

前端: [http://10.226.28.53/interfaceTest/HTTP\\_InterfaceCheck](http://10.226.28.53/interfaceTest/HTTP_InterfaceCheck)

后端: <http://10.226.28.53/myadmin/exePython/check>



使用手册.pdf

4.9MB

问题一：python模式报错

快速筛选

请选择环境

我的历史

小说快应用

查看全部

小说快应用

小说管理后台

编号	测试结果	执行环境	实际结果	断言结果	执行前耗时	执行后耗时
1	ERROR	小说快应用	<ERROR:发生异常: Traceback (most recent call last): File "F:\soso\test\AutotestEn	ERROR: 处理[准备]阶段时出现错误: ERROR: <ERROR:发生异常: Traceback (most recent call last): File "F:\soso\test\AutotestEn	5	

我的历史

小说快应用

查看全部

小说快应用


小说管理后台

编号	测试结果	执行环境	实际结果	断言结果	执行前耗时	执行后耗时
1	ERROR	小说快应用	<ERROR:发生异常: Traceback (most recent call last): File "F:\soso\test\AutotestEn	The settings file 'F:\soso\test\AutotestEn\settings.py' does not exist. ModuleNotFoundError: No module named 'faker'	5	0

在虚拟环境中安装faker

pip install faker -i http://pypi.douban.com/simple/ --trusted-host pypi.douban.com

问题二：安装jenkins插件

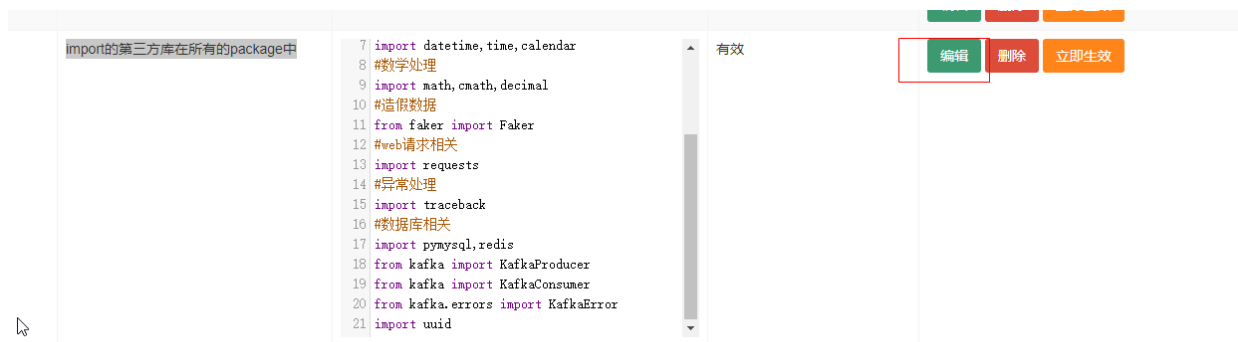


sosotest.hpi

1.73MB

问题三：导入的包在系统不存在时

在后台后台配置，平台配置管理→执行python代码管理，选择import的第三方库在所有的package中点击【编辑】



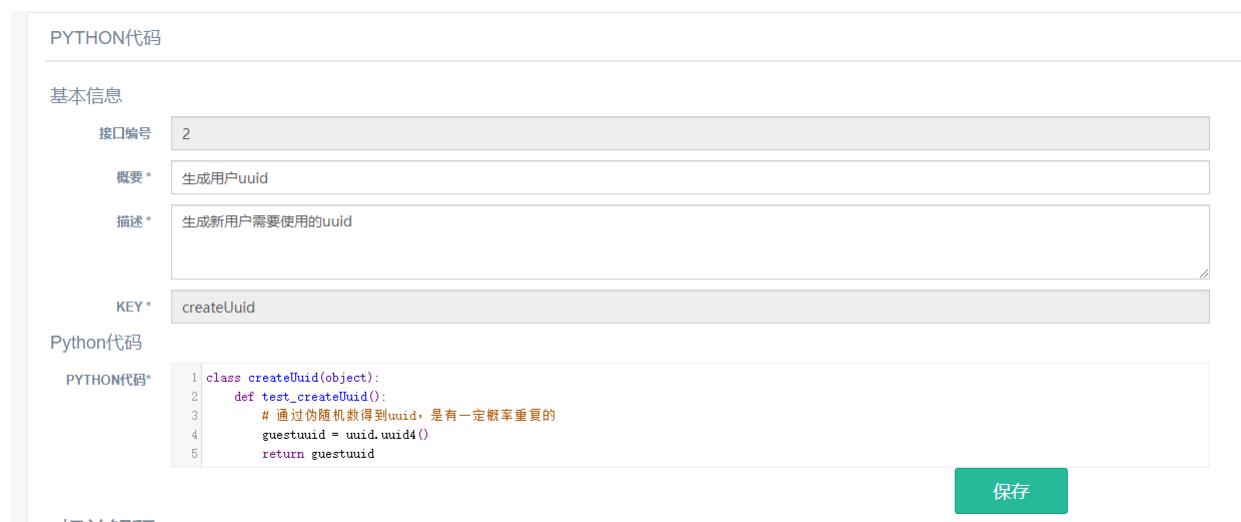
写入需要导入的包如import uuid 完成后点击【立即生效】

## 问题四：对于需要提前调用函数的接口

步骤一：数据服务下选择KEYWORD/PYTHON模式，点击【新建PYTHON模式】



写法如下：



```
class createUuid(object):
```

```
    @staticmethod
```

```
    def test_createUuid():
```

```
        # 通过伪随机数得到uuid, 是有一定概率重复的
```

```
guestuuid = uuid.uuid4()
return guestuuid
```

步骤二：在需要调用函数的接口的准备调用即可

准备

```
1 # python
2 DEBUG_MODE = True # 当执行完后，执行信息中会展示执行的python代码，执行后的作用域中的变量、类、函数等信息。
3 imports("createUuid") # 引入自定义python代码，参数是key
4 Uuid = createUuid.test_createUuid() # 调用类中的静态函数
5 log('生成的uuid是: %s' %Uuid)
6
imports("createUuid")
```

# python

DEBUG\_MODE = True # 当执行完后，执行信息中会展示执行的python代码，执行后的作用域中的变量、类、函数等信息。

imports("createUuid") # 引入自定义python代码，参数是key

Uuid = createUuid.test\_createUuid() # 调用类中的静态函数

log('生成的uuid是: %s' %Uuid)

## 问题五：接口关联如何做

1. 在接口一的断言恢复中写入 如下：

断言恢复

```
1 ASSERT("result":"ok");
2 userID = JSON_GET($CONST[RESP_TEXT],["data"]["id"]);
```

userID = JSON\_GET(\$CONST[RESP\_TEXT],["data"]["id"]);

python模式下的关联

# python

resultDict = json.loads(const("RESP\_TEXT"))

recordID = resultDict["data"]["items"][0]["id"]

2. 在接口二引用即可

PARAMS

HEADER(0)

Key	Value
userId	{{userId}}

{{userId}}

问题五：如何做数据驱动

1. 在接口准备写入python模式代码



# python

execute\_interface('HTTP\_INTERFACE\_1')

DEBUG\_MODE = True # 当执行完后，执行信息中会展示执行的python代码，执行后的作用域中的变量、类、函数等信息。

billTypelist = ["1","2"]

totalcount, passcount, failcount = 0,0,0

for billType in billTypelist:

execute\_current() # 执行当前接口的执行信息，使用变量user

# 输出执行后的输入和输出信息

log("请求参数是: %s" % context.processedParams)

备注：execute\_interface('HTTP\_INTERFACE\_1') 代表的是先执行登录接口拿到token

2. 参数中引用即可



{{billType}}

备注：

在接口调试的【查看详情】中的准备会打印我们的python日志

准备

```
billTypelist = ["1", "2"] ;(来源: [HTTP_INTERFACE_10:["1", "2"]])
totalcount = 0 ;(来源: [HTTP_INTERFACE_10:0])
passcount = 0 ;(来源: [HTTP_INTERFACE_10:0])
failcount = 0 ;(来源: [HTTP_INTERFACE_10:0])
novelToken = 2.2252675.2252675eSuMFdgKyXmoV6GlHrmg0mVjZeCllKfgpmykqMhTuz4_0000 ;(
52675.2252675eS...)
billType = 2 ;(来源: [HTTP_INTERFACE_10:2])
```

```
#####
python日志:
[HTTP_INTERFACE_10]开始执行...
执行接口[HTTP_INTERFACE_1]执行结果:PASS
请求参数是: pageIndex=1&billType=1&pageSize=15
请求参数是: pageIndex=1&billType=2&pageSize=15
[HTTP_INTERFACE_10]<DONE: PYTHON执行成功>
```

## 问题6 断言:

### 1. 最简单的断言

```
ASSERT({"data":{"total":1}});
#ASSERT({"data":{"items":[{"subject":"新用户奖
励","coin":0,"vipDay":3,"price":0}], "total":1}});
```

### 2. 断言是否是预期值

```
# python
configJumpUrl = "{{editJumpUrl}}"
resultDict = json.loads(const("RESP_TEXT"))
noverJumpUrl = resultDict["data"]["items"][0]["redirect"]["url"]
asserts( '$VAR[configJumpUrl] [==] $VAR[noverJumpUrl]')
```

### 3. 断言

```
```python ASSERT(清理user成功 [IN] $CONST[RESP_TEXT]);
ASSERT(16264 [NOT_IN] $CONST[RESP_TEXT]);
```

备注: python模式下的关联和断言

```
# python
asserts('16264' in const("RESP_TEXT"))
resultDict = json.loads(const("RESP_TEXT"))
recordID = resultDict["data"]["items"][0]["id"]
```

注意 模式一定要统一，不能上面是关键字下面是python

断言的值需要进行算法运算（使用**EVAL**）

## 关键字

**EVAL**( python\_scripts )

## 参数说明

参数	说明	举例
python_scripts	合法的python表达式	python语句

## 使用示例

只能在**准备/断言恢复**中使用

在**准备/断言恢复**中使用方式如下：

假如执行接口返回值如下：

```
a = 1;
```

```
b = EVAL($VAR[a]+1);
```

执行后b为2.

```
# python
```

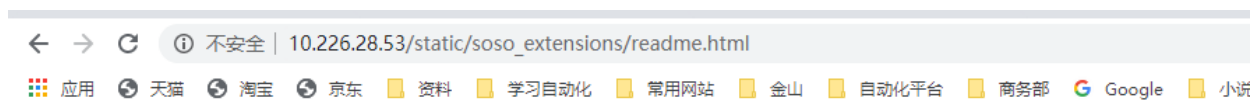
```
resultDict = json.loads(const("RESP_TEXT"))
```

```
novel_newReadCount = resultDict["data"]["novelReadCount"]
```

```
asserts( 'EVAL($VAR[novel_newReadCount]-$VAR[novelReadCount]) [=] 1')
```

## 问题7 chrome录制

[http://10.226.28.53/static/soso\\_extensions/readme.html](http://10.226.28.53/static/soso_extensions/readme.html)



### Chrome插件-zip版

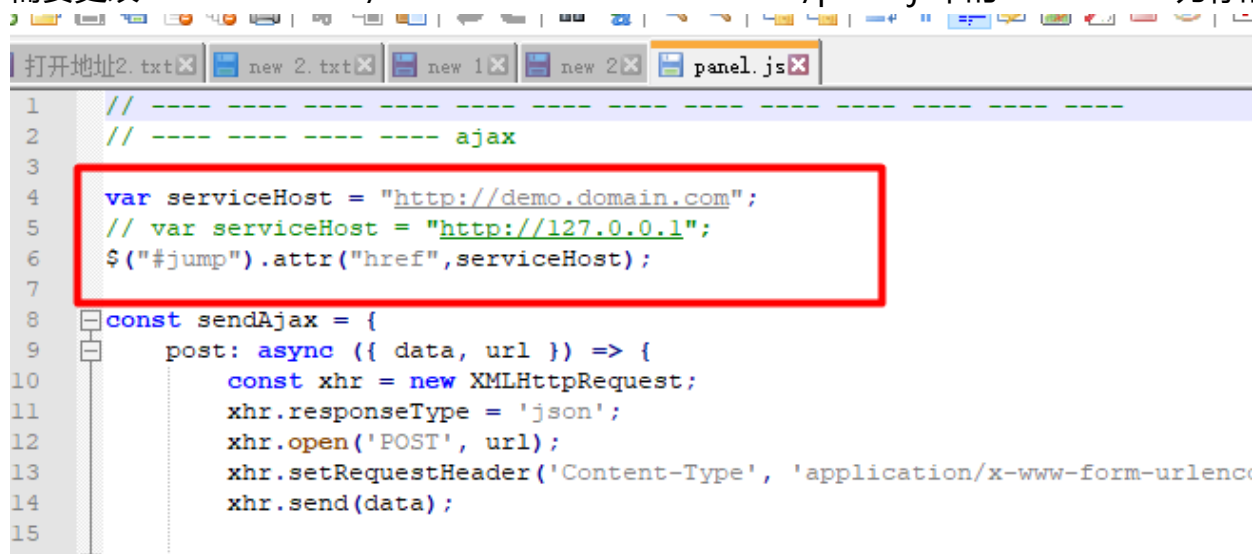
#### 安装说明:

- 1、下载后解压出文件夹sosotest-chrome-extension。
- 2、打开Chrome浏览器，在地址栏输入chrome://extensions，回车。
- 3、打开开发者模式。
- 4、点击按钮【加载已解压的扩展程序】，然后选择解压后的文件夹sosotest-chrome-extension，然后点击确定。

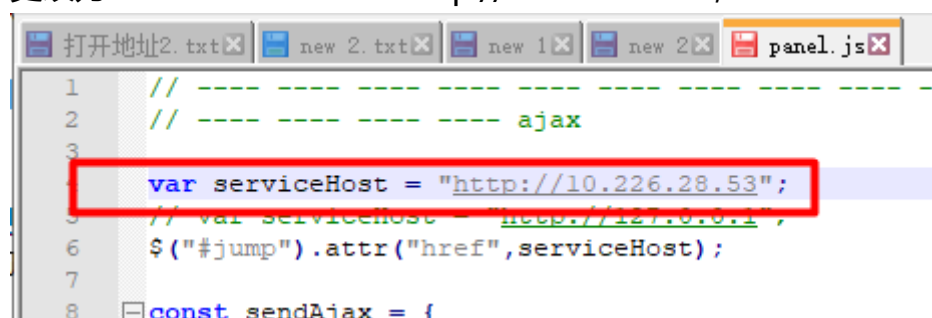
#### 使用说明:

- 1、先打开Chrome，然后登录自动化测试平台。
- 2、在同一个chrome中，打开一个新的Chrome tab，点击F12，点击右侧的sosotest-network。
- 3、在地址栏输入地址进行请求。
- 4、插件中会显示所有的http请求，点击要添加的接口，会直接跳转到用例添加页面。
- 5、然后就可以调试等，完成后保存即可。

需要更改AutotestWebD/sosotest-chrome-extension/panel.js中的serviceHost为你的域名



更改为 var serviceHost = "http://10.226.28.53";



#### 备注:

1. 一定要保证先登录测试平台，有会话
2. 执行信息中的使用服务配置地址一定要点一下

问题8 str类型的参数关联问题处理

传递参数userId 要求是str

POST

/api/v1/coin/offer

发放书币

发放书币

Parameters

No parameters

Request body

Example Value | Schema

ReqCoinOffer

description:

发放书币

action

number

默认为0, 如果24小时内已发放给当前用户, 则返回一个错误信息ERR\_COIN\_OFFER\_IN\_24HOUR, 为1时则直接给用户加书币

coin\*

number

数量

remark\*

string

理由备注

userId\*

string

用户ID

在接口准备

准备

```
1 # python
2 DEBUG_MODE = True # 当执行完后, 执行信息中会展示执行的python代码, 执行后的作用域中的变
3 userId = "2252769" # 调用类中的静态函数
4 log('生成的userId是: %s' %userId)
```

执行信息

引用这个参数{{userId}}



准备

```

1 # python
2 DEBUG_MODE = True # 当执行完后，执行信息中会展示执行的python代码，执行后的作用域中的变量、类、函数等信息。
3 userId = "2252769" # 调用类中的静态函数
4 log('生成的userId是: %s' % userId)

```

执行信息

novelConfig  自定义请求地址,例:http://127.0.0.1 自动重定向

POST /noveladmin/api/v1/coin/offer 20 1

PARAMS HEADER(3) BODY

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary

JSON(application/json)

```

{
  "userId": {{userId}},
  "coin": 1,
  "remark": "测试",
  "action": 1
}

```

断言恢复

```

1 ASSERT('stat': "OK")

```

报错:

编号	测试结果	执行环境	实际结果	断言结果	执行前耗时	执行耗时	执行后耗时	总耗时	操作
1	FAIL	小说管理后台	["message": "Unmarshal type error: n expected=string, got=number, field=userId, offset=20"]	FAIL: 预期结果断言失败! 预期结果["stat": "OK"]不包含于 实际结果["message": "Unmarshal type error: expected=string, got=number, field=userId, offset=20"]	120	71	18	209	<a href="#">查看详情</a>
2	PASS	小说管理后台			127	81	18	226	

问题在于:

变量替换的时候, 这个引号要自己加的

2:40:17 PM



子夜

变量替换的时候, 这个引号要自己加的



子夜

这个无法优化, 这个本身就是一个替换关系, 无法判断类型。



子夜

你简单的理解就是 变量引用的时候, 其实就是把变量的值 替换到着, 不会带着引号啥的给你。



子夜

假设有的地方要用单引号, 我怎么给你判断, 对吧? 所以这个要用户自己控制。

就是说自己控制 明白

所以:

最终的写法应该是

```
"userId": "{{userId}}"
```

### 问题9 sql语句执行

```
# python
```

```
DEBUG_MODE = True
```

```
configNovelID = 16264
```

```
resultDict = json.loads(const("RESP_TEXT"))
```

```
newNovelID = resultDict["data"]["items"][0]["novel"]["id"]
```

```
dbResultNovelID = db_select("default", "SELECT novel_id FROM wps_novel_a.read_log  
WHERE user_id = {{userId}} ORDER BY id DESC LIMIT 1")
```

```
log('获取的dbResultNovelID是: %s' %dbResultNovelID)
```

```
asserts( '$VAR[newNovelID] [IN] $VAR[dbResultNovelID]')
```

问题10 为啥任务单个执行都是正确的 一起放到任务集就执行失败了

任务集是多个任务并行 可能同一批数据做操作就会出问题 建议做好隔离喔

比如不同用户 不同模块才加入同一个任务集