博学谷
www.boxuegu.com

第二章 接口自动化框架编写

一样的在线教育，不一样的教学品质

目录

目录 Contents

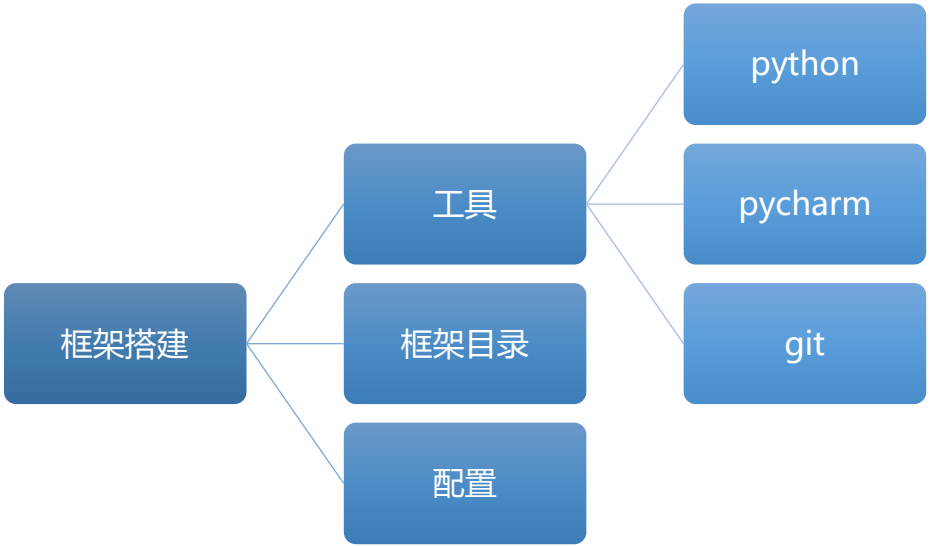
- 项目及框架搭建
- 接口用例编写
- Requests使用
- 配置文件
- 日志文件
- Pytest框架
- 结果断言
- 数据驱动
- Allure报告
- 邮件配置

项目及框架搭建

工欲善其事，必先利其器

项目及框架搭建

学习内容



项目及框架搭建

工具

https://www.python.org/downloads/

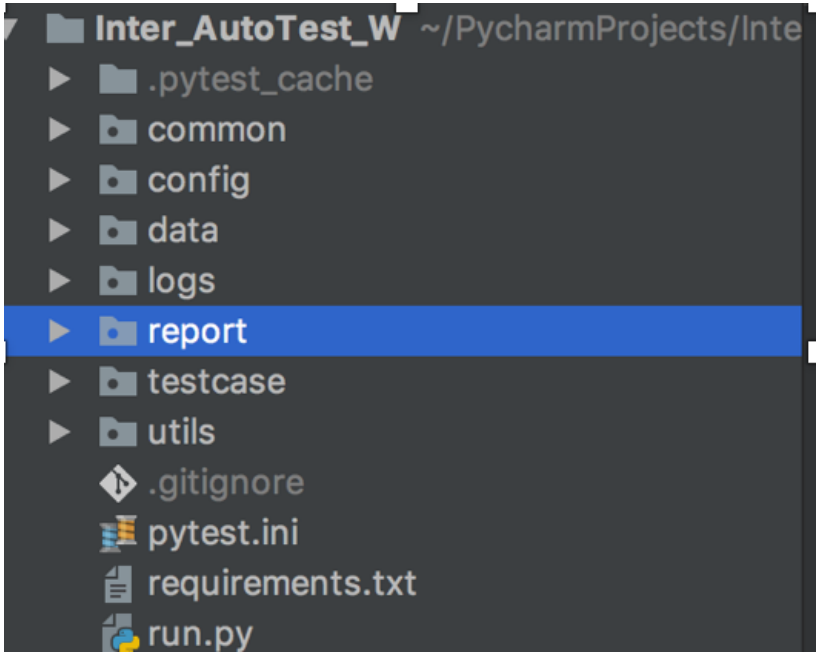
http://www.jetbrains.com/pycharm/

https://git-scm.com/downloads

项目及框架搭建

框架目录

创建项目：InterAutoTest_W



项目及框架搭建

配置

- 配置Python：Settings -> Project -> Project Interpreter
- 配置git：Settings -> Version Control -> Git
- 配置github： Settings -> Version Control -> GitHub
- 建立远程仓库并提交代码： VCS -> import into version control -> Share Project on Github

目录

目录 Contents

- 项目及框架搭建
- 接口用例编写
- Requests使用
- 配置文件
- 日志文件
- Pytest框架
- 结果断言
- 数据驱动
- Allure报告
- 邮件配置

接口用例编写

理清思路，避免遗漏

跟踪测试进展

分析缺陷

接口用例编写

被测接口

- 登陆
- 用户中心个人信息
- 获取商品列表数据
- 添加到购物车
- 保存订单

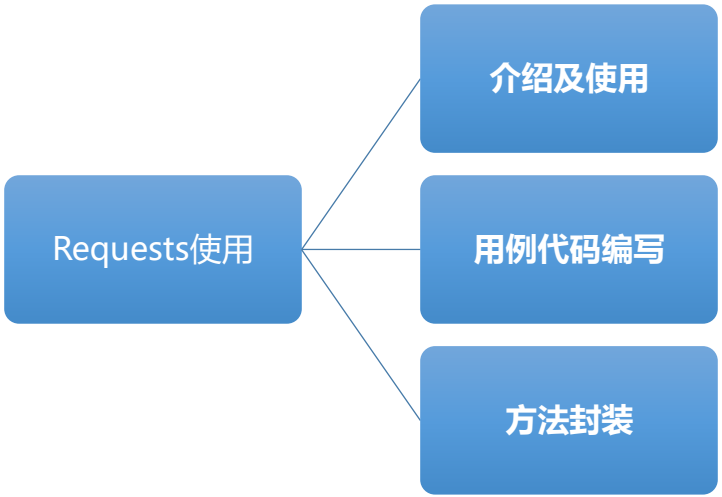
目录

- 项目及框架搭建
- 接口用例编写
- Requests使用
- 配置文件
- 日志文件
- Pytest框架
- 结果断言
- 数据驱动
- Allure报告
- 邮件配置

Requests使用

Requests使用

学习目标



Requests使用

介绍及使用

介绍：流行的接口http(s)请求工具，功能强大，简单方便，容易上手

官网： http://cn.python-requests.org/zh_CN/latest/

Requests: 让 HTTP 服务人类

发行版本 v2.18.1. (安装说明)

license Apache 2.0 | wheel yes | python 2.7 | 3.5 | 3.6 | 3.7 | codecov 66% | Say Thanks!

Requests 唯一的一个非转基因的 Python HTTP 库，人类可以安全享用。

警告：非专业使用其他 HTTP 库会导致危险的副作用，包括：安全缺陷症、冗余代码症、重新发明轮子症、啃文档症、抑郁、头疼、甚至死亡。

Requests使用

介绍及使用

pip3 install requests

Import requests
requests.get("http://www.baidu.com")

Requests使用

介绍及使用

请求返回介绍

```
r.status_code          #响应状态
r.content              #字节方式的响应体，会自动为你解码 gzip 和 deflate 压缩
r.headers              #以字典对象存储服务器响应头，若键不存在则返回None
r.json()               #Requests中内置的JSON
r.url                  # 获取url
r.encoding             # 编码格式
r.cookies              # 获取cookie
r.raw                  #返回原始响应体
r.text                 #字符串方式的响应体，会自动根据响应头部的字符编码进行
r.raise_for_status()   #失败请求(非200响应)抛出异常
```

Requests使用

用例编写

- 登录
- 个人信息
- 商品列表
- 购物车
- 订单

Requests使用

方法封装

请求返回

```
r.status_code        #响应状态
r.headers            #以字典对象存储服务器响应头，若键不存在则返回None
r.json()             #Requests中内置的JSON
r.url                # 获取url
r.cookies            # 获取cookie
r.text               #字符串方式的响应体，会自动根据响应头部的字符编码进行
```

Requests使用

方法封装

方法封装

重构

1. 封装requests公共方法
    - 增加cookies,headers参数
    - 增加函数判断url是否以https/http开头
    - 根据参数method判断get/post请求
2. 重构Post方法
3. 重构Get方法

Requests-总结

重难点

1. Requests基本使用
2. 请求返回值含义
3. Requests封装及重构

目录

- 项目及框架搭建
- 接口用例编写
- Requests使用
- 配置文件
- 日志文件
- Pytest框架
- 结果断言
- 数据驱动
- Allure报告
- 邮件配置

配置文件

Main_url= “http://211.103.136.242:8064/”

#登录用例
url = http://172.17.0.139:5004/authorizations/
data = {"username":"python",
"password":"12345678"}
r= requests.post(url,json=data)

#个人信息
url = http://172.17.0.139:5004/user/
headers = '{"Authorization": "JWT
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE1Nj
UwNjMwOTksInVzZXJfaWQiOjEsImVtYWlsIjoiOTUyNjcz
NjM4QHFxLmNvbSIsInVzZXJuYW1lIjoicHl0aG9uIn0.f3Gd
eSnzb3UGs-w1p1ejZ1rNLaaiBOAHUnN8_pq8LDE"}’
r = requests.get(url,headers=headers)

#登录用例
url = **Main_url +**” authorizations/”
data = {"username":"python",
"password":"12345678"}
r= requests.post(url,json=data)
#个人信息
url = **Main_url +**” user/”
headers = '{"Authorization": "JWT
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleH
OjE1NjUwNjMwOTksInVzZXJfaWQiOjEsImVtY
joiOTUyNjczNjM4QHFxLmNvbSIsInVzZXJuYW
icHl0aG9uIn0.f3GdeSnzb3UGs-
w1p1ejZ1rNLaaiBOAHUnN8_pq8LDE"}’
r = requests.get(url,headers=headers)

配置文件

JSON

```
{
    "base": {
        "log_level": "debug",
        "log_extension": ".log",
        "excel_sheet_name": "美多商城接口测试",
        "test": {
            "url": "http://172.17.0.139:5004/",
            "tag": "V1.0"
        },
        "online": {
            "url": "http://172.17.0.139:5004/",
            "tag": "V1.0"
        }
    }
}
```

YAML

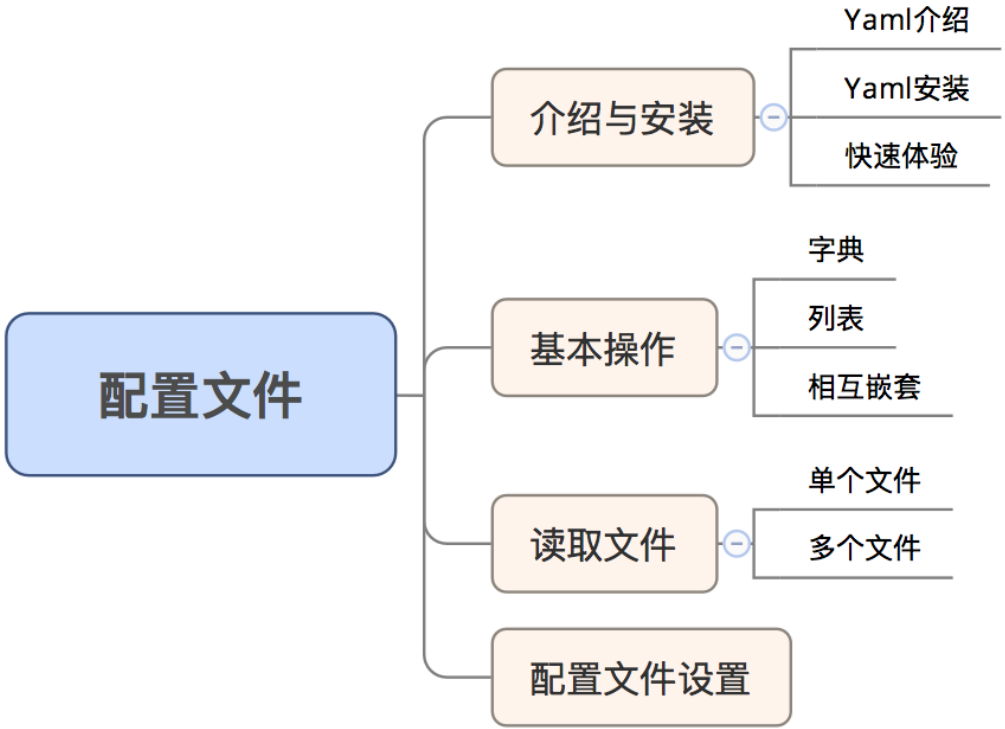
```
base:
  # log等级,debug  info warning error critical
  log_level : "debug"
  log_extension : ".log"
  excel_sheet_name : "美多商城接口测试"

  test:
    url : "http://172.17.0.139:5004/"
    tag : "V1.0"

  online:
    url : "http://172.17.0.139:5004/"
    tag : "V1.0"
```

配置文件

学习目标


配置文件
介绍与安装
Yaml介绍
Yaml安装
快速体验
基本操作
字典
列表
相互嵌套
读取文件
单个文件
多个文件
配置文件设置

配置文件

Yaml的介绍及安装

1. Yaml的介绍
   - Yaml 是一种所有编程语言可用的友好的数据序列化标准。语法和其他高阶语言类似，并且可以简单表达字典、列表和其他基本数据类型的形态

2. yaml格式作为文件的配置格式
   - yaml支持注释
   - 不必强求逗号括号等符号
   - 通过缩进来区分层级，视觉上清晰很多

3. Yaml安装
   - $ pip3 install PyYaml

Yaml的介绍及安装

快速体验

字典` {"name": "test_yaml", "result", "success"} `写成` Yaml `的形式，并输出结果

字典里的键值对用":"分隔

配置文件

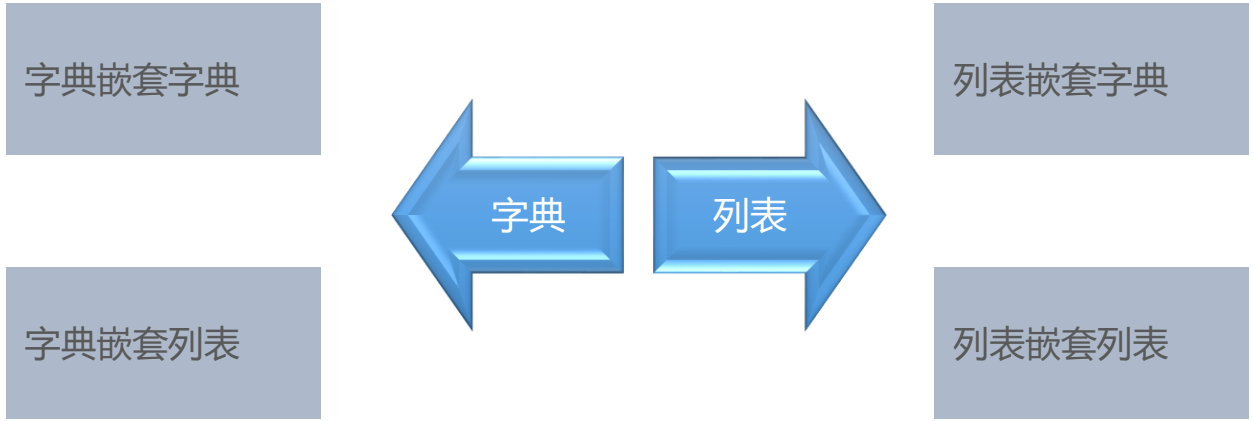
Yaml的基本操作

- 字典里的键值对用":"分隔
- 字典直接写key和value，每一个键值对占一行。
- 后面要有空格

- 一组按序排列的值（简称 "序列或列表"）
- 数组前加有 “-” 符号，符号与值之间需用空格分隔

配置文件

Yaml的基本操作



配置文件

读取Yaml文件

```
- username1: "test1"
- password1: "111"
  username2: "test2"
```

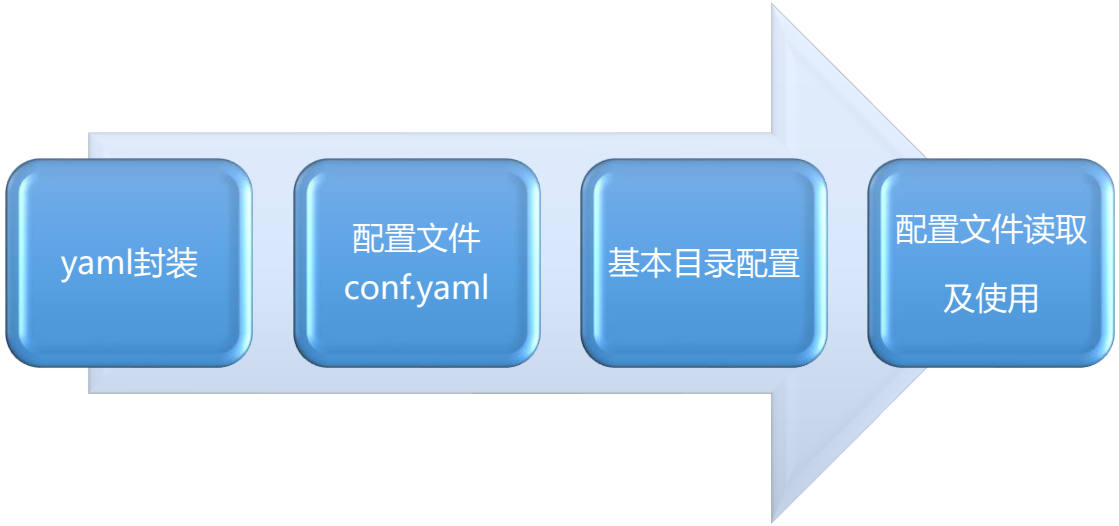
Yaml. safe_load()


```
---
"用户名称1": "test123"
"密码": "123456"
---
"用户名称2": "test456"
"密码": "123456"
```

Yaml.safe_load_all()

配置文件

Yaml配置文件设置



配置文件-总结

重难点

1. Yaml基本使用
2. Yaml封装
3. 配置文件设置

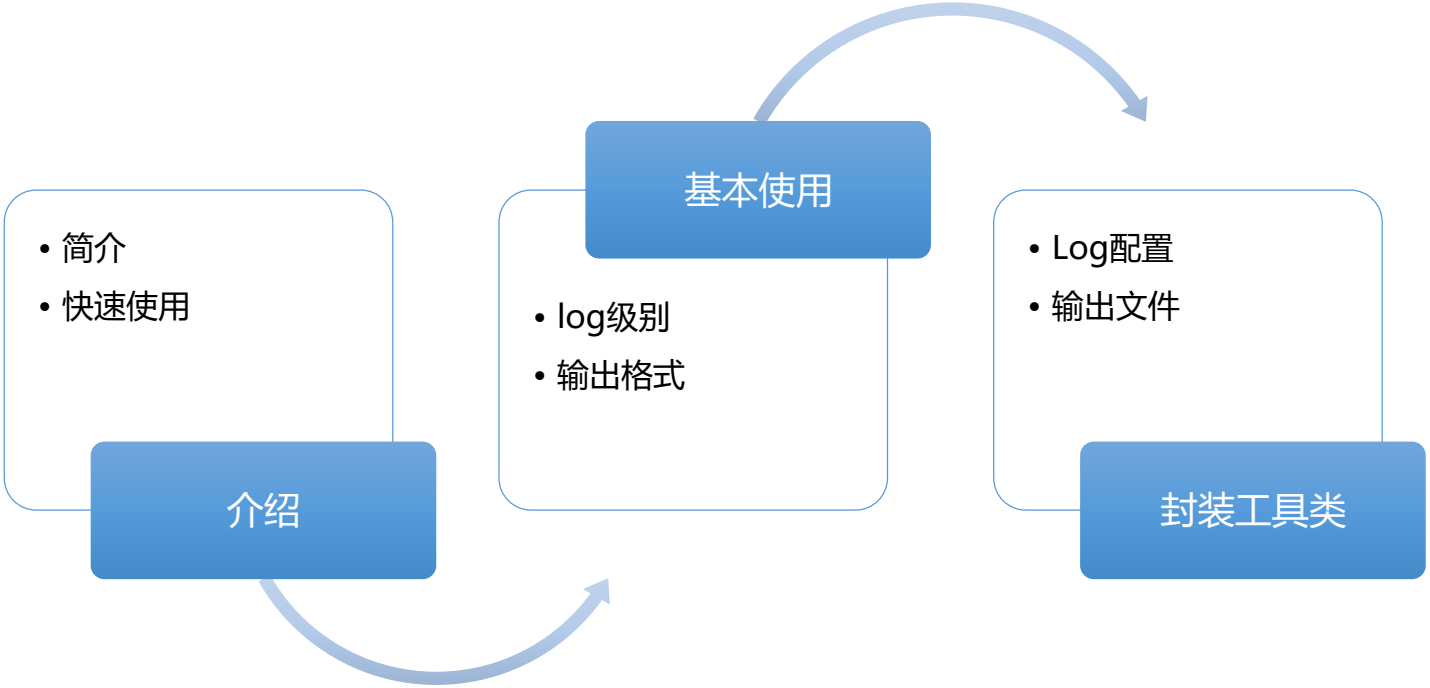
目录

- 项目及框架搭建
- 接口用例编写
- Requests使用
- 配置文件
- 日志文件
- Pytest框架
- 结果断言
- 数据驱动
- Allure报告
- 邮件配置

日志文件

日志文件

学习目标



日志文件

介绍

简介:

logging模块是Python内置的标准模块，主要用于输出运行日志，可以设置输出日志的等级、日志保存路径等

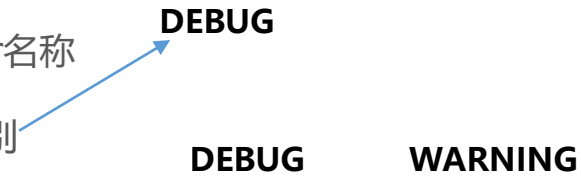
快速使用

日志文件

基本使用

日志输出控制台或文件

1. 设置logger名称
2. 设置log级别
3. 创建handler，用于输出控制台或写入日志文件
4. 设置日志级别
5. 定义handler的输出格式
6. 添加handler



日志文件

Format格式说明

- %(levelno)s: 打印日志级别的数值
- %(levelname)s: 打印日志级别名称
- %(pathname)s: 打印当前执行程序的路径，其实就是sys.argv[0]
- %(filename)s: 打印当前执行程序名
- %(funcName)s: 打印日志的当前函数
- %(lineno)d: 打印日志的当前行号
- %(asctime)s: 打印日志的时间
- %(thread)d: 打印线程ID
- %(threadName)s: 打印线程名称
- %(process)d: 打印进程ID
- %(message)s: 打印日志信息

日志文件

封装工具类

1. 封装Log工具类
2. 重构配置文件
3. 日志工具类应用

日志文件-总结

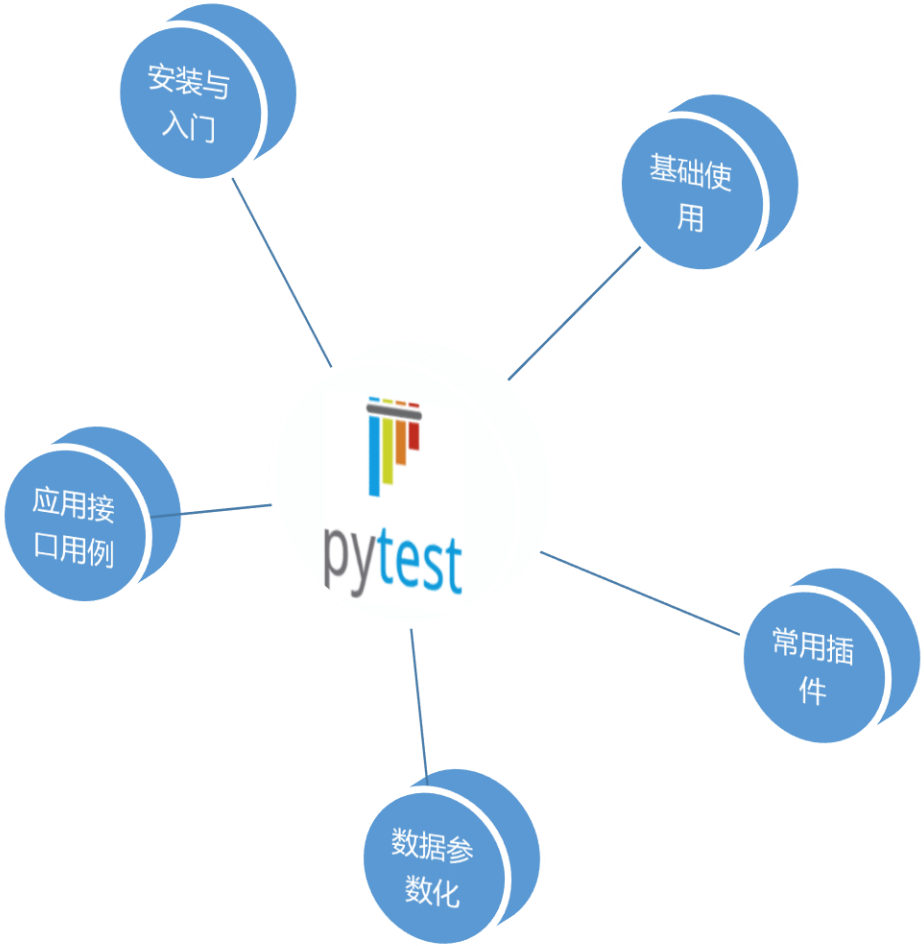
重难点

1. 日志输出控制台和日志文件
2. 日志封装以及日志级别配置

目录

- 项目及框架搭建
- 接口用例编写
- Requests使用
- 配置文件
- 日志文件
- Pytest框架
- 结果断言
- 数据驱动
- Allure报告
- 邮件配置

Pytest



Pytest

Pytest 安装与入门

介绍

- 简单灵活，
- 容易上手，
- 文档丰富
- 支持参数化，能够支持简单的单元测试和复杂的功能测试
- 具有很多第三方插件，并且可以自定义扩展
- 可以很好的和Jenkins工具结合

安装

- Pip3 install pytest

入门

- 创建你的第一个测试用例

Pytest

Pytest 基础使用

函数级别方法

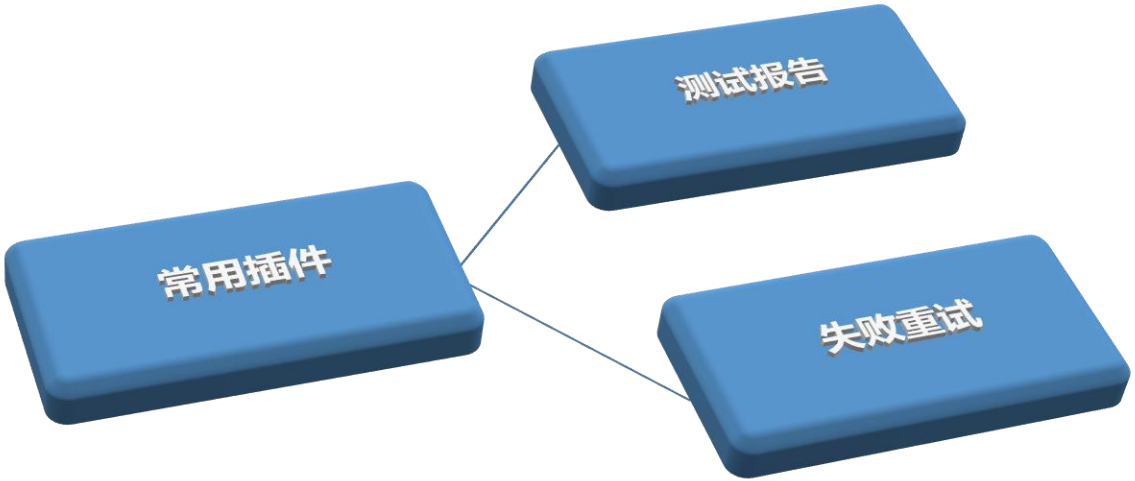
- 运行于测试方法的始末
- 运行一次测试函数会运行一次 setup 和 teardown

类级别方法

- 运行于测试类的始末
- 一个测试内只运行一次 setup_class 和 teardown_class，不关心测试类内有多少个测试函数

Pytest

Pytest 常用插件



Pytest 常用插件

测试报告

应用场景

- 自动化测试脚本最终执行是通过还是不通过，需要通过测试报告进行体现。

安装

- $ pip3 install pytest-html

使用

- 在配置文件中的命令行参数中增加 --html=用户路径/report.html

Pytest

pytest常用插件

失败重试

应用场景

- 当失败后尝试再次运行

安装

- $ pip3 install pytest-rerunfailures

使用

- 在配置文件中的命令行参数中增加 --reruns n
- 如果你期望加上出错重试的等待时间，--reruns-delay

Pytest

数据参数化

```
def login_1():
    # 1.url
    url = "http://211.103.136.242:8062//authorizations/"
    # 2.data
    data = {"username": "python",
            "password": "12345678"}
    # 3.发送post请求
    res = requests.post(url, json=data)
    print(res.text)

def login_2():
    # 1.url
    url = "http://211.103.136.242:8062//authorizations/"
    # 2.data
    data = {"username": "test123",
            "password": "123456"}
    # 3.发送post请求
    res = requests.post(url, json=data)
    print(res.text)
```

```
login_data=[{"url":"http://211.103.136.242:8062//authorizations/","data":{"user
name": "python",
        "password": "12345678"}},
        {"url": "http://172.17.0.139:5004/authorizations/", "data": {"username":
"test123",                                    "password": "123456"}}
        ]


@pytest.mark.parametrize("login",login_data)
def test_1(login):
    url = login["url"]
    data = login["data"]
    print("测试用例中login的值:%s%s" % (url, data))
    res = requests.post(url,json=data)
    print("请求结果：%s"%res.text)


if __name__ == "__main__":
    pytest.main(["-s","Test_Login.py"])
```

Pytest

数据参数化

1. 传入单个参数，pytest.mark.parametrize(argnames, argvalues)

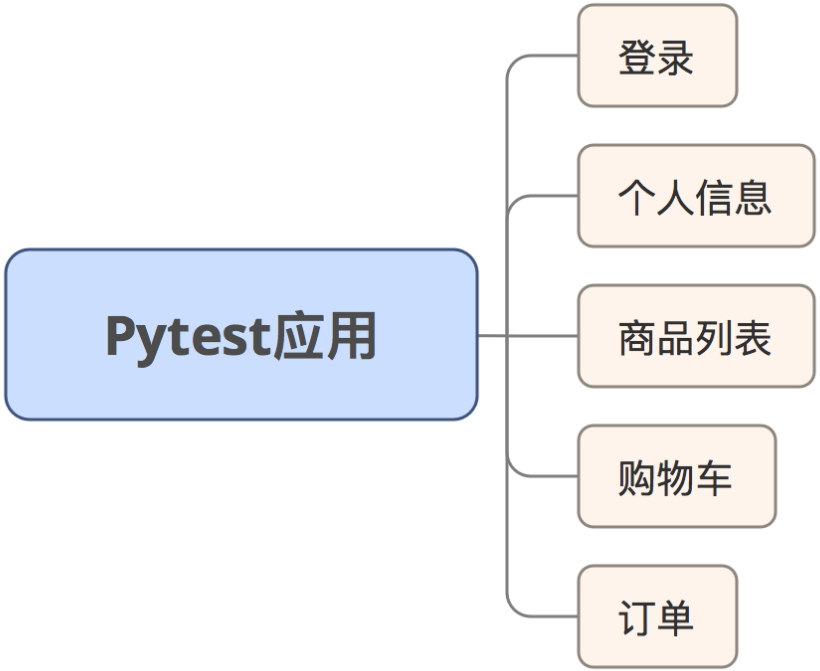
- argnames：参数名
- argvalues：参数对应值，类型必须为可迭代类型，一般使用list

2. 传多个参数，@pytest.mark.parametrize((‘参数名1，参数名2’)，[(参数1_data[0], 参数1_data[1]),(参数2_data[0], 参数2_data[1])])

- list的每个元素都是一个元组，元组里的每个元素和按参数顺序一一对应

Pytest

Pytest 应用接口用例



Pytest

Pytest 应用接口用例

运行原则

- 在不指定运行目录，运行文件，运行函数等参数的默认情况下，pytest会执行当前目录下的所有以test为前缀(test*.py)或以_test为后缀(_test.py)的文件中以test为前缀的函数

Pytest.ini

```
addopts = -s
# 当前目录下的scripts文件夹 -可自定义
testpaths = testcase
# 当前目录下的scripts文件夹下，以test_开头，以.py结尾的所有文件 -可自定义
python_files = test_*.py
# 当前目录下的scripts文件夹下，以test_开头，以.py结尾的所有文件中，以Test_开头的类 -可自定义
python_classes = Test_*
# 当前目录下的scripts文件夹下，以test_开头，以.py结尾的所有文件中，以Test_开头的类内，以test_开头的方法 -可自定义
python_functions = test_*
```

Pytest框架-总结

重难点

1. pytest基本使用
2. pytest配置文件规则
3. pytest常用插件：

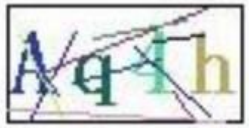
   https://plugincompat.herokuapp.com

目录

- 项目及框架搭建
- 接口用例编写
- Requests使用
- 配置文件
- 日志文件
- Pytest框架
- 结果断言
- 数据驱动
- Allure报告
- 邮件配置

结果断言

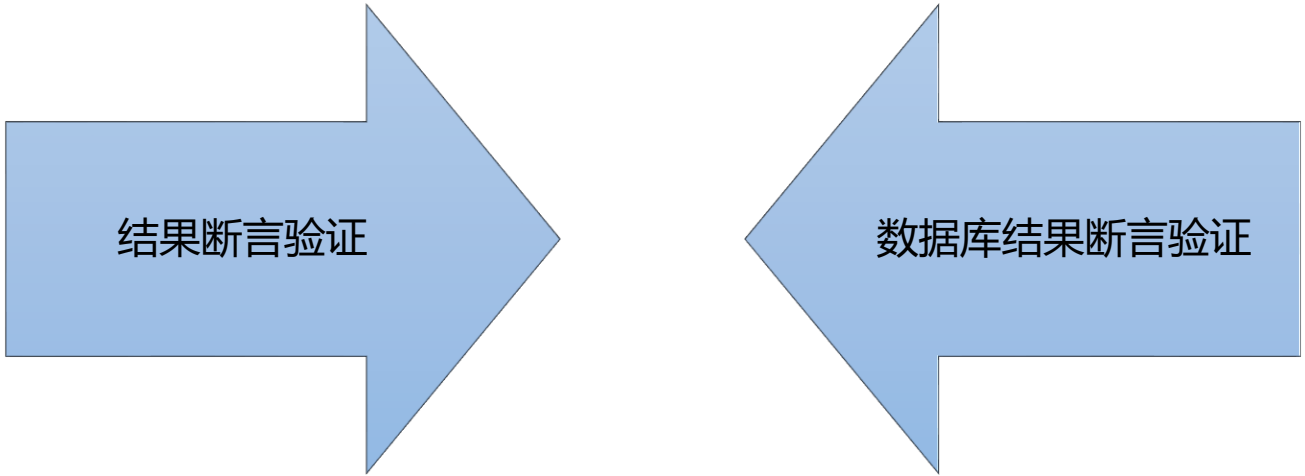
验证码：



看不清

结果断言

学习目标



结果断言

常用断言

介绍

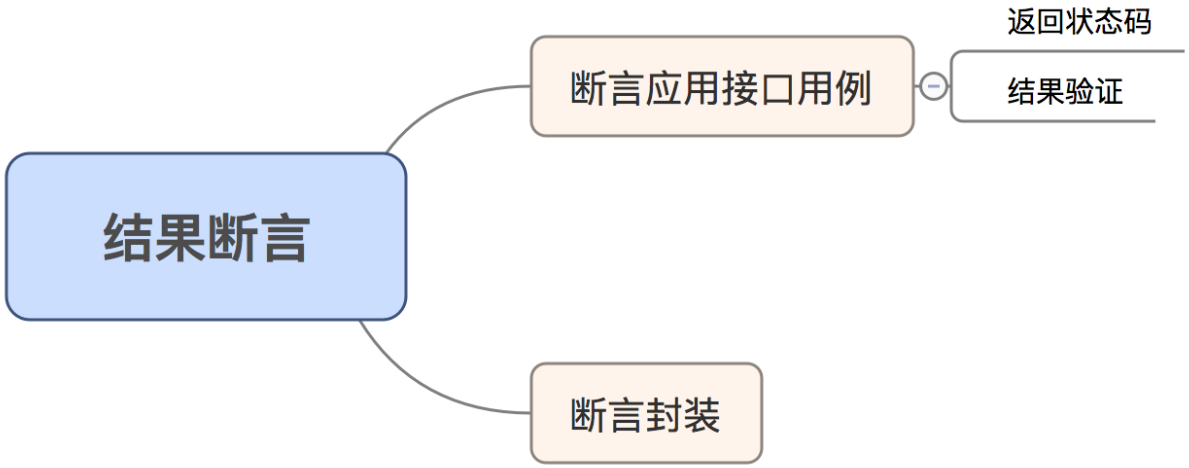
- 断言是自动化最终的目的，一个用例没有断言，就失去了自动化测试的意义了
- 断言用到的是 assert关键字
- 预期的结果和实际结果做对比

结果断言

常用断言

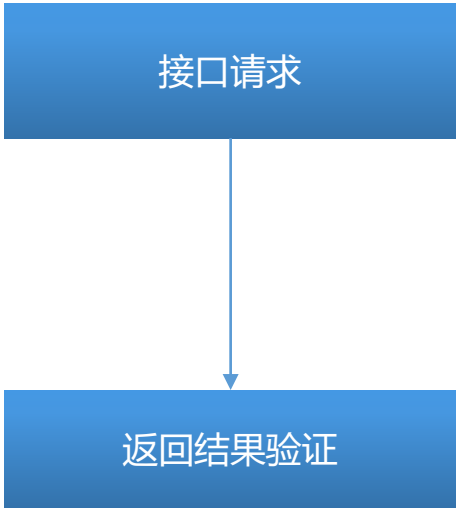
判断xx为真	• assert xx
判断xx不为真	• assert not xx
判断b包含a	• assert a in b
判断a等于b	• assert a == b
判断a不等于b	• assert a != b

结果断言



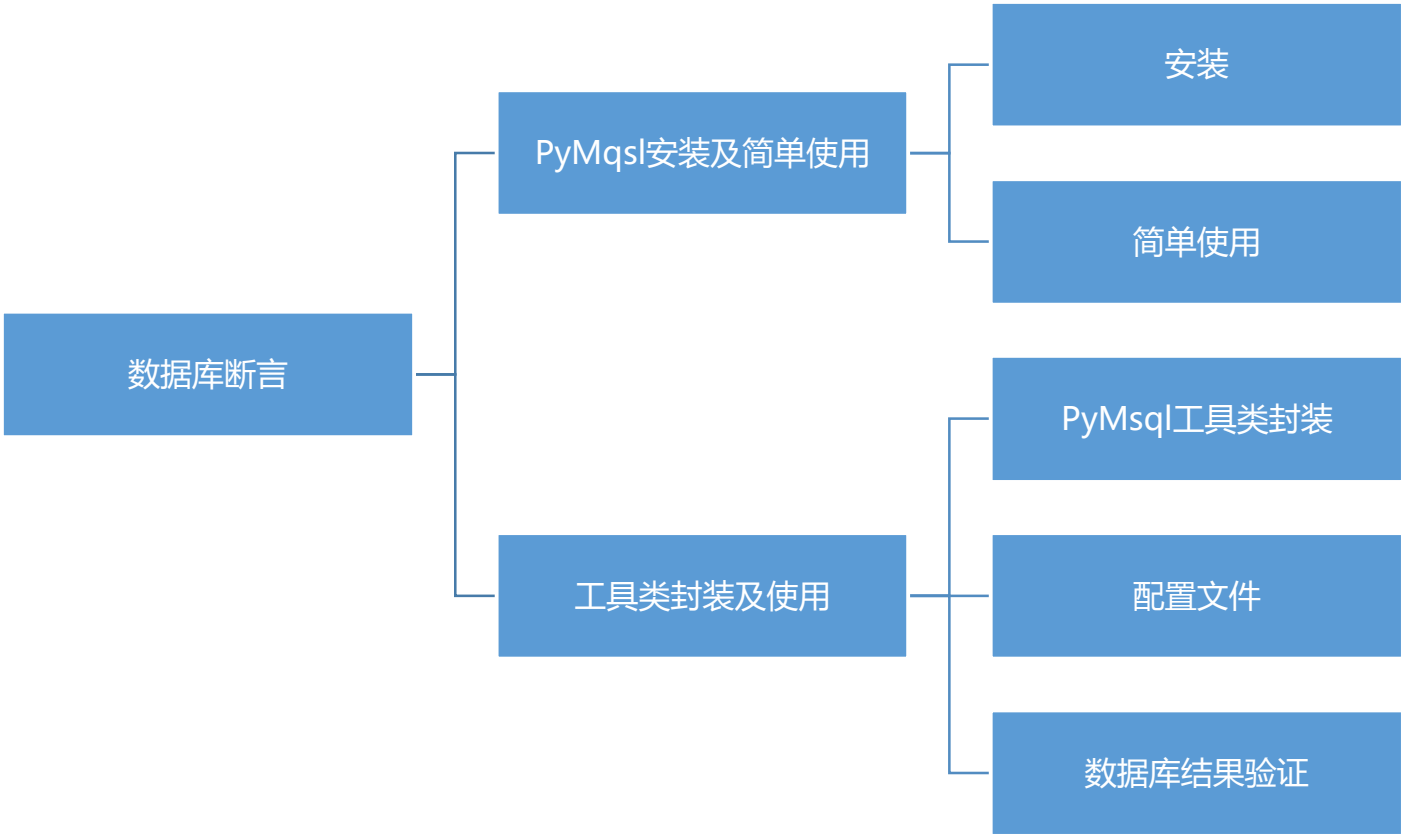
结果断言

数据库结果断言



结果断言

数据库结果断言



数据库结果断言

PyMqsl安装及简单使用

数据库结果断言

工具类封装及使用

结果断言-总结

重难点

1. 断言的目的

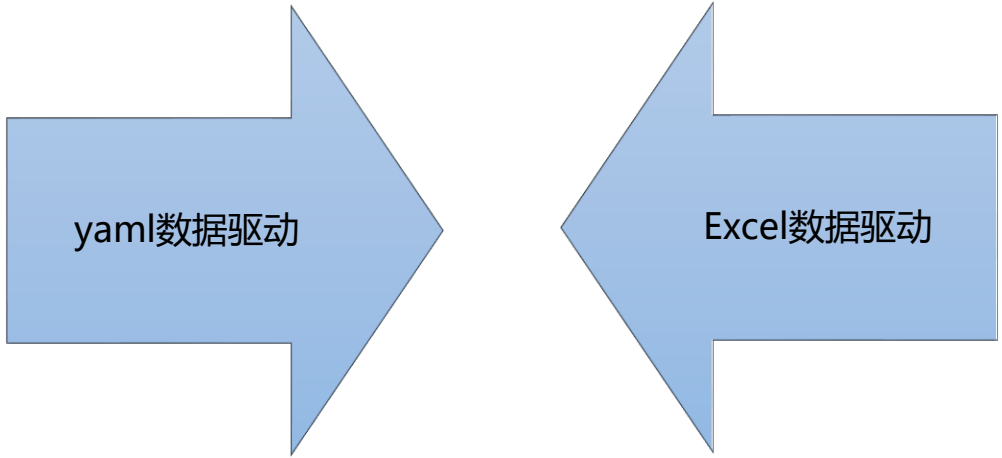
2. 断言结果验证：结果验证、数据库验证

目录

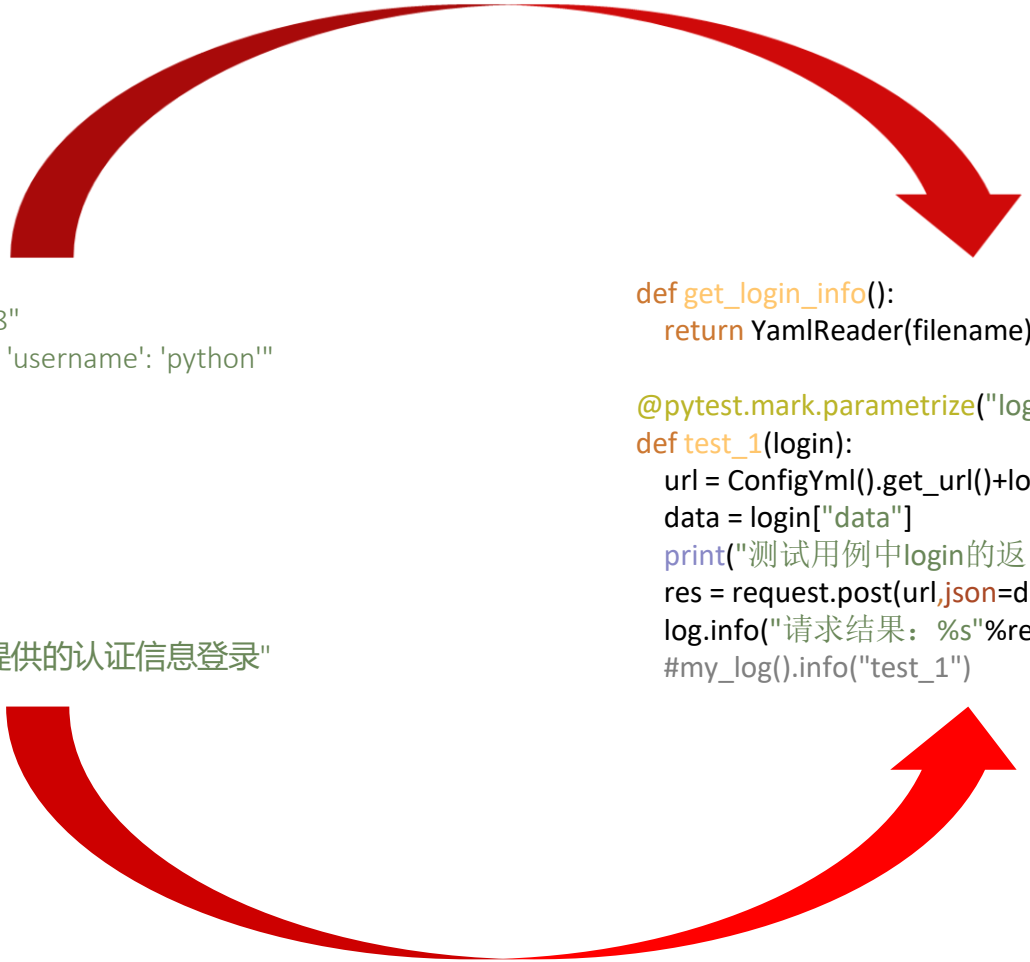
- 项目及框架搭建
- 接口用例编写
- Requests使用
- 配置文件
- 日志文件
- Pytest框架
- 结果断言
- 数据驱动
- Allure报告
- 邮件配置

数据驱动

学习目标



数据驱动



```
---
"case": "test_login_1"
"url": "authorizations"
"data":
  username: "python"
  password: "12345678"
"expect": "'user_id': 1, 'username': 'python'"

---
"case": "test_login_2"
"url": "authorizations"
"data":
  username: "test123"
  password: "test123"
"expect": "无法使用提供的认证信息登录"
```

```
def get_login_info():
    return YamlReader(filename).data_all()

@pytest.mark.parametrize("login",get_login_info())
def test_1(login):
    url = ConfigYml().get_url()+login["url"]
    data = login["data"]
    print("测试用例中login的返回值:%s%s" % (url, data))
    res = request.post(url,json=data)
    log.info("请求结果：%s"%res)
    #my_log().info("test_1")
```

数据驱动

Yaml数据驱动

数据驱动

Excel数据驱动

Excel数据驱动

Excel用例设计

用例ID	模块	接口名称	请求URL	前置条件	请求类型	请求参数类	请求参数	预期结果	实际结	备注	是否运行	headers	cookies	status_code	数据库验证
login_1	登录	登录参数为空	http://211.103.136.242:8064/authorizations/		POST	json		无效数据。期待为字典类型			y			400	
login_2	登录	登录名为空	http://211.103.136.242:8064/authorizations/		POST	json	{"username":"","password":"12345678"	username': ['该字段不能为空。']			y			400	
login_3	登录	登录名或密码为空	http://211.103.136.242:8064/authorizations/		POST	json	{"username":"","password":""}	password': ['该字段不能为空。']			y			400	
login_4	登录	登录成功	http://211.103.136.242:8064/authorizations/		POST	json	{"username":"python","password":"12345678"}	user_id': 1, 'username': 'python'			y			200	
login_5	登录	登录用户名或密码错误	http://211.103.136.242:8064/authorizations/		POST	json	{"username":"python","password":"test123"}	无法使用提供的认证信息登录			y			400	
info_1	个人信息	个人信息	http://211.103.136.242:8064/user/		get			身份认证信息未提供			y			401	
info_2	个人信息	获取个人信息正确	http://211.103.136.242:8064/user/	login_4	get			id': 1, 'username': 'python', 'mobile': '17701397029', 'ema			y	{"Authorization": "JWT ${token}$"}		200	select id,usern
cate_1	商品列表数据	商品列表数据正确	http://211.103.136.242:8064/categories/115/skus/		get	json	{ "page":"1", "page_size": "10", "ordering": "create_time" }				y			200	
cart_1	购物车	添加购物车成功	http://211.103.136.242:8064/cart/	login_4	post	json	{"sku_id": "3","count": "1", "selected": "true"}	sku_id': 3, 'count': 1			y	{"Authorization": "JWT ${token}$"}		201	
order_1	订单	保存订单	http://211.103.136.242:8064/orders/	login_4	post	json	{ "address":"1","pay_method":"1" }	order_id			y	{"Authorization": "JWT ${token}$"}		201	

Excel数据驱动

Excel读取

Excel数据驱动

Excel参数化运行

数据驱动-总结

重难点

1. Excel表格列分析与制定
2. Excel数据整体运行结构
3. Excel数据封装与实现
4. Excel数据关联
5. Yaml实现数据驱动

目录

- 项目及框架搭建
- 接口用例编写
- Requests使用
- 配置文件
- 日志文件
- Pytest框架
- 结果断言
- 数据驱动
- Allure报告
- 邮件配置

Allure

Allure

Overview

Categories

Suites

Graphs

Timeline

Behaviors

Packages

Behaviors

order | name | duration | status

Filter by status: 0 0 10 0 0

- 美多商城接口测试 10
  - 个人信息 2
    - #1 Info_1 个人信息 {'用例ID': 'Info_1', '模块': '个人信息', '接口名称': '... 16ms
    - #2 Info_2 获取个人信息正确 {'用例ID': 'Info_2', '模块': '个人信息', '接口名称': '... 244ms
  - 商品列表数据 1
  - 登录 5
  - 订单 1
  - 购物车 1

testcase.Test_Excel#test_run[case_run_list5]

Passed Info_1个人信息

Overview | History | Retries

Severity: normal

Duration: 16ms

Description

期望结果：身份认证信息未提供

测试结果：{'status_code': 401, 'result': {'detail': '身份认证信息未提供。'}, 'cookies': None}

Parameters

case_run_list: {'用例ID': 'Info_1', '模块': '个人信息', '接口名称': '个人信息', '请求...

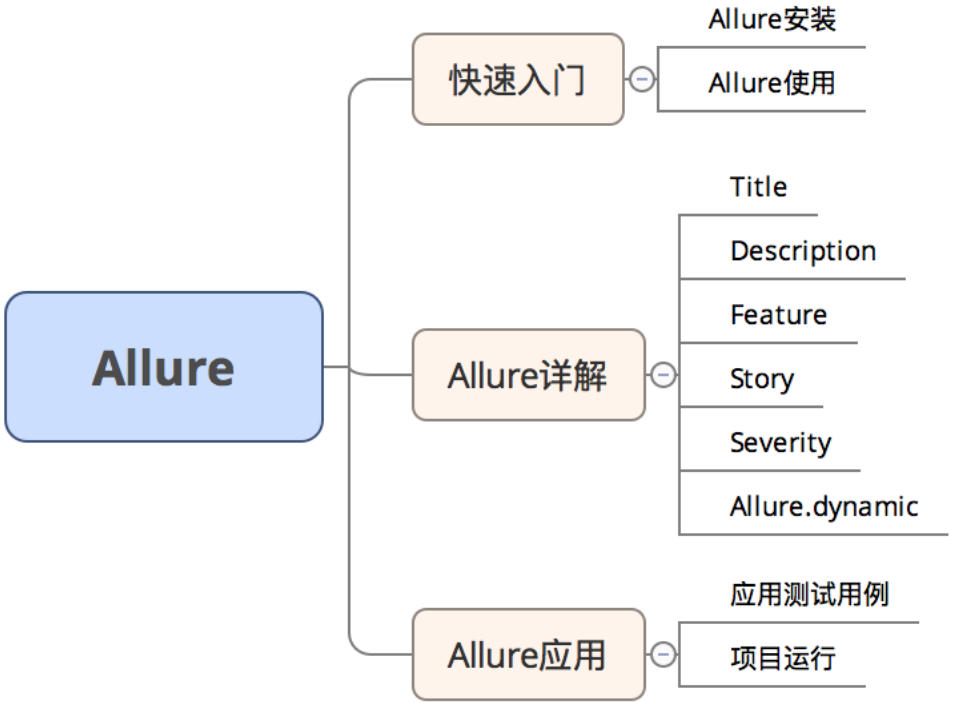
Execution

Test body

log 1 KB

Allure

学习目标


Allure
快速入门
Allure安装
Allure使用
Allure详解
Title
Description
Feature
Story
Severity
Allure.dynamic
Allure应用
应用测试用例
项目运行

Allure

快速入门 - 安装

python插件

- 命令安装

  pip3 install allure-pytest
- 源码安装

  https://pypi.org/project/allure-pytest

allure工具

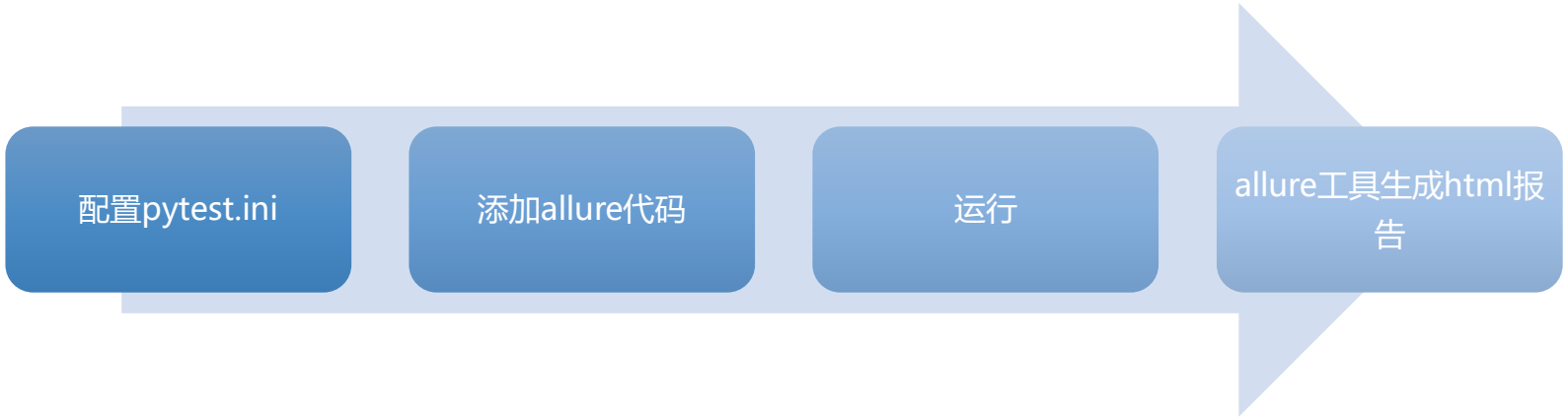
- 下载

  https://bintray.com/qameta/generic/allure2
- 前置条件

  已部署java环境
- 解压并配置系统环境变量

Allure

快速入门 - 使用



Allure

详解

1. Title
   - 可以自定义用例标题，标题默认为函数名
2. Description
   - 测试用例的详细说明
3. Feature
   - 定义功能模块，往下是Story
4. Story
   - 定义用户故事
5. Severity
   - 定义用例的级别，主要有BLOCKER,CRITICAL,MINOR,NORMAL,TRIVIAL等几种类型，默认是NORMAL
6. Allure.dynamic
   - 动态设置相关配置

Allure

Allure应用

Allure应用

博学谷
www.boxuegu.com

自动生成测试报告

Subprocess介绍

- 允许你产生新的进程，并且可以把输入，输出， 错误直接连接到管道，最后获取结果

官方网站:

- https://docs.python.org/2/library/subprocess.html

方法:

- subprocess.call(): 父进程等待子进程完成，返回退出信息(returncode，相当于Linux exit code)
- shell=True 的用法，支持命令以字符串的形式传入。

Allure报告-总结

重难点

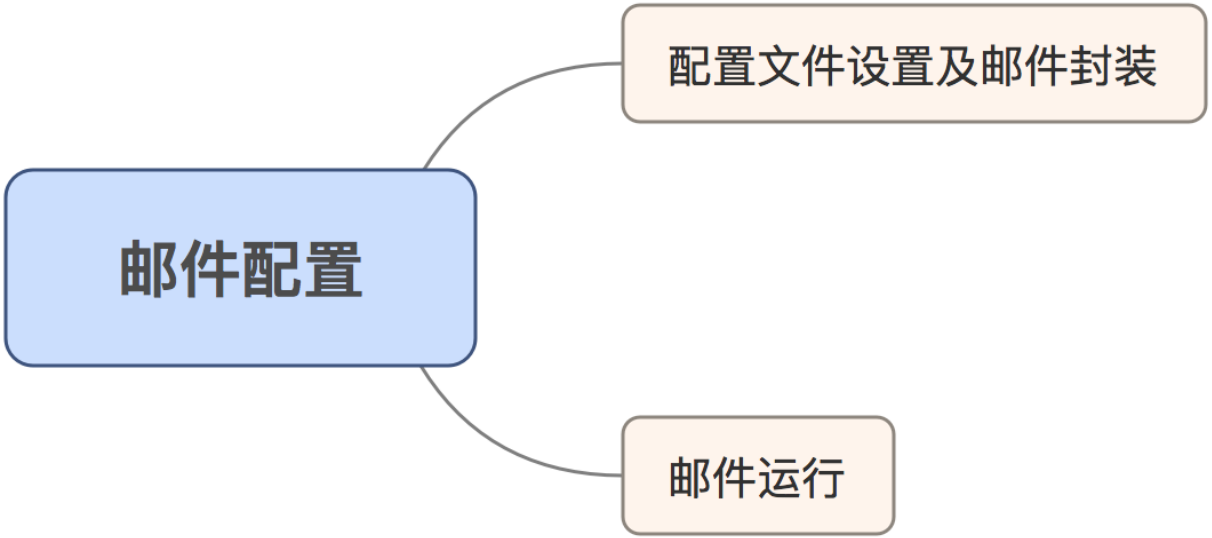
1. allure安装与配置
2. allure相关注解使用

目录

目录 Contents

- 项目及框架搭建
- 接口用例编写
- Requests使用
- 配置文件
- 日志文件
- Pytest框架
- 结果断言
- 数据驱动
- Allure报告
- 邮件配置

邮件配置



邮件配置-总结

重难点

1. 邮件配置

2. 邮件用户名密码要保护好

一样的在线教育，不一样的教学品质