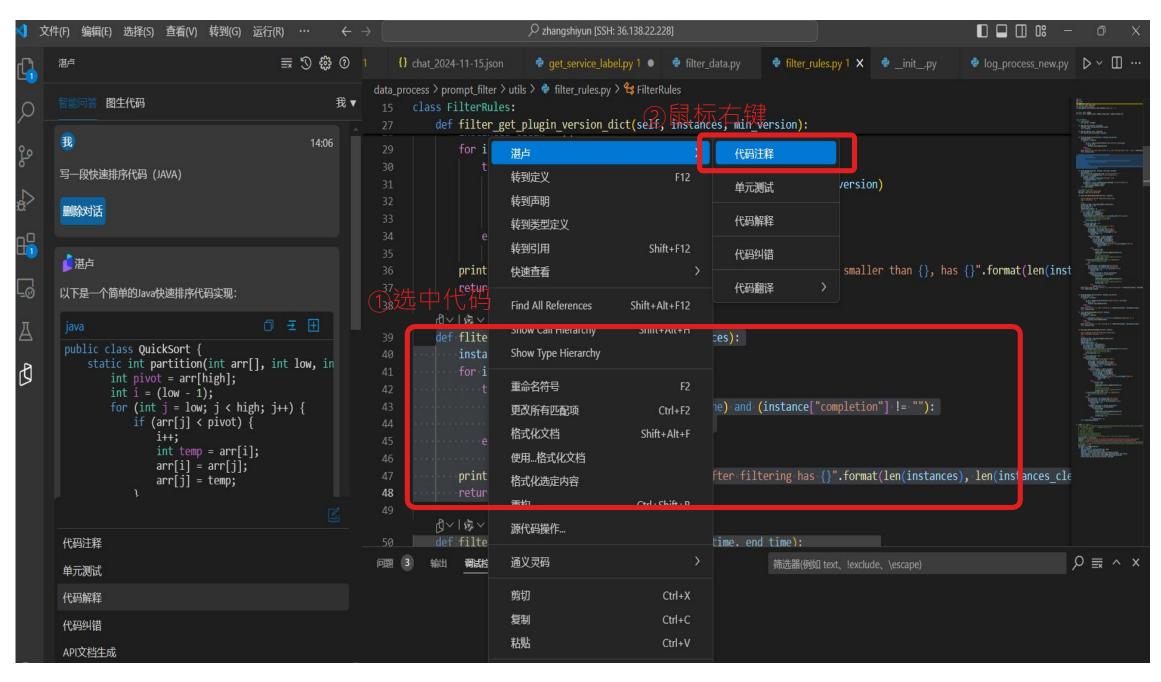
智能助手在对话框中对该代码进行解释,可对回答进行可以进行插入及复制、生成到新文件、差异对比、点赞和踩,意见反馈,操作同上。

#### 3.2.3 生成代码注释

用户选中一个函数,则进行函数注释和函数中行间代码注释,如仅选择一行代码,则进行该行代码注释。触发方式支持选中代码**右击选择工具触发**及选中在对**话框中输入命令触发**:

#### ▶触发方式1:右击触发

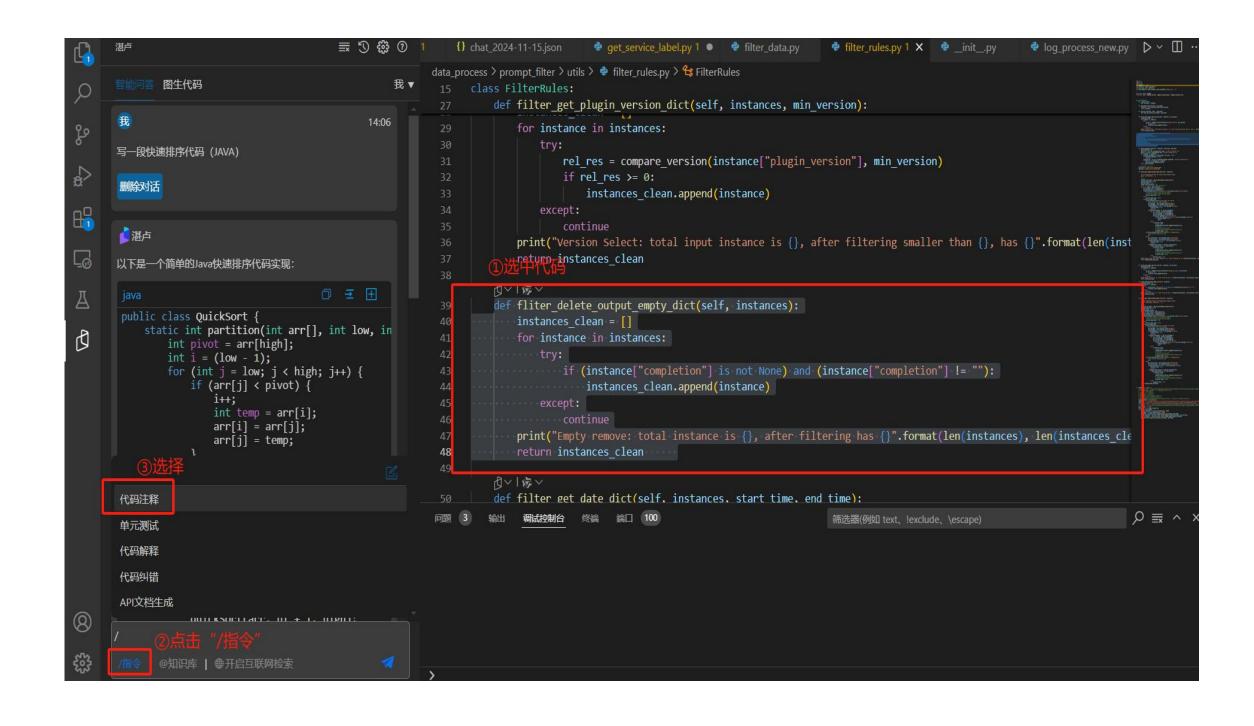
选中代码后,右击选择相应按钮触发:



智能助手在对话框中对该代码进行解释,可对回答进行可以进行插入及复制、生成到新文件、差异对比、点赞和踩,意见反馈,操作同上。

#### ■触发方式2:选中后通过命令触发

用户在代码编辑页面选中代码,在对话框中**输入命令"生成代码注释"**或点击"/代码注释"触发快捷操作选择"生成代码注释",对话框中自动将选中代码带入对话中。

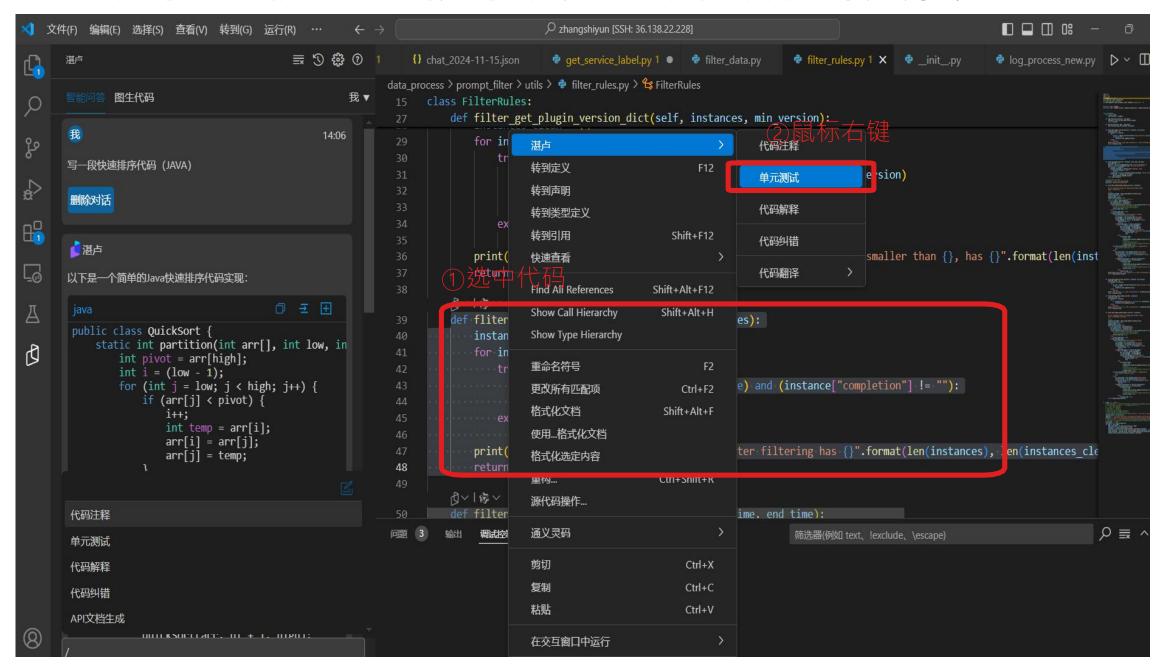


#### 3.2.4 生成单元测试

支持用户单元测试代码生成,触发方式支持选中代码右击选择工具触发及选中在对话框中输入命令触发。

#### ▶触发方式1:右击触发

用户在IDE代码编辑页面选中代码进行右击,选择湛卢下的"**单元测试**"



智能助手在对话框中对该代码进行解释,可对回答进行可以进行插入及复制、生成到新文件、差异对比、点赞和踩,意见反馈,操作同上。

#### ■触发方式2:选中后通过命令触发

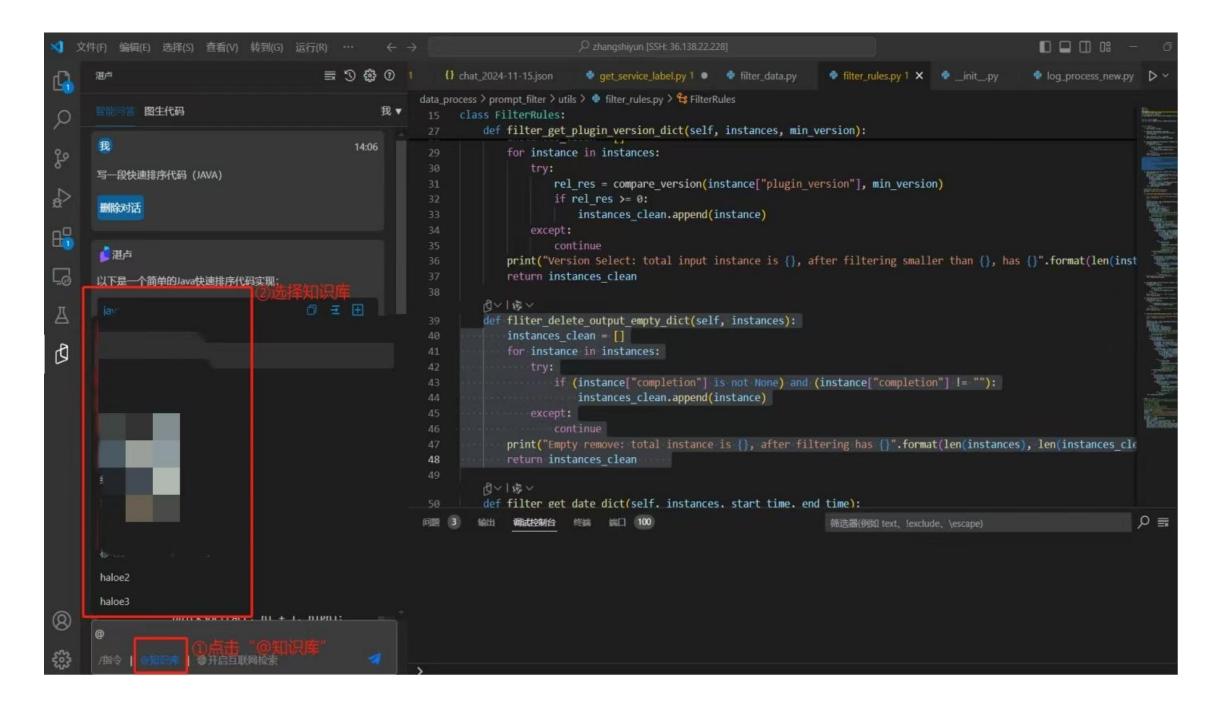
用户在代码编辑页面选中代码,在对话框中**输入命令"生成单元测试**"或点击"**/单元测试"**触发快捷操作,对话框中自动将选中代码带入对话中。智能助手在对话框中对该代码生成单元测试,代码片段可以进行插入及复制、生成到新文件、差异对比、点赞和踩,意见反馈,操作同上。

#### 3.2.5 指定代码库智能问答

支持针对指定代码仓库进行检索,或针对制定知识库问答,实现更精准的代码生成。

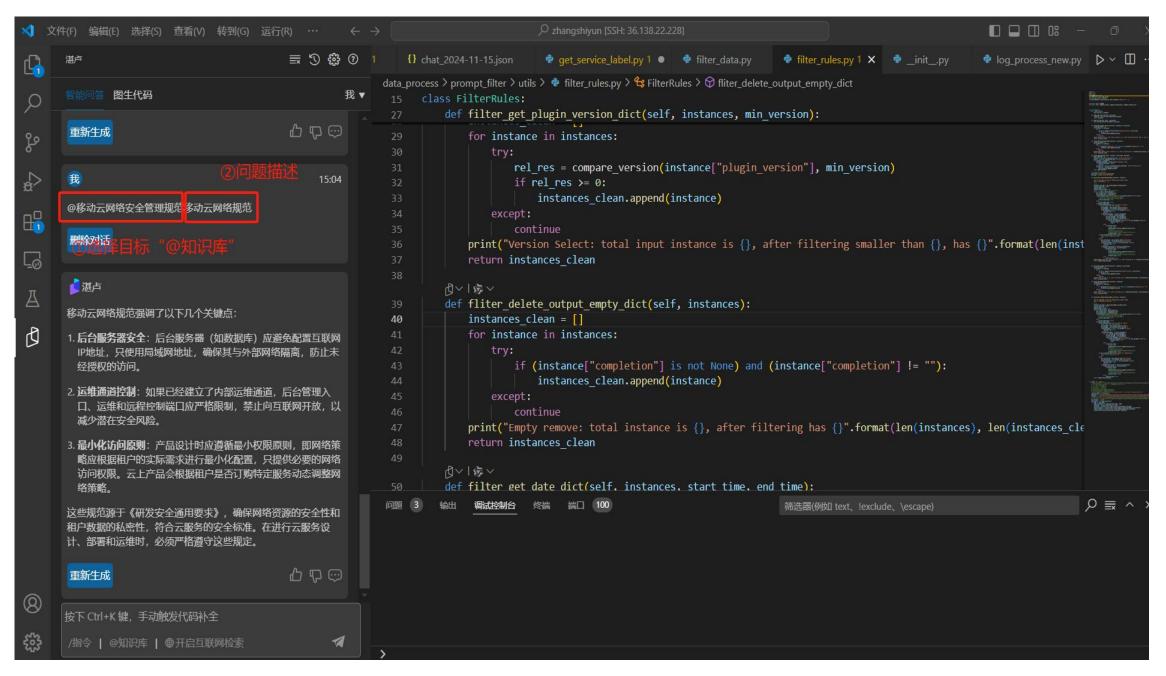
#### 【代码仓库问答

点击问答输入框下方"@知识库",选择目标代码仓库,并对问题加以描述,点击 发送。



#### 知识库问答

点击问答输入框下方"@知识库",选择目标知识库,并对问题加以描述,点击发送。



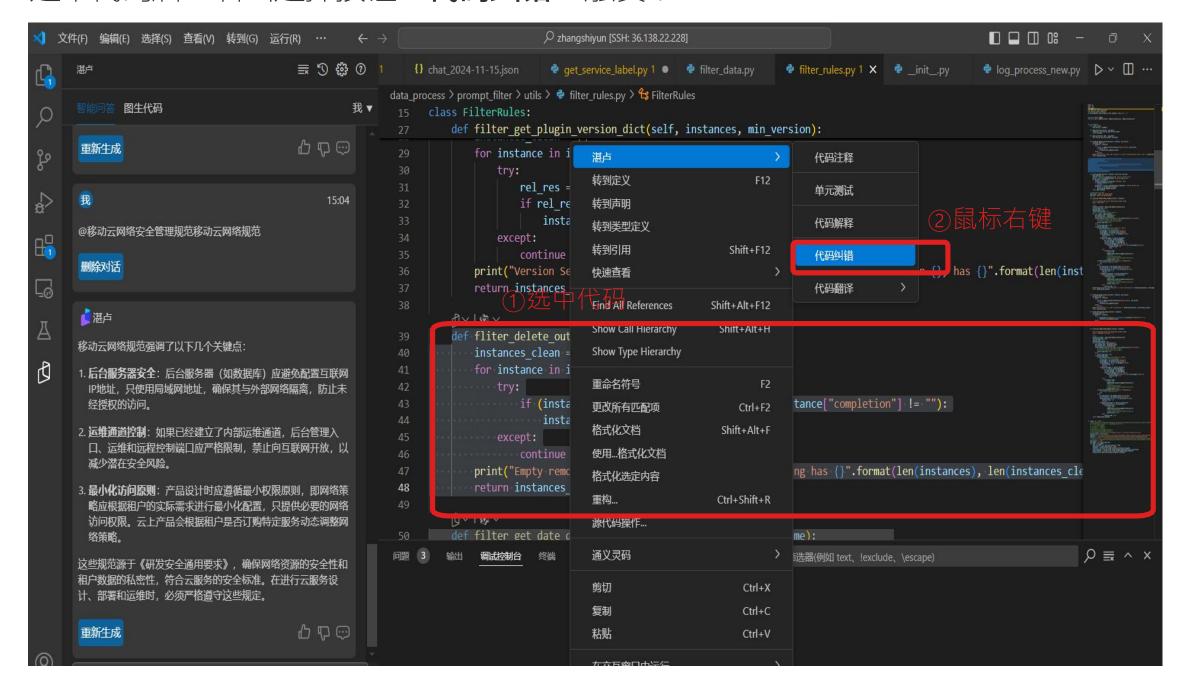
点击"@知识库"中的"@湛卢",支持在问答框询问湛卢相关使用问题。

#### 3.2.6 代码调优纠错

触发方式支持选中代码右击选择工具触发及选中在对话框中输入命令触发:

#### ■触发方式1:右击触发

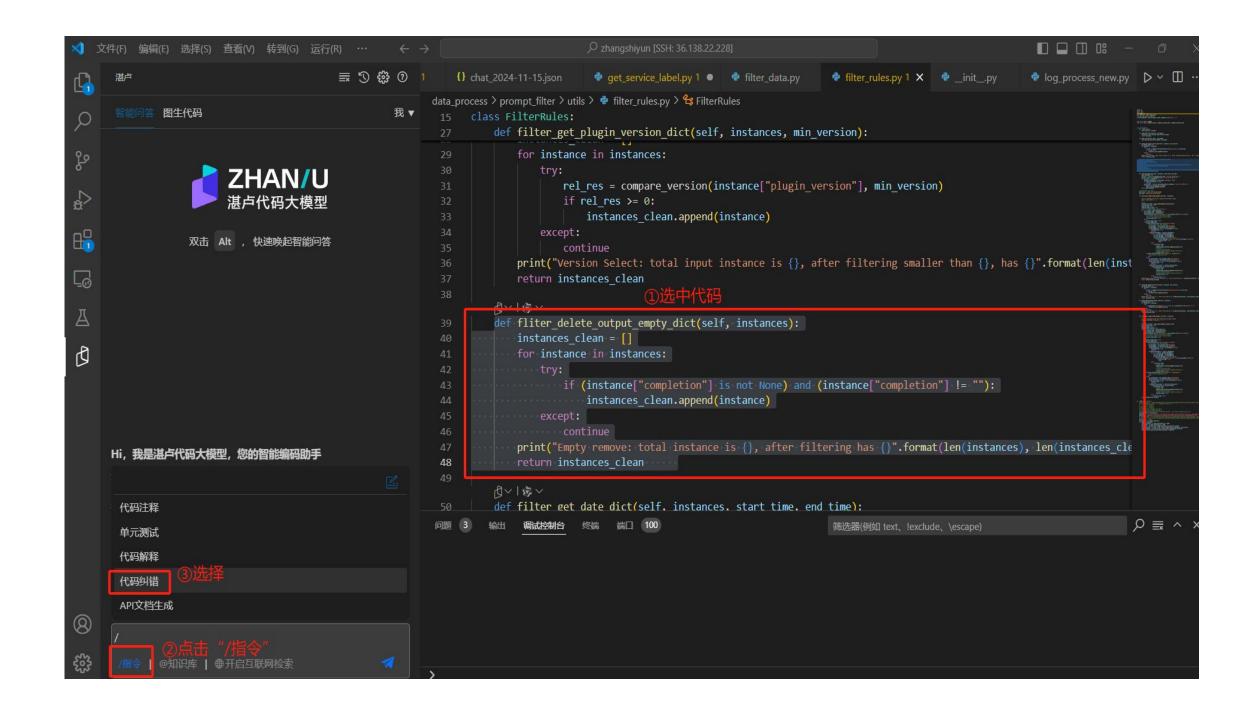
选中代码后,右击选择按钮"代码纠错"触发:



智能助手在对话框中生成对该段代码问题的描述、优化后的代码等,优化后的代码片段可以进行插入及复制、生成到新文件、差异对比、点赞和踩,意见反馈,操作同上。

#### 触发方式2: 选中后通过命令触发:

用户在代码编辑页面选中代码,在对话框中输入命令"代码调优"或点击"/代码纠错"触发快捷操作,对话框中自动将选中代码带入对话中,后续操作同上。

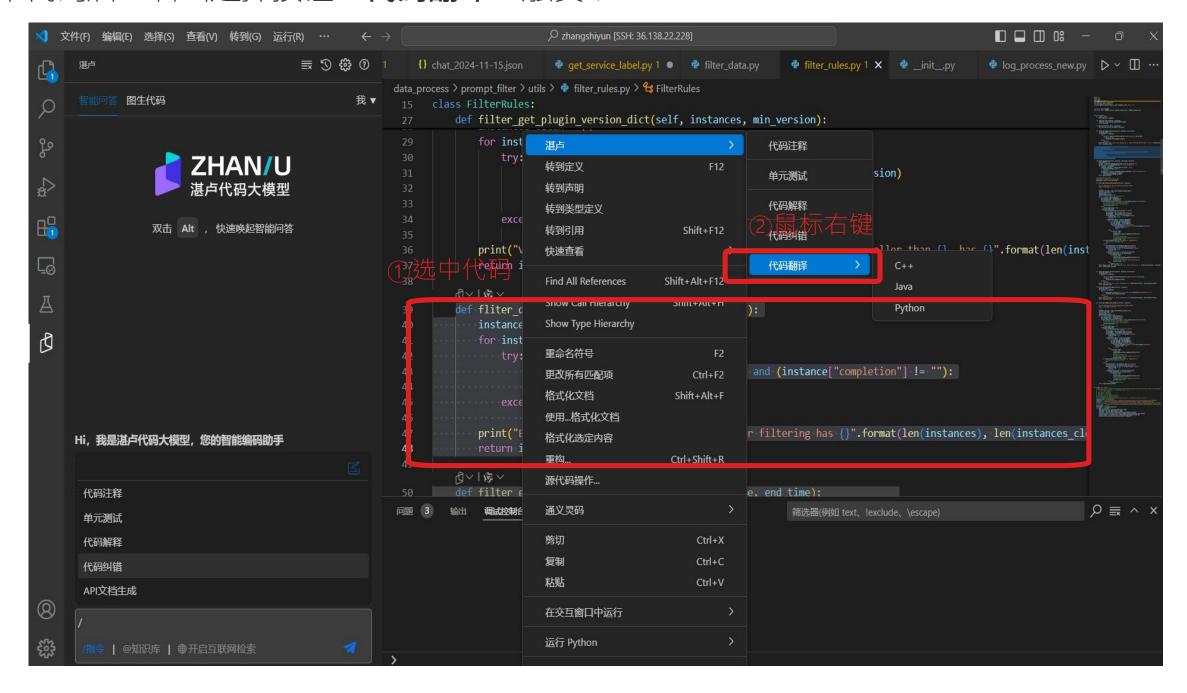


#### 3.2.7 代码翻译

触发方式支持选中代码右击选择工具触发及选中在对话框中输入命令触发:

#### ▶触发方式:右击触发

选中代码后,右击选择按钮"代码翻译"触发:



智能助手在对话框中生成对该段代码问题的翻译后的代码等,翻译后的代码片段可以进行插入及复制、生成到新文件、差异对比、点赞和踩,意见反馈,操作同上。

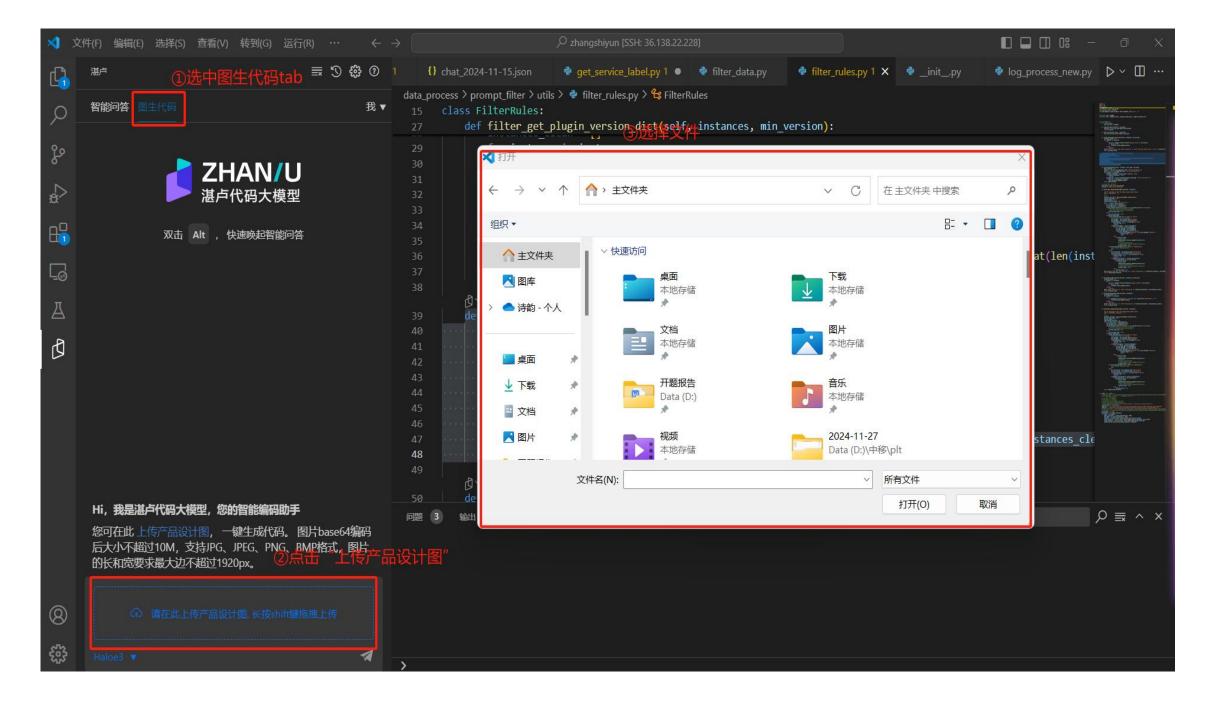
#### 3.2.8 图生代码

在左侧对话框标签页中选中"图生代码"标签,点击上传设计原型图,即可自动生成相应代码。

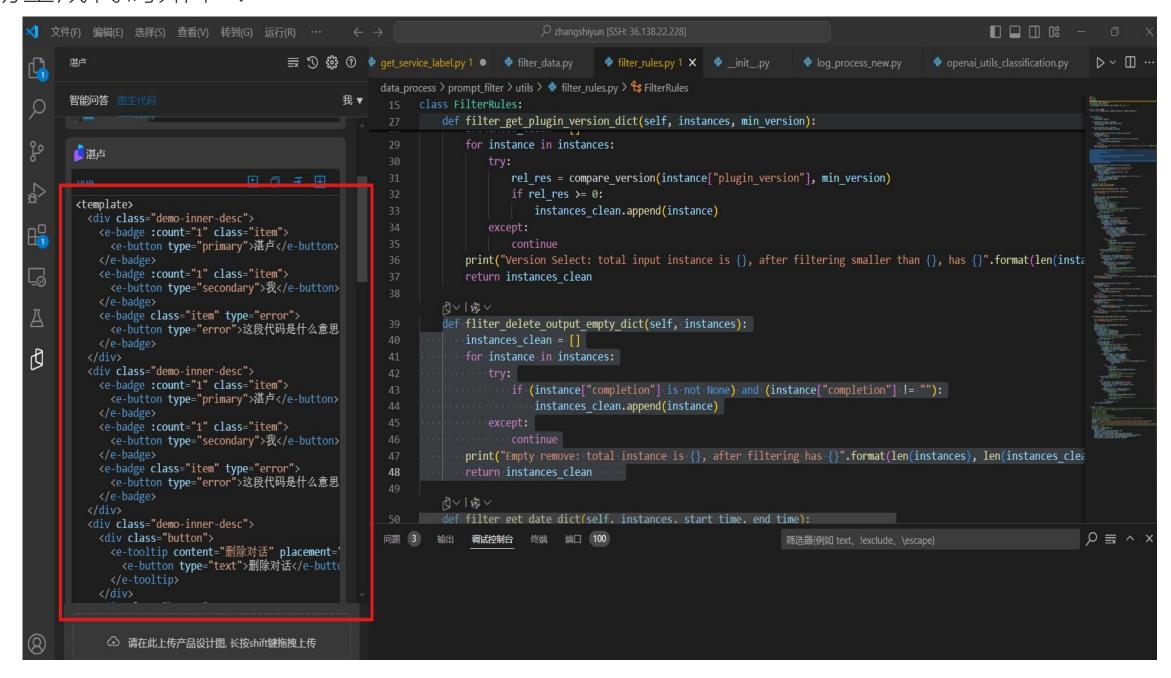
注意:图片base64编码后大小不超过10M,支持JPG、JPEG、PNG、BMP格式,图片的长和宽要求最大边不超1920px。

#### 操作方式:

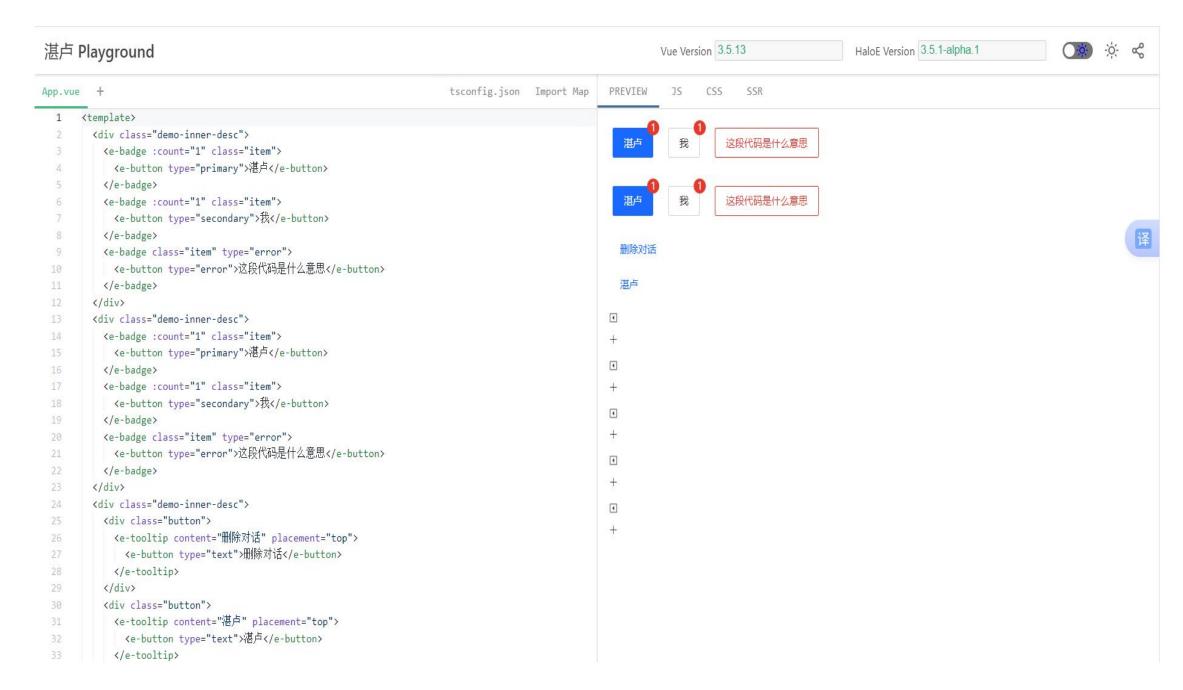
点击"图生代码"tab页,点击上传产品设计图:



#### 自动生成代码如下:



点击预览按钮 🔲,可支持预览:

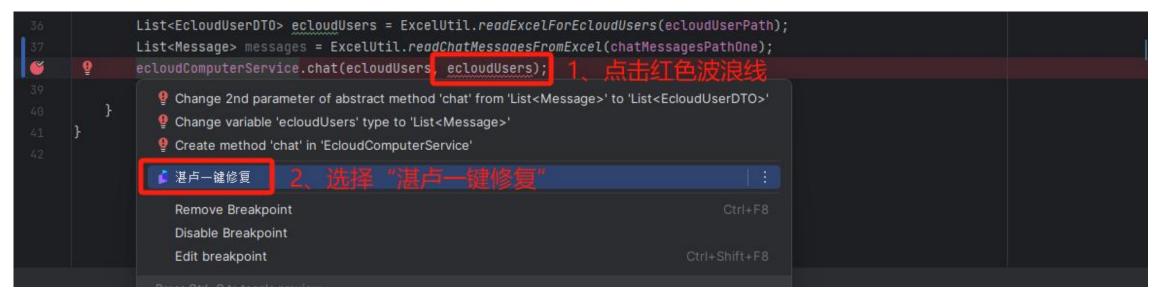


同时支持进行插入及复制、生成到新文件、差异对比、点赞和踩,意见反馈,操作同上。

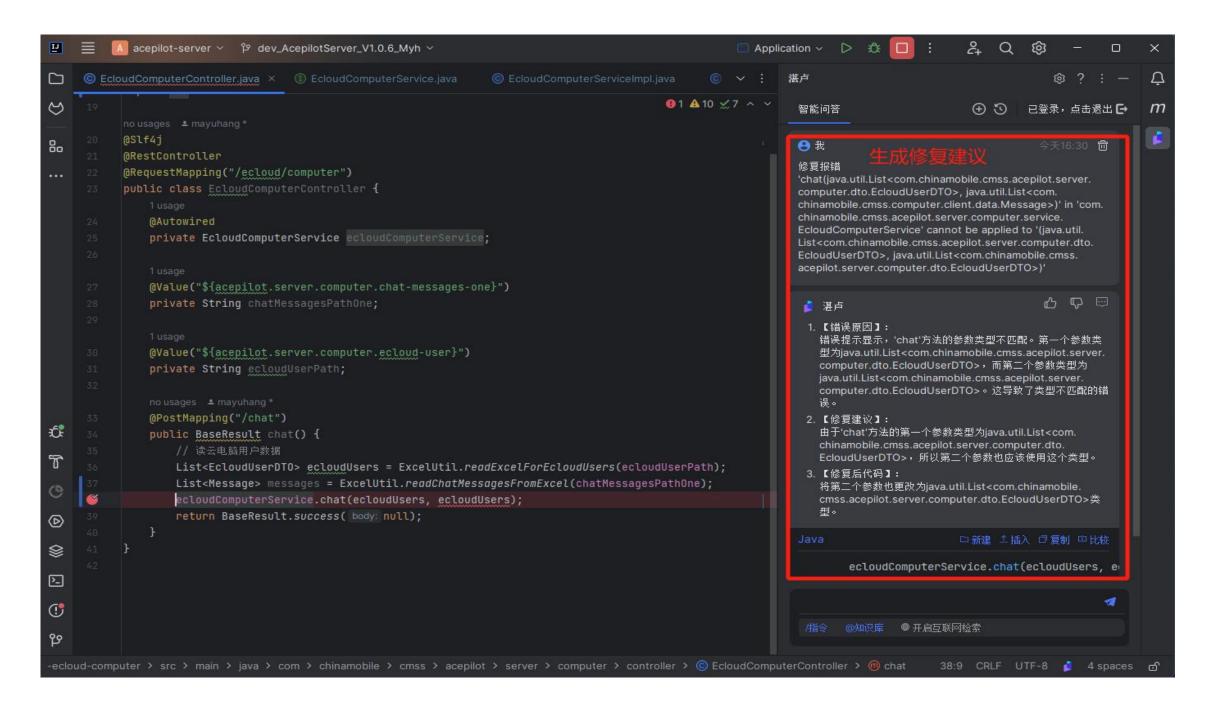
### 3.2.9 针对报错提示一键生成修复建议

当编译区出现编译错误或运行出现异常报错时,一键启动报错排查的智能答疑,可结合运行代码、异常堆栈等报错上下文,快速给出排查思路或修复建议代码。

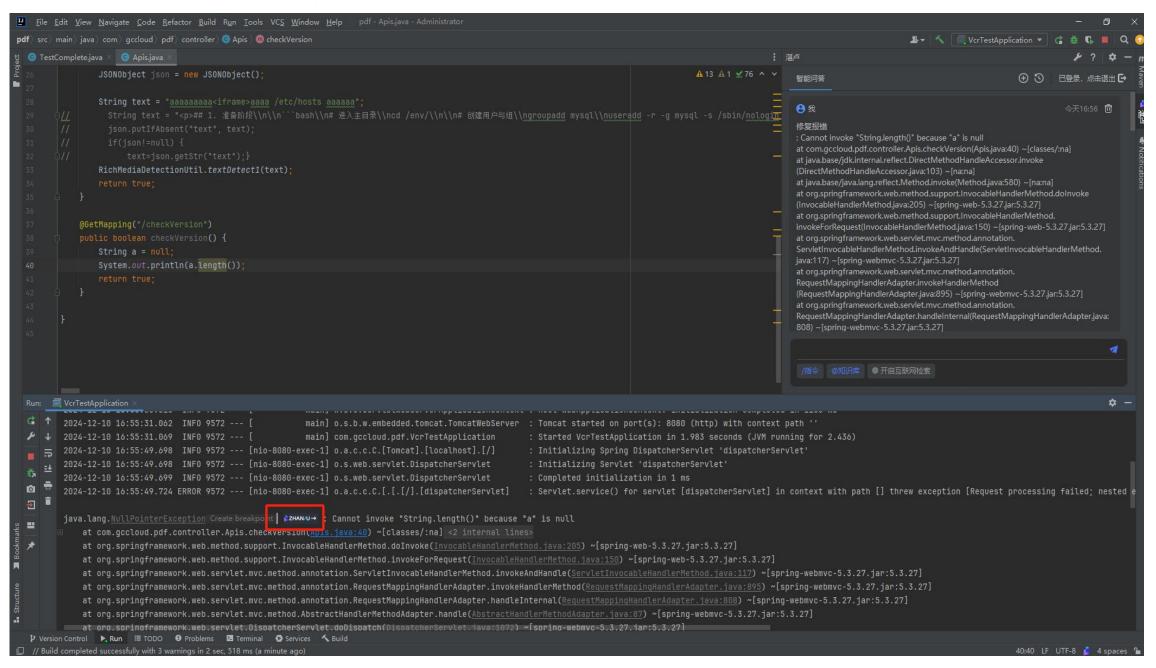
(1)针对编辑区编辑错误,用户可选择红色波浪线(编译错误处),选择湛卢一键修复功能



点击后,在问答框中生成修复建议

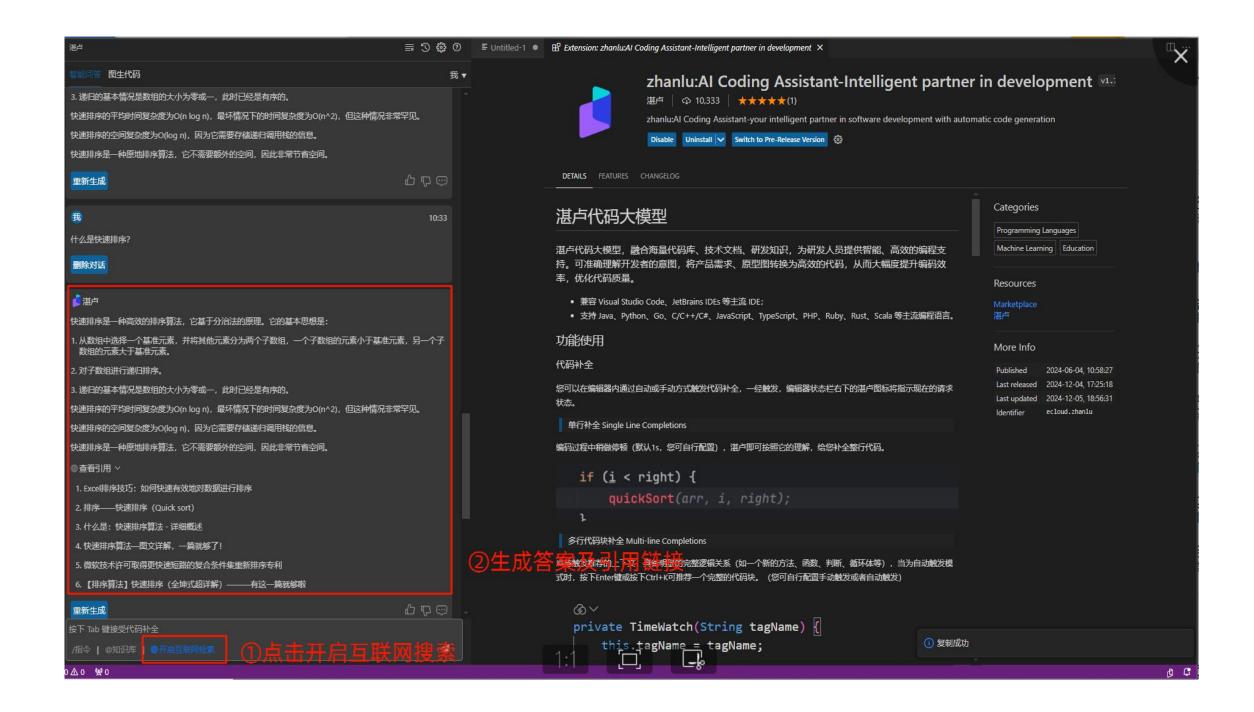


(2)针对运行报的堆栈错误,用户可点击报错堆栈处湛卢图标,一键生成修复意见



#### 3.2.10 互联网检索问答

点击对话框内"互联网问答"按钮,并输入相关关键词,可以发起互联网检索,湛卢会展示检索结果,使您不离开 IDE 就能浏览需要的信息,实现代码知识实时检索。

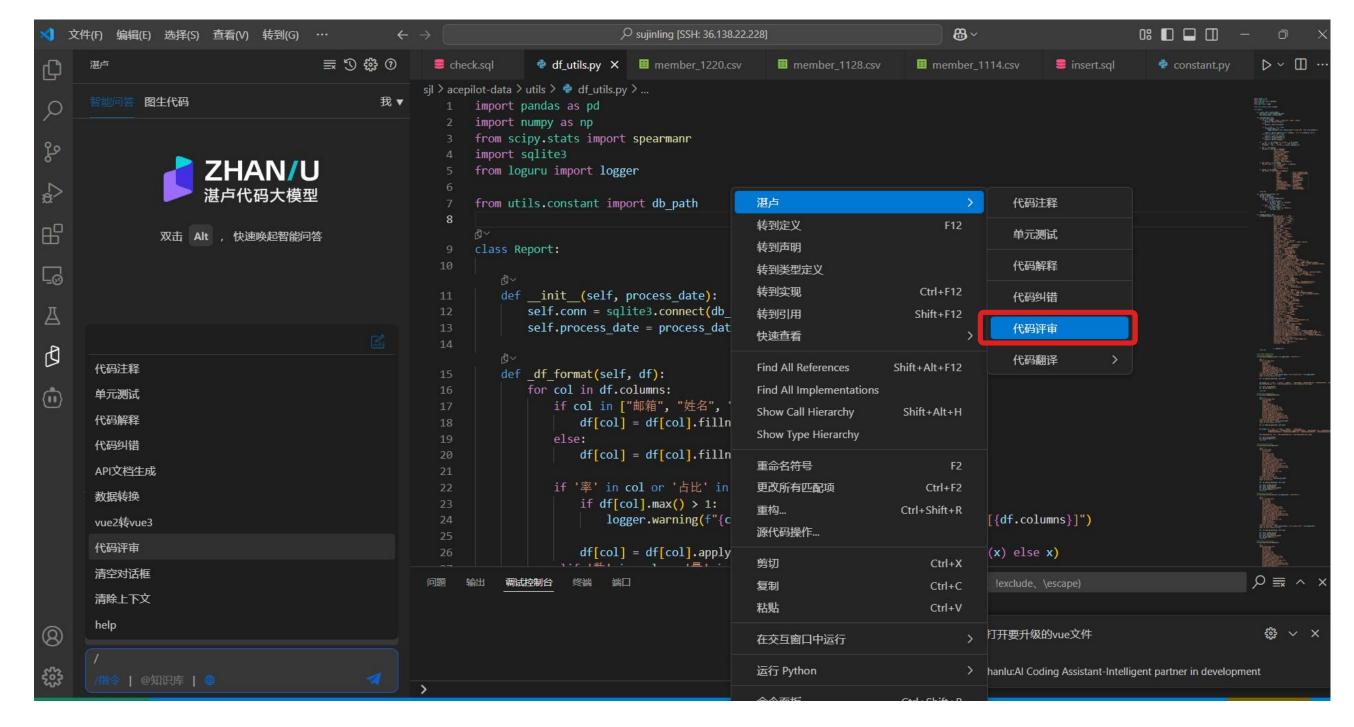


#### 3.2.11 代码评审

触发方式支持选中代码右击选择工具触发及选中在对话框中输入命令触发:

#### 触发方式1:右击触发

在要提交的代码文件中,右击选择按钮"代码评审"触发:



智能助手在对话框中对当前代码文件的增量代码生成评审意见和优化建议。

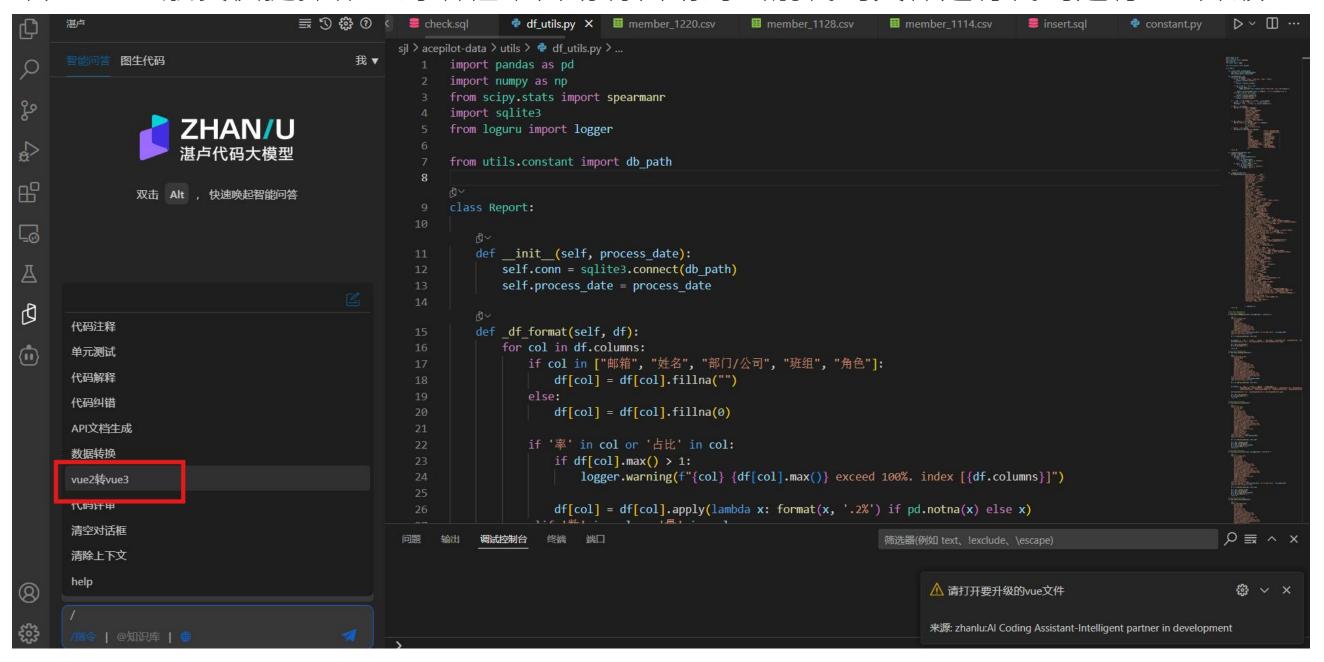
#### 触发方式2: 选中后通过命令触发:

用户在代码编辑页面,对话框中输入命令"代码评审"或点击"/代码评审"触发快捷操作,对话框中自动将自动对当前代码文件进行代码评审,生成评审建议。

#### 3.2.12 Vue升级

触发方式支持在对话框中输入命令触发:

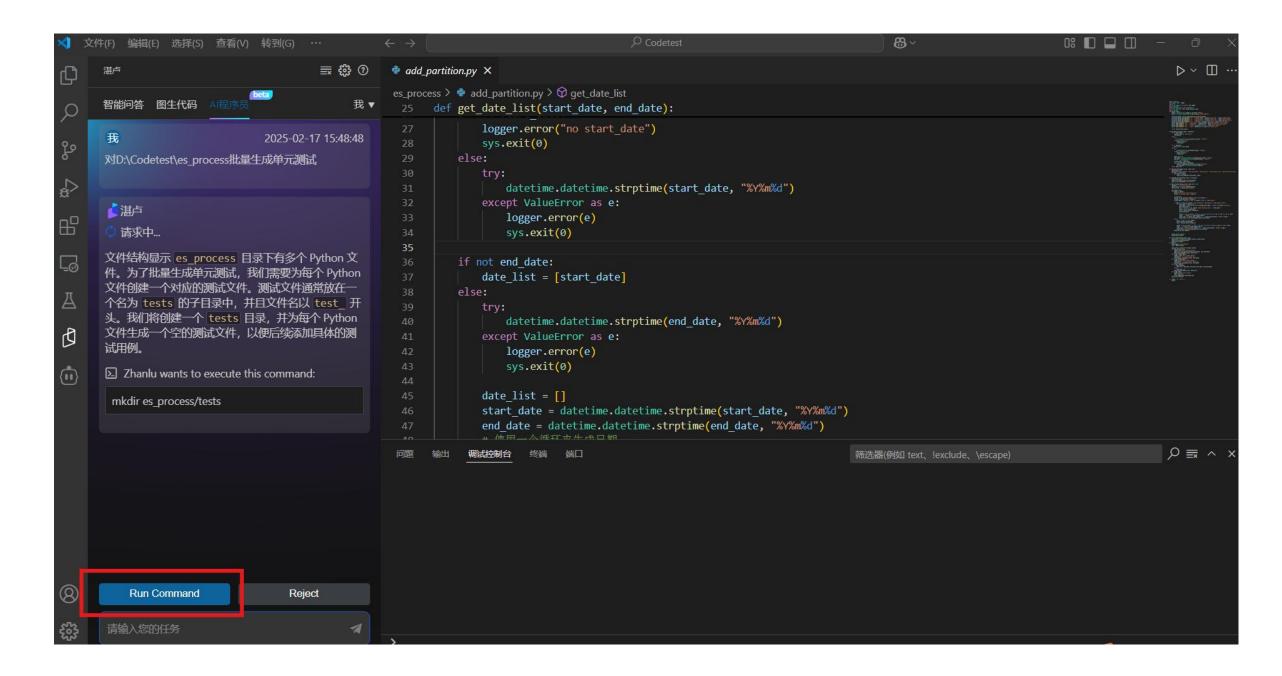
用户在要升级的Vue代码文件页面,对话框中输入命令 "vue2转vue3" 或点击 "/vue 转vue3" 触发快捷操作,对话框中自动将自动对当前代码文件进行代码进行vue升级。



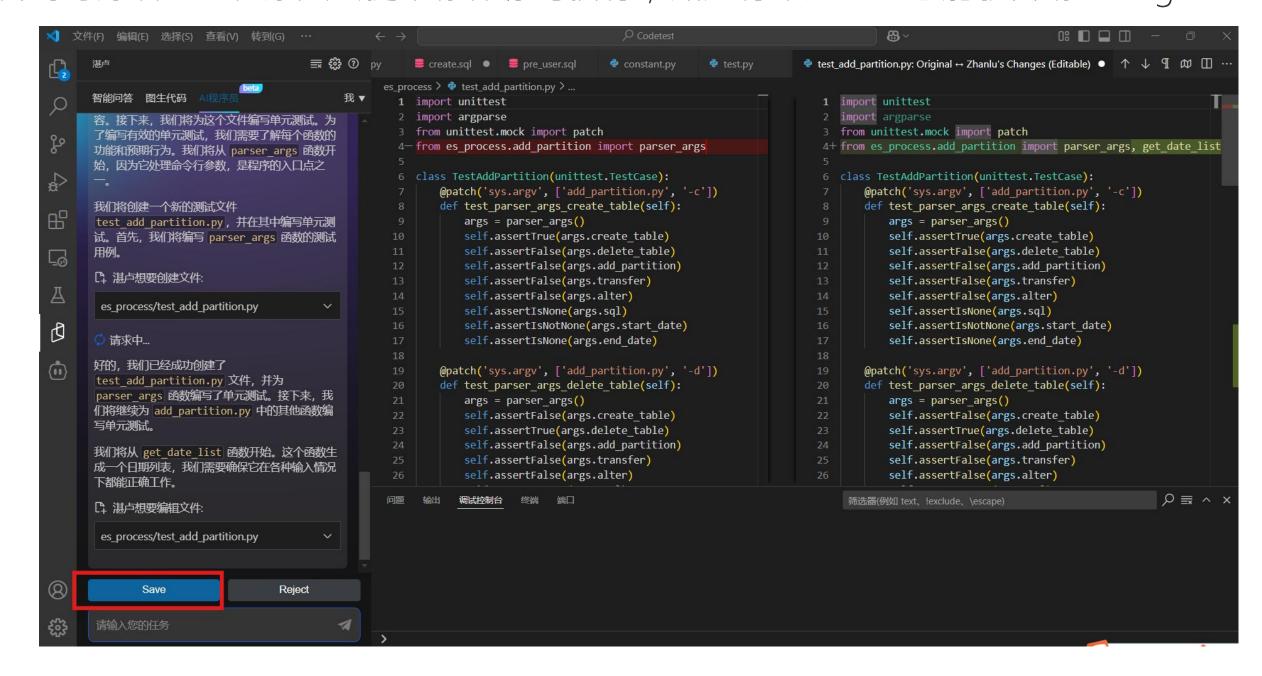
## 3.3 AI程序员

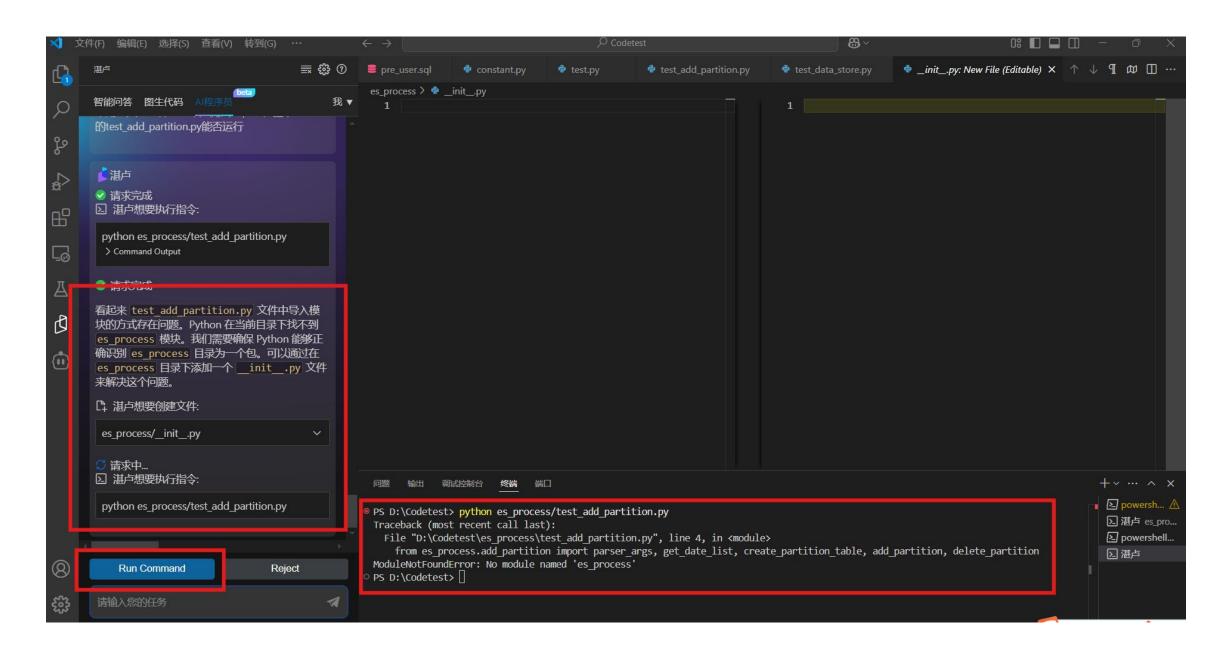
#### 3.3.1 项目级单元测试

在对话框中输入命令触发,需指定项目文件或路径,如在对话框中输入"对D:\Codetest\es\_process批量生成单元测试",AI程序员会自动对任务进行分析和拆解,点击"Run Command"表示接受并执行下一步,"Reject"表示不接受此次分析建议,点击"Reject",AI程序员会重新分析,并进行下一步。



代码写好后,AI程序员会提示你保存与执行;如运行不通过,也能够自动deBug。





除项目级单元测试外,AI程序员还支持项目级代码生成,以及项目级Vue2升级Vue3,同上只需在对话框中输入命令,如"用python写一个贪吃蛇html小游戏"、"对./XX文件中的所有Vue文件从Vue2升级到Vue3"即可。

## 3.4 常见问题解答



### ※ 湛卢支持哪些语言? Which languages does ZhanLu support?

湛卢支持C/C++、Java、Python、Go、JavaScript、TypeScript、Rust、 PHP、Kotlin、ObjectC等100+主流编程语言。

#### 是否会上传代码?上传哪些代码?

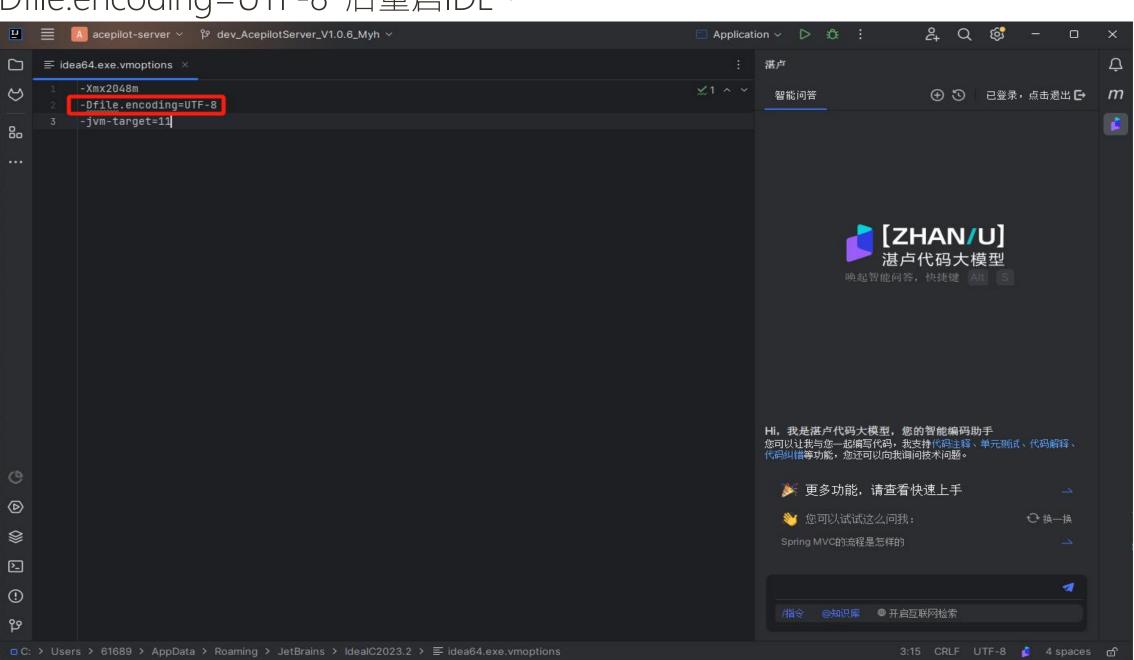
正常使用湛卢过程中需要利用正在编写的文件中的上文以及其他相关文件,来 推理出接下来的下文。因此确实会需要通过网络将编写的代码文件内容传输给 模型推理。我们承诺使用加密传输并保证不使用这部分上传的代码做任何 积累、分析、加工等后续动作。

#### **兰** 生成的内容不准是怎么回事?

大模型类产品的本身输出内容存在一定的不准确性,同时我们也建议您进行更 具体的提问,并结合代码片段,提供足够的上下文信息,优化代码结构,添加 详细的注释,确保使用最新版本的湛卢插件,保持网络稳定,从而提升问答效果。

### 生成的问答回答内容中文乱码怎么办?

增加-Dfile.encoding=UTF-8 后重启IDE。



期待大家在使用过程中,动动你们的小手,积极"点赞"或"拉踩",帮助我们更好地优化模型!

