

获取当前时间

我们在工作流的开发中会需要用到“当前时间”。

比如存入飞书表格，避免名字重复或者根据需要在画板上展示当前日期，比如拼接文件名，我们会在后面拼接一个当前时间（以python为例）：

 标题

语言 Python ▾

```
1 import datetime
2 now = datetime.datetime.now()
3 formatted_now = now.strftime("%Y-%m-%d %H:%M:%S")
4
5 async def main(args: Args) -> Output:
6     # 构建输出对象
7     ret: Output = {
8         "docTitle": f"组合后的文案({formatted_now})",
9         "tableName": formatted_now
10    }
11    return ret
```

但是如果多个工作流需要就需要重复写代码，这时候为了减少工作量，也可以选择调用获取时间的插件，在插件商店里搜索，发现有很多别人发布的时间相关的插件：

添加插件

时间

创建插件

资源库工具

应用工具

收藏

探索工具

全部

新闻阅读

便利生活

图像

实用工具

网页搜索

科学与教育

社交

排序: 最受欢迎 仅展示官方插件

没找到想要的插件? 提交反馈

Z-AI @taopl | 发布于 2024-06-27 23:37 201

3.7M | 15.6K | 69ms | 100%

get_current_datetime

获取当前的日期和时间，并以格式化的字符串形式返回。该函数获取本地的日期和时间，并将其以'YYYY-MM-DD HH:M...

添加

3.7M | 15.6K | 69ms | 100%

获取时间

通过代码获取时间

1个工具 8.6K个智能体正在使用

Jerry @JerryOnlyZRJ | 发布于 2024-07-02 17:26 51

127.4K | 8.6K | 82ms | 100%

getDateNow

获取当前时间，支持按特定格式进行格式化

format 参数

添加

127.4K | 8.6K | 82ms | 100%

时间工具

可以进行获取指定时区的时间、转换时间至指定时区、时间格式转换的操作。

3个工具 5.8K个智能体正在使用

扣子官方 | 发布于 2024-08-19 10:55 366

2.2M | 5.8K | 16ms | 98.1%

convert_format

时间格式转换。

time to_format from_format 参数 查看示例

添加

400.4K | 1.6K | 21ms | 99.8%

可能大家会觉得，既然有现成的插件可以用，为什么我还要自己去封装、重复的造轮子呢？

一是因为我们要从简单的插件入手，学习如何编写、测试、发布；二是因为不是所有插件都能直接符合我们的需求，所以我们要学习，写出符合我们自己项目需要的插件。

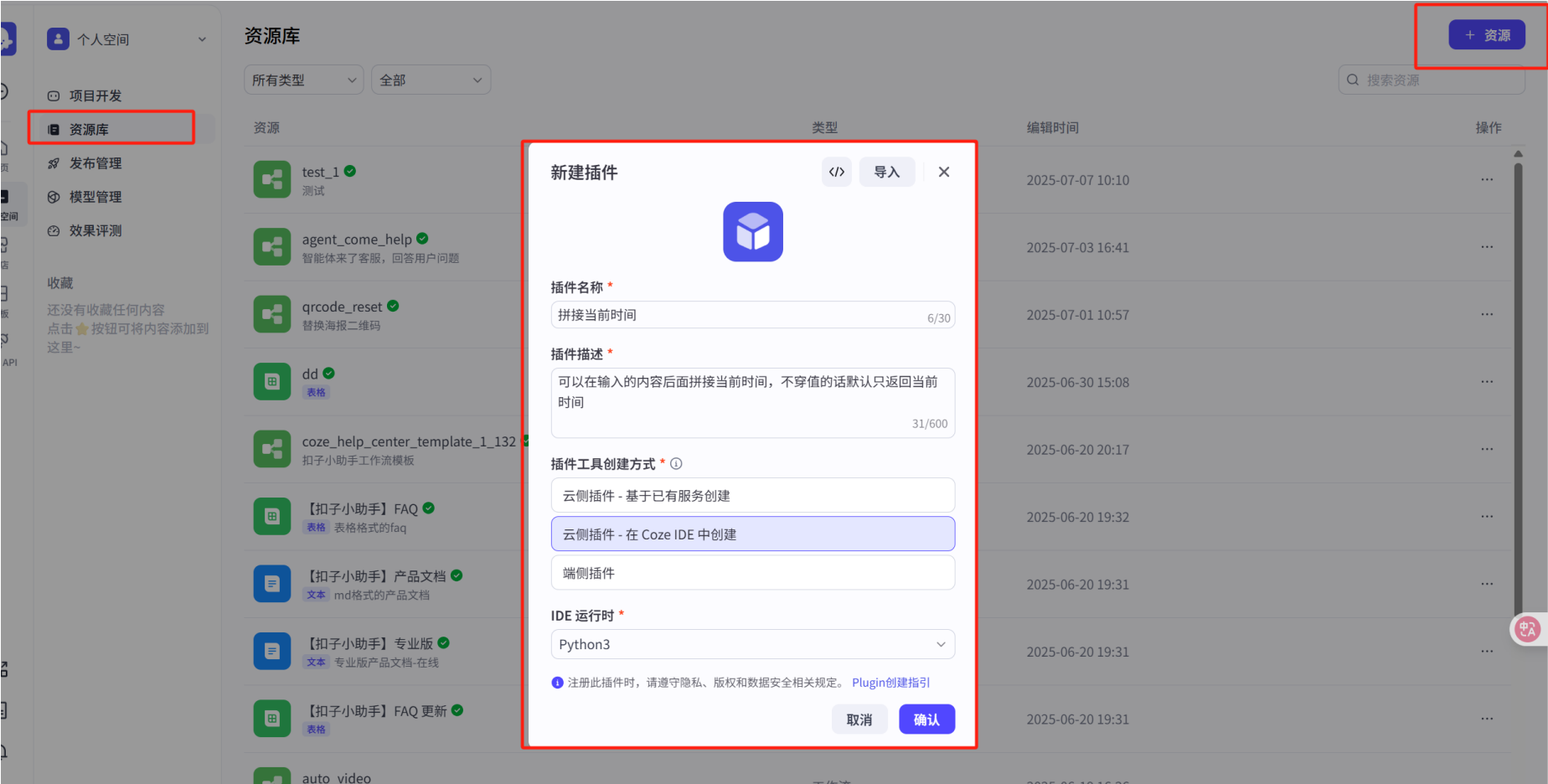
需求分析

比如说这些时间插件都是直接返回当前时间，如果我要把返回的时间拼接上文件名，那在工作流中还需要添加一个文本节点或者代码节点将数据拼接起来，那为什么不能考虑写一个插件一步到位呢？

可以的朋友！可以的！

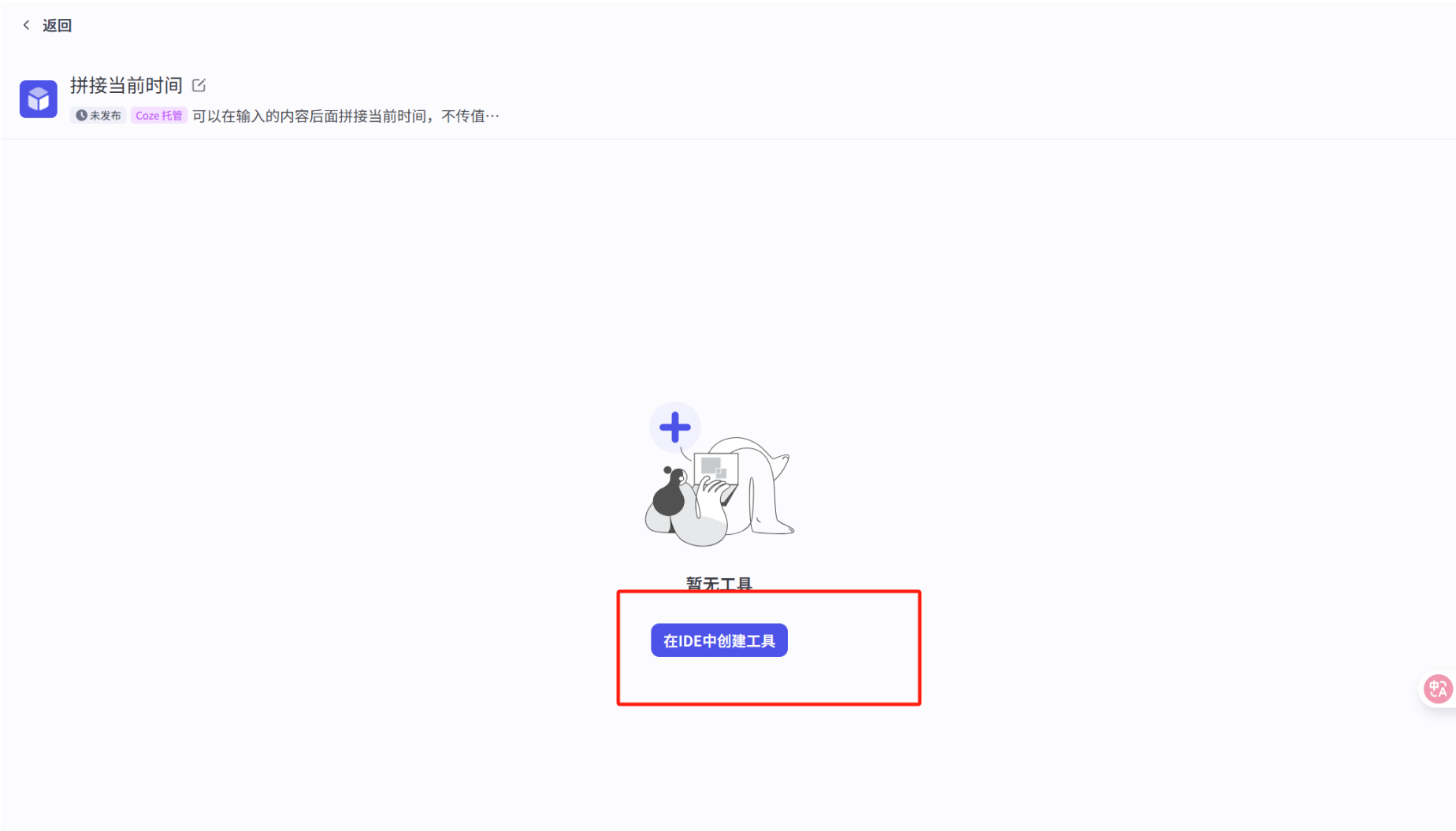
创建插件

在 资源库 中点击 +资源 > 插件 填写插件的基础信息新建一个插件，我们直接选择在coze IDE中创建，这里IDE的运行环境选择python3，

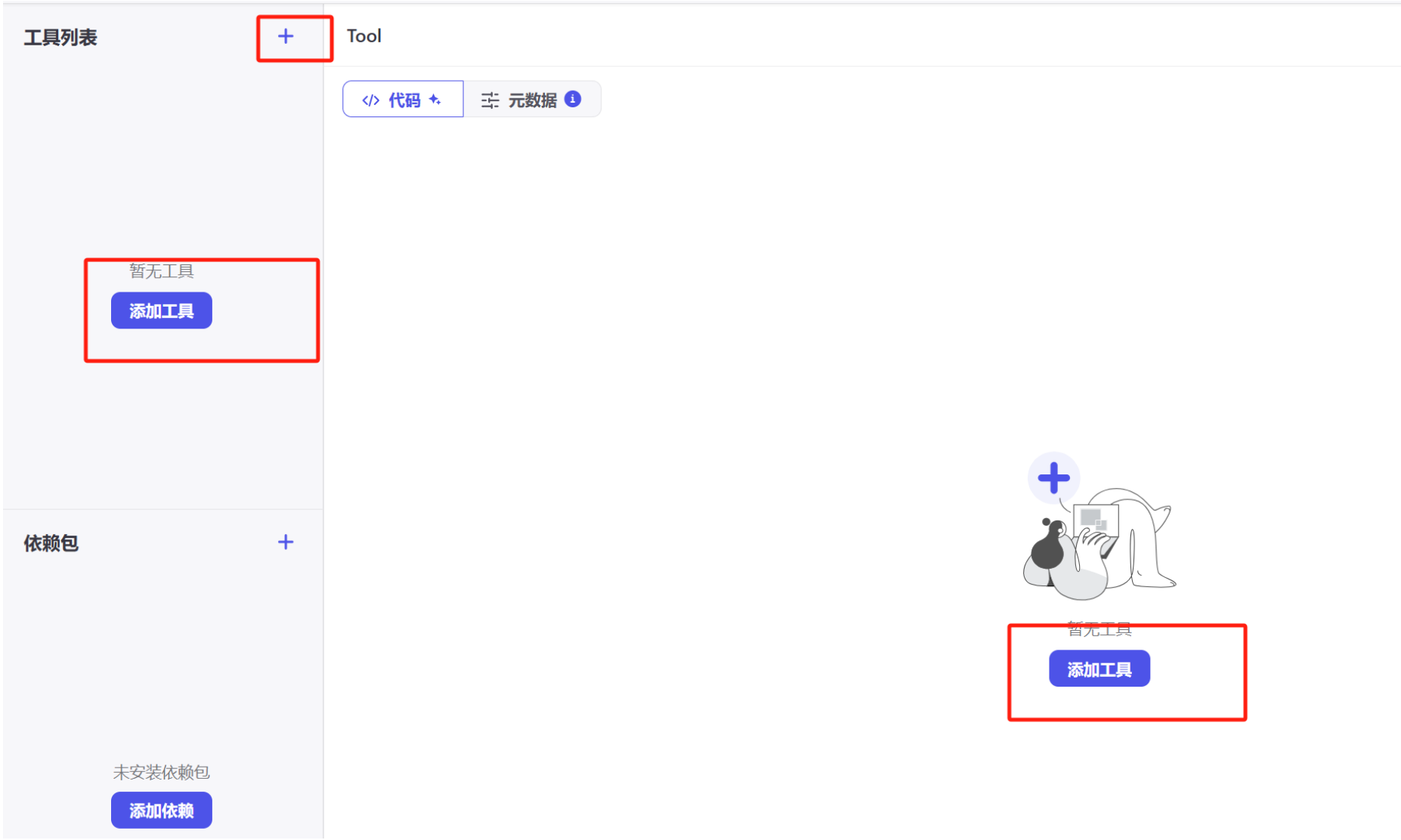


创建工具

创建工具就相当于一个接口、api，一个插件里可以创建多个工具。



以下几个地方都可以点击：



写上名字 `splice_current_time`，然后对这个工具的用途写个介绍。

创建工具

工具名称 *

splice_current_time19/30

工具介绍 *

将输入的内容拼接当前时间并返回，不传默认只返回当前时间27/600

取消

确定

创建完成之后会看到有一段默认的代码

工具列表

splice_current_time

未测试

依赖包

splice_current_time

未上线

未测试

</> 代码

元数据

```
1 from runtime import Args
2 from typings.splice_current_time.splice_current_time import Input, Output
3
4 """
5 Each file needs to export a function named `handler`. This function is the entrance to the Tool.
6
7 Parameters:
8 args: parameters of the entry function.
9 args.input - input parameters, you can get test input value by args.input.xxx.
10 args.logger - logger instance used to print logs, injected by runtime.
11
12 Remember to fill in input/output in Metadata, it helps LLM to recognize and use tool.
13
14 Return:
15 The return data of the function, which should match the declared output parameters.
16 """
17 def handler(args: Args[Input])->Output:
18     return {"message": "Hello, world!"}
```

中间的文本可以看作是一个简单的教程，它为我们解释了为什么代码的基础结构是这样的，每个参数是什么意思：

```
"""
每个文件都需要导出一个名为`handler`的函数。这个函数是工具的入口。

参数：
args: 入口函数的参数。
args.input - 输入参数，你可以通过 args.input.xxx 获取测试输入值。
args.logger - 用于打印日志的日志记录器实例，由运行时注入。

记得在元数据中填写输入/输出，这有助于LLM识别和使用工具。

返回：
函数的返回数据，应与声明的输出参数匹配。
"""
```

添加元数据

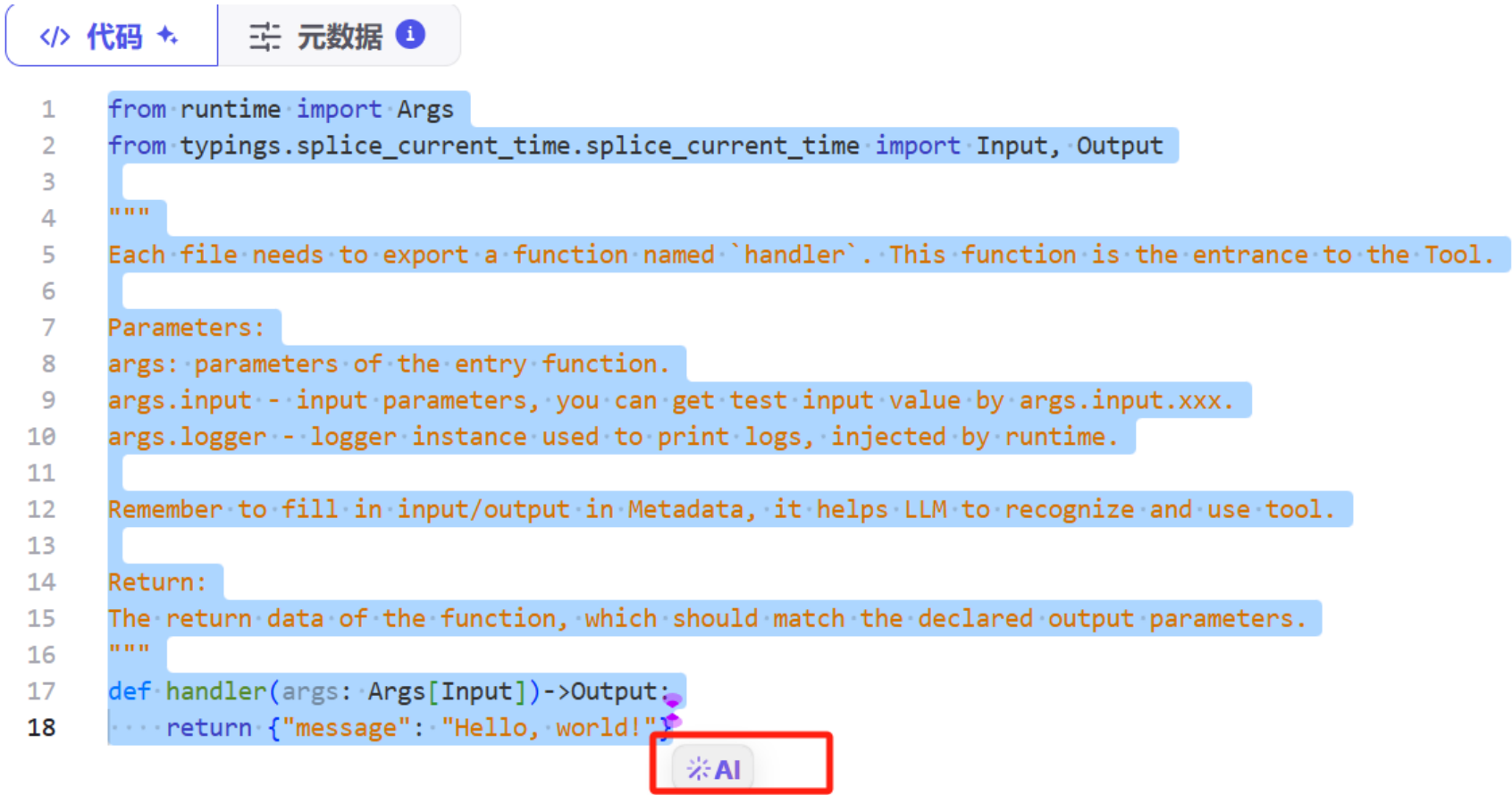
切换到元数据，分别编辑输入、输出参数，参数名的格式只支持字母、数字或下划线，编辑完记得点击保存。



用AI写代码

这个代码我们直接就让AI帮我们写就行了

第一种，我们选中代码范围，会出来一个AI的按钮：



第二种，使用ctrl+I唤起

未测试

</> 代码

≡ 元数据

AI 代码助手

✨ Use AI with

Ctrl I

```
1 import { Args } from '@runtime';
2 import { Input, Output } from '@/
  typings/new/new';
```

To write sum and minus function



ESC

Close



Code Task



History



使用快捷键 (⌘ + I) 让 AI 帮助编写或修改代码

```
17 def handler(args: Args[Input]) -> Output:
18     |
19     return {"message": "Hello, world!"}
```

```
_time.splice_current_time import I
```

function named `handler`. This fur

ry function.

rs, you can get test input value by
ce used to print logs, injected by

output in Metadata, it helps LLM to

tion, which should match the declar

AI在写的时候会参考我们测试里的测试数据，所以需要提前生成一下测试数据



这时候就可以唤起AI写代码了，我的指令是：将content拼接上当前时间并输出 ,因为我设置的输入参数名字叫做content，如果你写的别的，需要替换成对应的名字。

可以看到右侧绿色部分就是AI新增的代码，我们点击接受，那右侧绿色的部分会替换掉左侧红色的部分，如果单独点击ALT+Y表示接受替换这一块儿的代码，ALT+N就是不接受这一块儿的代码，维持原来的代码

1

from runtime import Args

2

from typings.splice_current_time.splice_current_time import

3-

4

"""

5

Each file needs to export a function named `handler`. This

6

Parameters:

7

args: parameters of the entry function.

8

args.input - input parameters, you can get test input value

9

args.logger - logger instance used to print logs, injected

10

Remember to fill in input/output in Metadata, it helps LLM

11

Return:

12

The return data of the function, which should match the dec

13

"""

14

def handler(args: Args[Input])->Output:

15

16

17

18-

1

from runtime import Args

2

from typings.splice_current_time.splice_curr

3+

from datetime import datetime

4

"""

5

Each file needs to export a function named `handler`. This

6

Parameters:

7

args: parameters of the entry function.

8

args.input - input parameters, you can get test input value

9

args.logger - logger instance used to print logs, injected

10

Remember to fill in input/output in Metadata, it helps LLM

11

Return:

12

The return data of the function, which should match the dec

13

"""

14

def handler(args: Args[Input])->Output:

15

current_time = datetime.now().strftime("%Y-%m-%d %H:%M:

16

result = f"{args.input.content} {current_time}"

17

return {"output": result}

18+

19+

20+

21+

将content拼接上当前时间并输出

请输入内容或输入 '/' 后选择指令

接受

拒绝

测试代码

点击右上角的绿色按钮打开测试，点击自动生成生成测试的数据，也可以自己替换，最后点击运行，看一下代码是否能运行成功

splice_current_time 未上线 未测试

</> 代码

元数据

1

from runtime import Args

2

from typings.splice_current_time.splice_current_time import Input, Output

3

from datetime import datetime

4

"""

5

Each file needs to export a function named `handler`. This function is the entrar

6

Parameters:

7

args: parameters of the entry function.

8

args.input - input parameters, you can get test input value by args.input.xxx.

9

args.logger - logger instance used to print logs, injected by runtime.

10

Remember to fill in input/output in Metadata, it helps LLM to recognize and use t

11

Return:

12

The return data of the function, which should match the declared output parameter

13

"""

14

def handler(args: Args[Input])->Output:

15

current_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

16

result = f"{args.input.content} {current_time}"

17

return {"output": result}

18

19

20

21

测试代码

自动生成

2

输入

{

"content": "wyivqmdb"

}

运行

3

输出

看一下运行结果

splice_current_time 未上线 测试通过

</> 代码 元数据

```
1 from runtime import Args
2 from typings.splice_current_time.splice_current_time import Input, Output
3 from datetime import datetime
4 """
5 Each file needs to export a function named `handler`. This function is the entrance
6
7 Parameters:
8 args: parameters of the entry function.
9 args.input - input parameters, you can get test input value by args.input.xxx.
10 args.logger - logger instance used to print logs, injected by runtime.
11
12 Remember to fill in input/output in Metadata, it helps LLM to recognize and use
13
14 Return:
15 The return data of the function, which should match the declared output parameter
16 """
17 def handler(args: Args[Input]) -> Output:
18     current_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
19     result = f"{args.input.content} {current_time}"
20     return {"output": result}
21
```

控制台 测试

测试代码

输入

{ "content": "智能体来了" }

运行

输出

{ "output": "智能体来了 2025-07-07 17:03:44" }

{args.input.content} 这个就是输入的content的值

{current_time} 是当前时间，时间的格式我们是可以通过strftime里传参控制的 %Y-%m-%d %H:%M:%S 中Y取的Year的首字母，代表年，后面依次是月-日 时:分:秒

```
def handler(args: Args[Input]) -> Output:
    current_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    result = f"{args.input.content} {current_time}"
    return {"output": result}
```

运行

输出

{ "output": "智能体来了 2025-07-07 17:03:44" }

如果我这里的最终的展示格式我想呈现为：输入的内容-年/月/日 X时X分，那么我可以修改代码为：

```
1 from runtime import Args
2 from typings.splice_current_time.splice_current_time import Input, Output
3 from datetime import datetime
4 """
5 Each file needs to export a function named `handler`. This function is the entrance
6
7 Parameters:
8 args: parameters of the entry function.
9 args.input - input parameters, you can get test input value by args.input.xxx.
10 args.logger - logger instance used to print logs, injected by runtime.
11
12 Remember to fill in input/output in Metadata, it helps LLM to recognize and use
13
14 Return:
15 The return data of the function, which should match the declared output parameter
16 """
17 def handler(args: Args[Input]) -> Output:
18     current_time = datetime.now().strftime("%Y/%m/%d %H时%M分")
19     result = f"{args.input.content}-{current_time}"
20     return {"output": result}
```

```
{
  "content": "智能体来了"
}
```

运行

输出

```
{
  "output": "智能体来了-2025/07/07 17时13分"
}
```

优化代码

但现在有个问题，拼接的效果实现了，如果我不传值的情况我预期是只输出时间的，我们测试一下：

```
</> 代码 元数据
1 from runtime import Args
2 from typings.splice_current_time.splice_current_time import Input, Output
3 from datetime import datetime
4 """
5 Each file needs to export a function named `handler`. This function is the entrance
6
7 Parameters:
8 args: parameters of the entry function.
9 args.input - input parameters, you can get test input value by args.input.xxx.
10 args.logger - logger instance used to print logs, injected by runtime.
11
12 Remember to fill in input/output in Metadata, it helps LLM to recognize and use
13
14 Return:
15 The return data of the function, which should match the declared output parameter
16 """
17 def handler(args: Args[Input]) -> Output:
18     current_time = datetime.now().strftime("%Y/%m/%d %H时%M分")
19     result = f"{args.input.content}-{current_time}"
20     return {"output": result}
```

```
{
  "content": ""
}
```

运行

输出

```
{
  "output": "-2025/07/07 17时14分"
}
```

从输出结果可以看出来，输出的数据并不符合我们的要求，多了一个“-”，这个作为连接符被我们写死在了代码里，那怎么处理呢？

优化1.0

可以在这里添加一个逻辑，如果content不为空就输出当前这个拼接的result，如果为空就直接输出当前时间，那么我可以告诉AI：

如果content不为空输出拼接后的值，如果为空直接输出当前时间



ESC 关闭 | / AI 指令 | ↑↓ 历史对话



```
17 def handler(args: Args[Input]) -> Output:
18     ... current_time = datetime.now().strftime("%Y/%m/%d %H时%M分")
19     ... result = f"{args.input.content}-{current_time}"
20     ... return {"output": result}
21
```

那么AI运行后给了我们解决方案，可以看到代码里使用了if-else来判断，

15 The return data of the function, which should match the declared output parameters.

如果content不为空输出拼接后的值，如果为空直接输出当前时间

请输入内容或输入 '/' 后选择指令



接受

拒绝



16 """

17 def handler(args: Args[Input]) -> Output:

18 current_time = datetime.now().strftime("%Y/%m/%d %H时%M分")

19 result = f"{args.input.content}-{current_time}"

20 return {"output": result}

17 def handler(args: Args[Input]) -> Output:

18 current_time = datetime.now().strftime("%Y/%m/%d %H时%M分")

19+ if args.input.content:

20+ result = f"{args.input.content}-{current_time}"

21+ return {"output": result}

22+ else:

23+ return {"output": current_time}

再来运行一下，就正常了：

4 """

5 Each file needs to export a function named `handler`. This function is the

6

7 Parameters:

8 args: parameters of the entry function.

9 args.input - input parameters, you can get test input value by args.input

10 args.logger - logger instance used to print logs, injected by runtime.

11

12 Remember to fill in input/output in Metadata, it helps LLM to recognize a

13

14 Return:

15 The return data of the function, which should match the declared output

16 """

17 def handler(args: Args[Input]) -> Output:

18 current_time = datetime.now().strftime("%Y/%m/%d %H时%M分")

19 if args.input.content:

20 result = f"{args.input.content}-{current_time}"

21 return {"output": result}

22 else:

23 return {"output": current_time}

24

```
{
  "content": ""
}
```

运行

输出

```
{
  "output": "2025/07/07 17时28分"
}
```

优化2.0

但是又有新的问题出来了，现在的代码意味着，我在输入内容拼接时间的时候，中间永远有个 - ，那可能有时候我不想要这个 - ，我想换成一个空格或者一个冒号 : ，甚至换成文字 生成于 ，这都是不固定的，那就有两种解决方案：

- 1. 方案一：把现在代码里的 - 删掉，想用什么作为中间的拼接桥梁就和content一起传入进来，比如： 智能体来了 、 智能体来了: 、 智能体来了生成于 。但是这样做有个问题，比如用户输入的是 ``智能体来了 我还是得添加一个节点把中间的这个 桥梁` 给拼接上再传给创建，这就又麻烦了一步。
- 2. 方案二：新增一个输入参数作为中间的桥梁，连接两边的内容，传什么就是什么，更加灵活。

那这里我们再在元数据里添加一个输入参数

后开

取消

保存

输入参数

名称*	描述*	类型*	必填	删除	新增子项
content	需要拼接时间的内容	String	<input checked="" type="checkbox"/>		
middle_str	展示在content和当前时间中间	String	<input checked="" type="checkbox"/>		

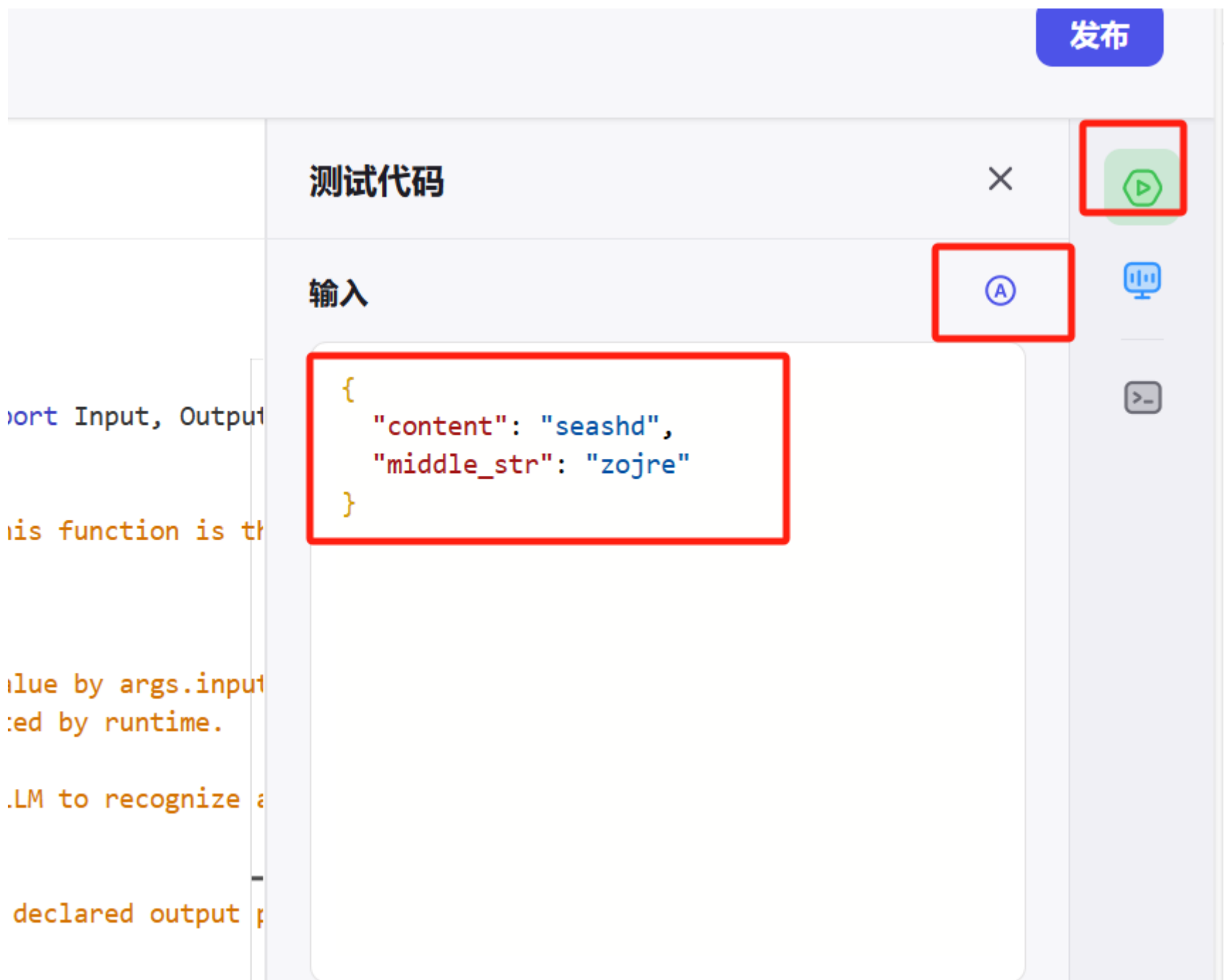
+ 新增参数

输出参数

编辑

名称*	描述	类型*	必填	删除	新增子项
-----	----	-----	----	----	------

然后回到我们的代码界面，先更新一下测试数据



然后告诉AI我们的需求 无论content是否为空，直接输出content拼接middle_str拼接当前时间的值，这个IDE自带的AI生成的代码不是一次性就符合要求的，有时候可能需要多尝试几次

无论content是否为空，直接输出content拼接middle_str拼接当前时间的值

请输入内容或输入 '/' 后选择指令

接受

拒绝



输出代码不满意可以点击重新生成

```
1  n runtime import Args
2  n typings.splice_current_time.splice_current_time import Input, Output
3  n datetime import datetime
4
5  n file needs to export a function named `handler`. This function is the entrance to the Tool.
6
7  ameters:
8  s: parameters of the entry function.
9  s.input - input parameters, you can get test input value by args.input.xxx.
10 s.logger - logger instance used to print logs, injected by runtime.
11
12 ember to fill in input/output in Metadata, it helps LLM to recognize and use tool.
13
14 urn:
15 return data of the function, which should match the declared output parameters.
16
17 :f handler(args: Args[Input])->Output:
18     current_time = datetime.now().strftime("%Y/%m/%d %H:%M:%S")
19     if args.input.content:
20         result = f"{args.input.content}-{current_time}"
21         return {"output": result}
22     else:
23         return {"output": current_time}
```

```
17 :f handler(args: Args[Input])->Output:
18     current_time = datetime.now().strftime("%Y/%m/%d %H:%M:%S")
19     result = f"{args.input.content}{args.input.logger}"
20     return {"output": result}
```

因为原代码里有if-else，所以重新生成的时候可能会出现还是有if-else的情况，但是我不希望出现，提示词里也写了，所以重新生成几次，测试一下输出效果。

</> 代码 +

元数据 ⓘ

```
1 from runtime import Args
2 from typings.splice_current_time.splice_current_time import Input, Output
3 from datetime import datetime
4 """
5 Each file needs to export a function named `handler`. This function is t
6
7 Parameters:
8 args: parameters of the entry function.
9 args.input - input parameters, you can get test input value by args.inpu
10 args.logger - logger instance used to print logs, injected by runtime.
11
12 Remember to fill in input/output in Metadata, it helps LLM to recognize
13
14 Return:
15 The return data of the function, which should match the declared output
16 """
17 def handler(args: Args[Input])->Output:
18     current_time = datetime.now().strftime("%Y/%m/%d %H时%M分")
19     result = f"{args.input.content}{args.input.middle_str}{current_time}"
20     return {"output": result}
21
22
```

控制台 测试 v

输入

{
"content": "智能体来了",
"middle_str": "我是拼接词"
}

运行

输出

output:"智能体来了我是拼接词2025/07/07 18时39分"

发布

测试通过后结果没问题就可以发布了

对象存储 -... Prompt Advance | P... 智能体来了 - 专注AI... YrEdu后台管理 Home · Streamlit LangChain MasterCl... LeiaPix 定制个人3D头像.pdf Comfy V

的内容后面拼接当前时间，不传值的话默认只返回当前时间

发布

splice_current_time 未上线 测试通过

</> 代码 +

元数据 ⓘ

```
1 from runtime import Args
2 from typings.splice_current_time.splice_current_time import Input, Output
3 from datetime import datetime
4 """
5 Each file needs to export a function named `handler`. This function is t
6
7 Parameters:
8 args: parameters of the entry function.
9 args.input - input parameters, you can get test input value by args.inpu
10 args.logger - logger instance used to print logs, injected by runtime.
11
12 Remember to fill in input/output in Metadata, it helps LLM to recognize
13
14 Return:
15 The return data of the function, which should match the declared output
16 """
17 def handler(args: Args[Input])->Output:
18     current_time = datetime.now().strftime("%Y/%m/%d %H时%M分")
19     result = f"{args.input.content}{args.input.middle_str}{current_time}"
20     return {"output": result}
21
22
```

测试代码

输入

{
"content": "智能体来了",
"middle_str": "我是拼接词"
}

运行

输出

{
 output:"智能体来了我是拼接词2025/07/07 18时39分"



发布成功了



发布后如果是自己使用的话，可以在自己的资源库工具中添加使用




如果想让别人也搜索到，能够使用，可以上架到插件商店



关于插件的内容会根据你创建插件时写的描述以及对输出参数的描述自动生成：

插件信息

正在使用默认图标，更换一个独特的图标
可以有效提升插件的关注度！
立即更换



拼接当前时间

guokexin @AgentCome_kexin

可以在输入的内容后面拼接当前时间，不传值的话默认只返回当前时间

1 工具

splice_current_time

将输入的内容拼接当前时间并返回，不传默认只返回…

关于插件 *

插件功能：“拼接当前时间”插件可在输入内容的后面拼接当前时间。若不传入内容，该插件默认只返回当前时间。

工具介绍：

1. 工具名称：splice_current_time

- 功能介绍：该工具可将输入的内容拼接当前时间并返回，若不传入内容，默认只返回当前时间。输入参数有两个，“content”为需要拼接时间的内容，“middle_str”展示在“content”和当前时间中间。
- 使用场景：
 - 当你需要对一些文本、记录添加时间标签时，可将文本作为“content”传入，利用该工具快速拼接时间。例如在记录日志、笔记时，方便了解内容的生成时间。
 - 若想强调时间与内容的分隔效果，可传入“middle_str”，比如使用“-”作为分隔符，使拼接后的内容看起来更清晰。
 - 若只是单纯想获取当前时间，不传入任何参数，即可得到当前时间。

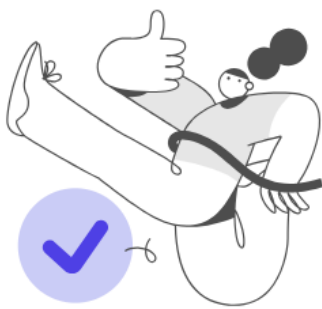
368/5000

分类 *

实用工具

然后提交，等待审核

插件信息



审核中

审核通过后将在插件商店展示

查看更多

上架插件注意事项

上架插件的时候记得把插件的默认题标换掉，不然会审核不通过，最新的就是更改了图标后重新发布上架的



扣子团队 刚刚

Hi, guokexin, 你的插件商品“拼接当前时间”审核已通过，可点击链接查看详情页：

<https://coze.cn/store/plugin/7524289669056495670>。



扣子团队 13 分钟前

Hi, guokexin, 你的插件商品“拼接当前时间”因不符合[扣子发布规范](#)，未通过审核。审核不通过原因为“插件素材信息（icon、名称、描述）”包含违规内容，请修改后重新提交。

拓展

我们的插件还有优化的地方，现在我们的时间格式是固定死的，那是不是可以再新增一个输入参数，通过输入 `%Y-%m-%d %H:%M:%S`，用来控制时间的格式呢？

大家可以自己优化一下代码哦~