ProgrammerSought

search

Python + Requests + Pytest + Excel + Allure (2) Interface Automation Test Battle (with code)

tags: test python

TOP

byPreviousBlog, we have already put the environment, start writing code.

1 First use Python simulation request:

1.1 GET request

```python
import requests

'''
 URL Parameters Request method: URL parameter
 For example: request http://httpbin.org/get?first_name=hello&last_name=word

'''

params={"first_name":"hello","last_name":"word"}
responds=requests.get("http://httpbin.org/get",params=params)
print(responds.text)
print(responds.url)
```

1.2.1 POST request

```
import requests


params={"first_name":"hello","last_name":"word"}
headers={"Content-Tpye":"application/x-www-form-urlencoded"}
responds=requests.post("http://httpbin.org/post",data=params,headers=headers)
print(responds.text)
print(responds.url)
print(responds.request)
```

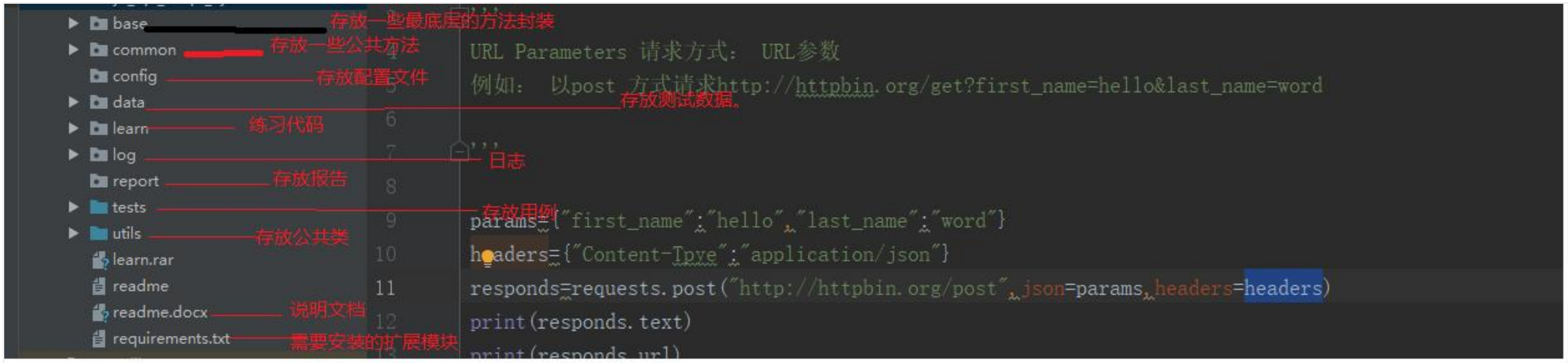
1.2.2 post json

```
import requests

'''
 URL Parameters Request method: URL parameter
 For example: request http://httpbin.org/get?first_name=hello&last_name=word in a POST method

'''

params={"first_name":"hello","last_name":"word"}
headers={"Content-Tpye":"application/json"}
responds=requests.post("http://httpbin.org/post",json=params,headers=headers)
print(responds.text)
print(responds.url)
print(responds.request)
```

2 Pytest project actual combat:

2.1. Create a subdirectory in turn is as follows:

Base: Store some of the bottom-up method package, protocol, request transmission, etc. Common: Store some public methods. Config: Store profile. DATA: Store test data. Log: Store the log. Report: Store report. Saving case. UTILS: Store public categories. Readme: Used to explain the document. Requirements.txt: Used to record all dependent package extreme version numbers, easy to deploy, can automatically generate and install through the PIP command.



```
'''
URL Parameters 请求方式： URL参数
例如： 以post 方式请求http://httpbin.org/get?first_name=hello&last_name=word

'''

params={"first_name":"hello","last_name":"word"}
headers={"Content-Tpye":"application/json"}
responds=requests.post("http://httpbin.org/post",json=params,headers=headers)
print(responds.text)
print(responds.url)
```

2.2 Packaging Request Method (Create Method.py below the Base Directory)

2.2.1 There are two ways to encapsulate:

The first: directly call the Request method below the Requests library, and define all the parameters you need to use, divided into line arguments and row parameters, the actual parameters must be transmitted, the row parameters can be given to the default value, when calling Re-assay can also use the default value. This method is less code, and the request will automatically send request to the server according to the incoming parameters.

```
import requests
import json

class ApiRequest(object):

         # ----- The first request method package the Request library, the call can be used according to
    def send_requests(self,method,url,data=None,params=None,headers=None,cookies=None,json=None,files=No
        self.r=requests.request(method,url,data=data,params=params,headers=headers,cookies=cookies,json=
        return self.r
```

Second: Pack each request in the form of functions, respectively, and call Requests to send requests. Take GET and POST as an example: PUT, DELETE requests Press the same method package, each request mode package is completed, then define a master method, directly call the main method to automatically invoke each request function according to the request method, here you can also ask the request The summary package is sent directly to the respective functions. This method package involves more code amount, and has not been able to use the Requests library well, so the first method is recommended.

```
import requests
import json

class ApiRequest(object):

     def get(self,url,data=None,headers=None):
         if headers is not  None:
             res=requests.get(url=url,data=data,headers=headers)
         else:
             res=requests.get(url=url)
             return res.json()

           "" "POST request" ""
     def post(self,url,data,headers):
         if headers is not None:
             res=requests.post(url=url,data=data,headers=headers)
         else:
             res=requests.post(url=url,data=data)
         if str(res)=="<Response [200]>":
             return  res.json()
         else:
             return  res.text()



           "" "" "" "" ""
     def all_method(self,method,url,data=None,headers=None):
         if method=='get' or method=='GET':
             res=self.get(url,data,headers)
         elif method=='post' or method=='POST':
             res=self.post(url,data,headers)
         elif method=='put' or method=='PUT':
             res=self.post(url,data,headers)
         elif method=='delete' or method=='DELETE':
             res =self.delete(url,data,headers)
         else:
                           Res = 'request mode is incorrect'
         return  json.dump(res,ensure_ascii=False,indent=4,sort_keys=True,separators=(',',':'))
```

2.2.2 Packaging Reading Documents (create public.py in the CommON directory):

TOP

1, here you need to use Python's OS library, the OS library is the Python standard library, contains hundreds of functions, commonly used path operations, process management, environmental parameters, etc.

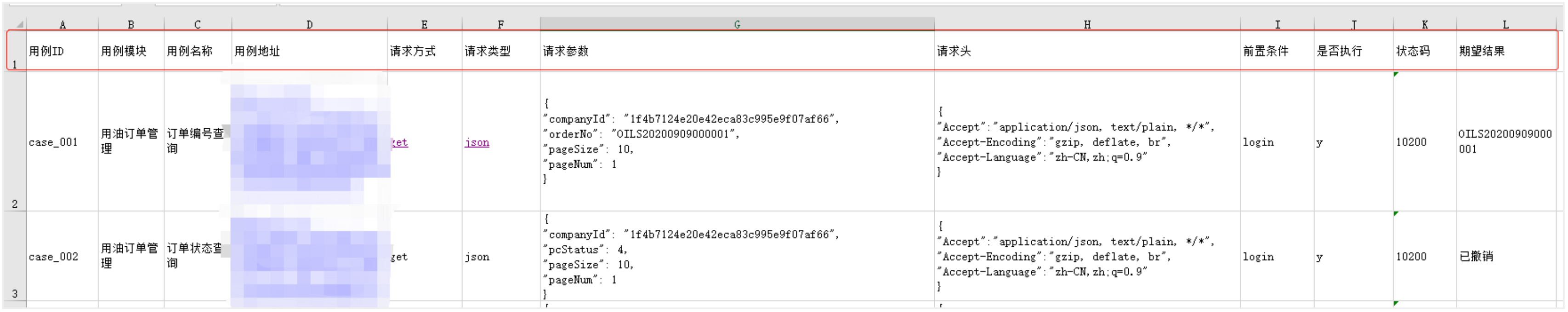
2. Package the files in the acquisition directory and acquisition directory, directly call and passed into the corresponding parameters. As follows:

```
import  os
 Os.path.DirName (__ file __) # Get the current directory
 Os.path.Dirname (Os.Path.DirName) # Get the previous level of the current directory

 # Get the specified directory
def fileDir(data):
    """
         : param data: catalog
         : return: return
    """
    base_path=os.path.dirname(os.path.dirname(__file__))
         Return Os.path.join (Base_path, DATA) # returns the obtained directory
 # Get the file under the path, call the two parameters to replace it, otherwise use the default parame
def filePath(fileDir="data",fileName="data.xls"):
    """
         : param filedir: catalog
         : param filename: file name
         : return: return
    """
    return os.path.join(os.path.dirname(os.path.dirname(__file__)),fileDir,fileName)
```

2.2.3, prepare data (create data.xlsx in the data directory)

Write the interface to the Excel table and use the fields you need to use into Excel and then read directly.

TOP



用例ID	用例模块	用例名称	用例地址	请求方式	请求类型	请求参数	请求头	前置条件	是否执行	状态码	期望结果
case_001	用油订单管理	订单编号查询	[redacted]	get	json	{ "companyId": "1f4b7124e20e42eca83c995e9f07af66", "orderNo": "OILS20200909000001", "pageSize": 10, "pageNum": 1 }	{ "Accept":"application/json, text/plain, */*", "Accept-Encoding":"gzip, deflate, br", "Accept-Language":"zh-CN,zh;q=0.9" }	login	y	10200	OILS20200909000001
case_002	用油订单管理	订单状态查询	[redacted]	get	json	{ "companyId": "1f4b7124e20e42eca83c995e9f07af66", "pcStatus": 4, "pageSize": 10, "pageNum": 1 }	{ "Accept":"application/json, text/plain, */*", "Accept-Encoding":"gzip, deflate, br", "Accept-Language":"zh-CN,zh;q=0.9" }	login	y	10200	已撤销

2.2.4 Package Reading File Method (Utils Directory Create OperationExcel.py)

Define a file class separately define a method to get the sheet table, define a method to get all the data stored lists and*Define global variables for headers*。

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-
 # Read Excel table data
import xlrd
import xlwt
from common import public
class OperationExcel:


         # Get the shell table
    def getSheet(self):

                 Book = xlrd.open_workbook (public.filepath ()) # The file parameters have been passed

                 Return book.sheet_by_index (0) # Get the Sheet table according to the index


         # Read all data in the form of a list
    def getExceldatas(self):            data = []
                 Title = self.getsheet (). Row_values (0) # 0 gets the first line is the header
                 For Row In Range (1, Self.getsheet (): # get obtained from the second line
            row_value=self.getSheet().row_values(row)
                         Data.Append (Dict (Zip (Zip (Title, Row_Value)) #
        return data

class ExcelVarles:
         Case_id = "Use case ID"
         Case_module = "Use Delivery Module"
         Case_name = "Used Case Name"
         Case_url = "Use case address"
         Case_method = "Request Method"
         CASE_TYPE = "Request Type"
         Case_data = "Request Parameters"
         Case_Headers = "Request Head"
         Case_preposition = "Pre-Condition"
         Case_srun = "Do Not Execute"
         Case_code = "Status Code"
         Case_Result = "Expected Results"
# hc = OperationExcel()
# print(hc.getExceldatas())


```

2.2.5 Package Login Method (Create Login.py below the Common Directory)

Get the Token to use it to interface, use the @ Pytest.FixTure decorator Each time you perform all the cases before the login method gets to TOKEN and returns.

```
import pytest
```

```
# import requests
                    # import json
@pytest.fixture(scope='session')
 #@pytest.FixTure Decorator How to run once before running the module
def token():
         # "" "" Get token and return "" "
    # url="xxxx"
         # haders = {...} # Request header information
         # Data = {...} # Request parameters
    # r=requests.post(url=url,data=json.dumps(data),headers=headers)
         # # Return to all token information
    # token=str('Bearer' + r.json()['data']['access_token'])
    token = '23423525252'
    print(token)
    return token
```

2.2.6, Packaging Example (Tests Directory Create Test_Gwyc_API_All.py)

You need to use the @ Pytest.mark.Parametrize () decorator to make a package case, call getExcelDataS () to store the use case to the decorator, "DATA" is an alias. As shown in the following image, the separate request head and parameters are empty, and insert Token into Headers, so that each interface can be used to log in token, do not have to call to get token.

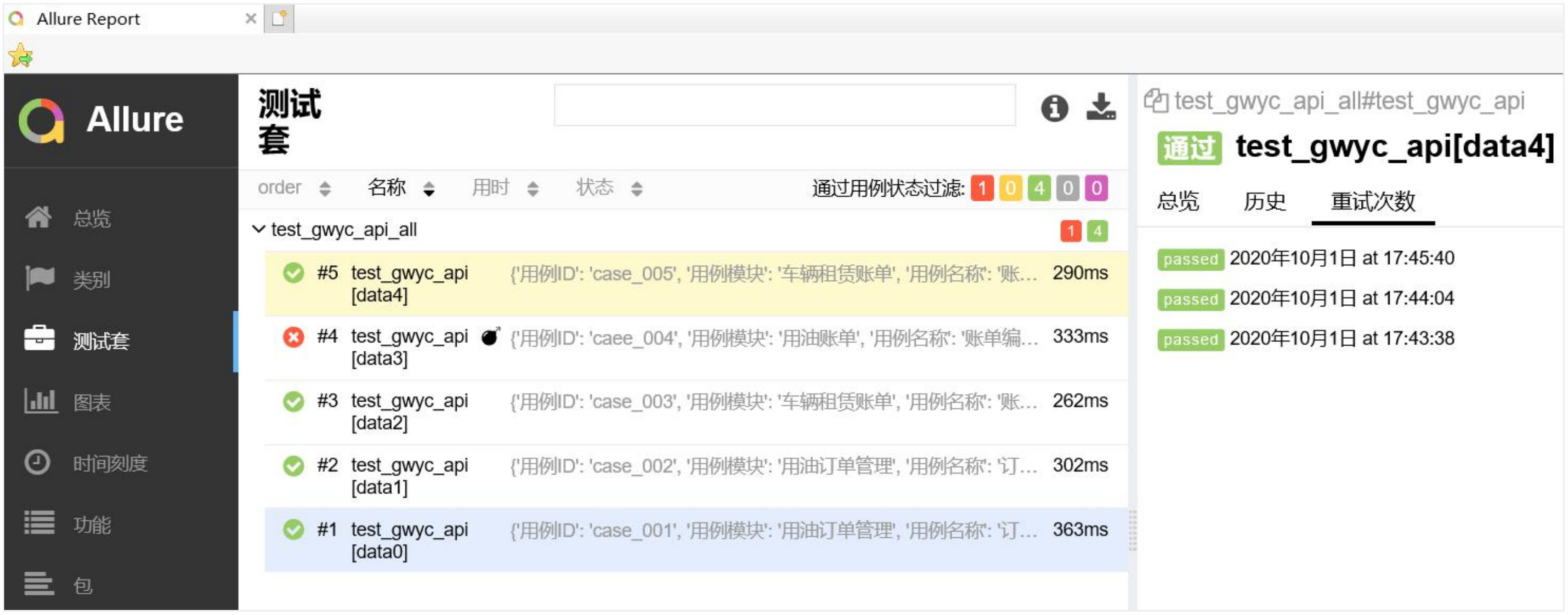
```
import  pytest
import json
from base import method
from utils import operationExcel
 # Parameterization runs all use cases
 @ pytest.mark.Parametrize ('data', OperationExcel.OperationExcel (). getExcelData ()) # decorator for pack
def test_gwyc_api(data):
         #
```

```
    headers=data[operationExcel.ExcelVarles.case_headers]
    if len(str(headers).split())== 0:
        pass
    elif len(str(headers))>= 0:
                Headers = JSON.LOADS (Headers) # Convert to Dictionary
 #HEADERS ['Authorization'] = Company_Login_Token # Get logging back to the Token and add it to the read he
        headers=headers

         # Time as the request parameter is empty
    params=data[operationExcel.ExcelVarles.case_data]
    if len(str(params).split())== 0:
        pass
    elif len(str(params))>= 0:
        params=params

         #

    case_code=str(data[operationExcel.ExcelVarles.case_code])
    def case_result_assert(r):
                askERT R.JSON () ['Errcode'] == Case_code # status code
                Assert Data [OperationExcel.Excelvarles.case_Result] in json.dumps (r.json (), EnSure_asci

         #
    if data[operationExcel.ExcelVarles.case_method]=='get':
        r=method.ApiRequest().send_requests(
            method='get',
            url=data[operationExcel.ExcelVarles.case_url],
            data=params,
            headers=headers
        )
        #print (r.json())
        case_result_assert(r=r)
    elif data[operationExcel.ExcelVarles.case_method]=='post':
        r = method.ApiRequest().send_requests(43 |             method='post',
            url=data[operationExcel.ExcelVarles.case_url],
            json=json.loads(params),
            headers=headers)
        # WriteContent (r.json () ['data'] ['access_tonken'] #
        print (r.json())
        case_result_assert(r=r)

if __name__ == '__main__':
         "" "Executes and generates allure test reports" ""
         Pytest.main (["- S", "- v", "- alluredir", "./ report / result"]) # Run the output and generate JS
         Import SubProcess # Take a child process through the subProcess package in the standard library, a
         SubProcess.Call ('allure generate report / result / -o repert / html --clean', shell = true) # rea
                                                  # - Clean If there is a directory, it will be cleared.
         SubProcess.call ('allure open -h 127.0.0.1 -p 9999 ./report/html', shell=true) # Generates a local
```

2.2.7, use all init to generate test reports.

Allure is a Pytest's plugin package requires download and installation and configures the path where the BIN directory in the Allure can be used in the PATH environment variable.

Store the generated JSON file to the directory where the use case is located, after the execution is complete, then generate a report directory below, which contains the result directory and html directory, the former is stored by the JSON file, the latter stores the HTML report generated after reading



test_gwyc_api_all 1 4

#5	test_gwyc_api [data4]	{'用例ID': 'case_005', '用例模块': '车辆租赁账单', '用例名称': '账…	290ms
#4	test_gwyc_api [data3]	{'用例ID': 'caee_004', '用例模块': '用油账单', '用例名称': '账单编…	333ms
#3	test_gwyc_api [data2]	{'用例ID': 'case_003', '用例模块': '车辆租赁账单', '用例名称': '账…	262ms
#2	test_gwyc_api [data1]	{'用例ID': 'case_002', '用例模块': '用油订单管理', '用例名称': '订…	302ms
#1	test_gwyc_api [data0]	{'用例ID': 'case_001', '用例模块': '用油订单管理', '用例名称': '订…	363ms

test_gwyc_api_all#test_gwyc_api

通过 test_gwyc_api[data4]

passed 2020年10月1日 at 17:45:40
passed 2020年10月1日 at 17:44:04
passed 2020年10月1日 at 17:43:38

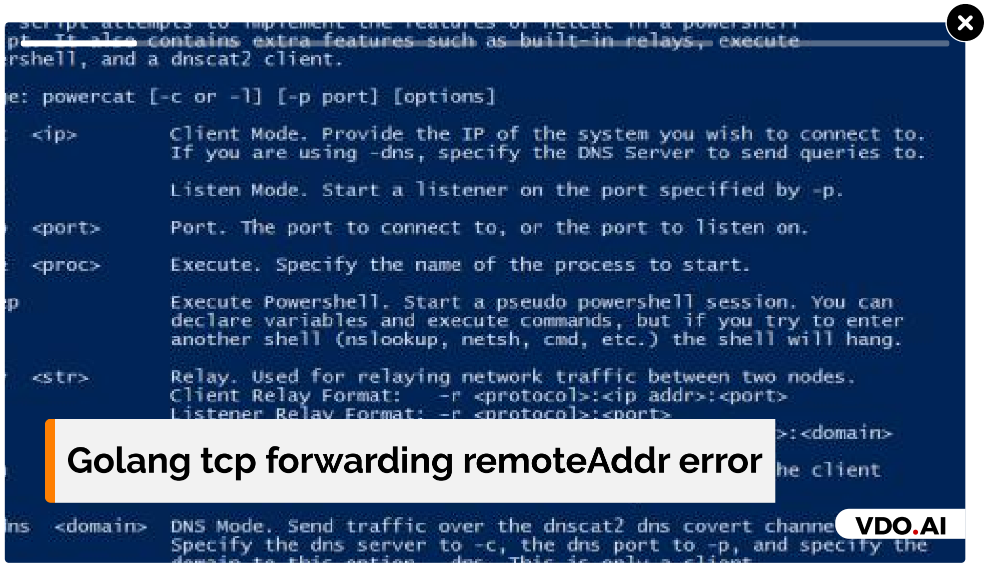
2.2.8 Package Log Method and Send Email (Log Directory Create Log.PY)

The reference role played in the automation test is relatively small to test the test report, directly call directly according to the corresponding rules.

```
import yagmail
from loguru import logger

#
 Contents = ['t         h h           h   /,
 'You can find an audio file attached.', '../tests/report/html/index.html']
 #Link Mailbox Server
yag = yagmail.SMTP( user="XXXXX@126.com", password="XXXXXXX", host='smtp.126.com')
 # send email
 Yag.send ('gao14@126.com ", 'Christmas Blessing', 'SDDSDS')
 Logger.add ('../ log / mail.log') #
```

```
logger.info("test test test")
```

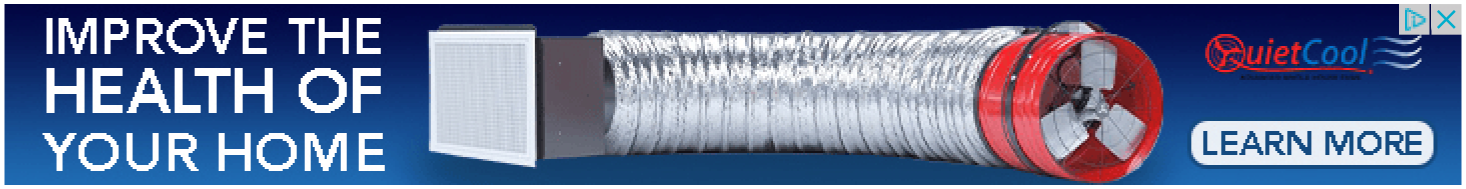
Details of the specific code:link https://download.csdn.net/download/fish_study_csdn/13979653



```
script attempts to implement the features of netcat in a powershell
pt. It also contains extra features such as built-in relays, execute
rshell, and a dnscat2 client.

e: powercat [-c or -l] [-p port] [options]

  <ip>        Client Mode. Provide the IP of the system you wish to connect to.
              If you are using -dns, specify the DNS Server to send queries to.

              Listen Mode. Start a listener on the port specified by -p.

  <port>      Port. The port to connect to, or the port to listen on.

  <proc>      Execute. Specify the name of the process to start.

p             Execute Powershell. Start a pseudo powershell session. You can
              declare variables and execute commands, but if you try to enter
              another shell (nslookup, netsh, cmd, etc.) the shell will hang.

  <str>       Relay. Used for relaying network traffic between two nodes.
              Client Relay Format:   -r <protocol>:<ip addr>:<port>
              Listener Relay Format: -r <protocol>:<port>
                                                   >:<domain>

                                                   he client

ns  <domain>  DNS Mode. Send traffic over the dnscat2 dns covert channe
              Specify the dns server to -c, the dns port to -p, and specify the
```

Golang tcp forwarding remoteAddr error

VDO.AI



Intelligent Recommendation

Use python to realize interface test automation (using pytest, requests)

...

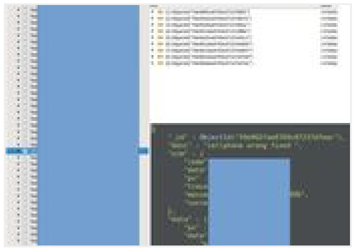
TOP

[Interface automated test] Based on pytest+requests+allure for interface testing V2.0

Interface automation test realization: pytest + request + allure V1.0 versionExplore API automated testing There are some problems with the automated script written, and some optimizations are made fo...

The near-term goal is to realize the interface automation test framework of Python pytest+allure

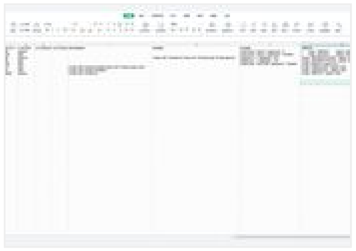
I found that the framework is relatively good, perhaps because I gave up a long period of research. Also, I have been thinking about how to integrate python into the testing work, but I have not taken...



Use mongodb+pytest+allure+jenkins to build API interface automation test

1. Introduction When I learned robotframework before, I knew that someone recommended this framework, shouting pytest Dafa is good, just recently to reconstruct a project's server-side interface autom...

TOP



Python interface automated test framework (Pytest+Allure+json path+xlrd+excel, supports Restful interface specification)

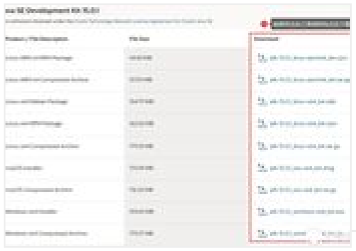
Article Directory nonsense Features Operating mechanism Known issues Environment and dependence Directory Structure Execution order config.ymal show EXcel use case display Script list Request method p...

Just a few minutes could mean saving hundreds of dollars on your energy bill
Find out how
©2018 Pacific Gas and Electric Company

More Recommendation

Jenkins implement interface automation continuous integration (Python + Pytest + Allure + Git)

TOP



When making an automated test with Python, we write a good code and then need to perform the test report, then we can further complete the automation from Jenkins. With Jenkins, we can combine the Git...

[Practical] Python + Pytest + Allure + Jenkins implement interface automation

Interface automation starting from 0 Daily exposure interface automation can be divided into two categories from the actual goal: Interface automation for simulating test data This type of interface i...

Interface test automation framework-Python+requests+Excel tes t case driver

People just can't be lazy! Get up and review the test case-driven interface automation test framework written! ** Organize ideas ** Use Python to read the test cases in the Excel sheet to get the vari...

BCB6 POST data via HTTPS interface _Python + Requests + Pytes t + Yaml + Allure implementation interface automation

Click on "Program", choose to "set to Star Tag" Quality articles, first time delivery! This project implements the technical selection of interface automation:Python+Requests+Pytes...

TOP

Interface Automation Framework (Pytest+request+Allure)

Foreword: Interface automation refers to the automation of the interface level of the simulation program. Because the interface is not easy to change and the maintenance cost is smaller, it is very po...

Related Posts

- Python+Requests+PyTest+Excel+Allure (1) Interface automation test actual combat (with code)
- Pytest + Allure + JSONPATH + Requests + Excel interface automation test framework

TOP

- pytest+allure +requests interface automation
- Realize interface automation based on Python+Requests+Pytest+YAML+Allure!
- Interface Test Python + Pytest + Allure Interface Automation Framework
- [Python] Pytest+Request+Allure+Jenkins interface automation test Demo
- requests+pytest+allure interface automated test
- Requests+pytest+allure interface automated test steps
- Requests Interface Test (Request + Pytest + Allure)
- Pytest+Allure interface automation

SOS CHILDREN'S VILLAGES USA
You can give a child the ongoing support to overcome generations of inequality.
Sponsor a child

TOP

Popular Posts

- New Year back home worry dear to grab votes, 12306python rush tickets rush tickets artifact to help you go home (including source code)
- r1 note day24 golang package understanding and use
- jboss-as-7.1.1.Final JNDI configuration and use Spring
- Bean post processor
- "VogueMe" handwritten Q&A is here! This font is very Li Yifeng!
- Proxy mode
- Dialogue with time and space (53)
- Python crawler series (2.3-requests library emulates user login)
- How coTurn runs on Windows platform
- CCF certification 201703-3 Markdown

TOP

Western
Exterminator Company
Your local pest control experts

TOP

Recommended Posts

- How does computer vision achieve self-transcendence? Larger and more accurate data
- Beginner Hadoop-Use of MapReduce Java API
- Android Bluetooth Mesh networking code detailed
- [Advanced analysis of C language] 5. Variable attributes
- Yen value counterattack, comes with noise reduction, a headset, right up admiration South A1
- Analysis of the role of spring aware various interfaces
- Android development, using Kotlin to implement WebView
- SEARCH (text の DOCTOR)
- [Essay] Particle swarm algorithm in "Who Moved My Cheese".
- Ajax simple implementation

TOP

Related Tags

python

test

software test

automated test

Test Engineer

Interface test

Performance Testing

Basic skills of testers: Jenkins

unit test

# python knowledge

TOP

mortgage.net

Your Refinance Rate

1.75%* RATE 1.81%* APR

Loan amount	$540,000
Loan term	15-Year Fixed
Credit score	Excellent

Calculate Payment

NO REGISTRATION, NO LOGIN REQUIRED
WON'T AFFECT CREDIT SCORE.

No SSN Required. *Terms & conditions apply.
NMLS # 1912050 (nmlsconsumeraccess.org). J917

Advertising Sponsor Link

- I'm Feeling Lucky

TOP

- Hard work, too tired, take a break
- Contact us to advertise

TOP