pytest参数化

熟悉unittest单元测试框架的小伙伴知道，使用ddt进行数据驱动测试，那么身为功能更加强大且更加灵活的Pytest框架怎么可能没有数据驱动的概念呢？Pytest使用@pytest.mark.parametrize装饰器来实现数据驱动测试的，也就是常说的参数化。

parametrize语法
parametrize(self,argnames, argvalues, indirect=False, ids=None, scope=None)
参数说明:
argnames：参数名。

argvalues：参数对应值，类型必须为list。如果只有一个参数，里面则是值的列表：

如：@pytest.mark.parametrize("username", ["yy", "yy2", "yy3"])。如果有多个参数，则需要用元组来存放值，一个元组对应一组参数的值，如：@pytest.mark.parametrize("name,pwd", [("yy1", "123"), ("yy2", "123"), ("yy3", "123")])。

indirect：如果设置成True，则把传进来的参数当函数执行，而不是一个参数。

ids：用例的ID，传一个字符串列表，用来标识每一个测试用例，自定义测试数据结果，增加可读性。

1.单个数据

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
#====#====#====#====
import pytest
data=["小明","小花"]
@pytest.mark.parametrize("name",data)
def test_dome(name):
    print("test_dome")
    print(name)


if __name__=='__main__':
    pytest.main(['13pytest参数化.py','-s'])
```

2.一组数据

a.列表嵌套字典

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
```

```
#====#====#====#====
import pytest
data=[
    {"username":"admin1","passwd":"123"},
    {"username":"admin2","passwd":"321"}
]
@pytest.mark.parametrize("name",data)
def test_dome(name):
    print("test_dome")
    print(name)


if __name__=='__main__':
    pytest.main(['13pytest参数化.py','-s'])
```

b.列表嵌套列表

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
#====#====#====#====
import pytest
data=[
    ["maker","123"],
    ['maker2','222']
]
@pytest.mark.parametrize("name,passwd",data)
def test_dome(name,passwd):
    print("test_dome")
    print(name)
    print(passwd)


if __name__=='__main__':
    pytest.main(['1pytest参数化.py','-s'])
```

c.列表嵌套元组

```
import pytest
data=[
    ("maker","123"),
    ("maker2","222")
]
@pytest.mark.parametrize("name,passwd",data)
def test_dome(name,passwd):
    print("test_dome")
    print(name)
    print(passwd)


if __name__=='__main__':
```

```
    pytest.main(['1pytest参数化.py','-s'])

3.使用场景
a.修饰函数时,往函数内传递数据,如果上面的例子
b.修饰类时,往类内的成员函数传递数据
import pytest
data=[
    ("maker","123"),
    ("maker2","222")
]
@pytest.mark.parametrize("name,passwd",data)
class Testmaker():
    def test_maker(self,name,passwd):
        print("test_maker")
        print(name,passwd)

    def test_maker2(self,name,passwd):
        print("test_maker2")
        print(name,passwd)
#注意,如果修饰类,那么类中的所有成员函数都必须接受数据,不然报错

if __name__=='__main__':
    pytest.main(['1pytest参数化.py','-s'])

4.多个参数化装饰器,修饰一个函数
import pytest
data=[
    ("maker","123"),
    ("maker2","222")
]
mydata=['1111','2222']

@pytest.mark.parametrize("name,passwd",data)
@pytest.mark.parametrize("myname",mydata)
class Testmaker():
    def test_maker(self,name,passwd,myname):
        print("test_maker")
        print(name,passwd)
        print(myname)

#注意:测试用例会被调用多装饰器参数中的数据相除叠加的次数,如果上面的就是4次
test_maker
("maker","123"),
1111
test_maker
("maker","123"),
2222
test_maker
("maker2","222")
1111
test_maker
("maker2","222")
2222
```

```

if __name__=='__main__':
    pytest.main(['1pytest参数化.py','-s'])

5.标识每个测试用例
import pytest
data=[
    (10,20,30),
    (40,50,90)
]

ids=["a:{}+b:{}=expect:{}".format(a,b,expect) for a,b,expect in data]

def add(a,b):
    return a+b

@pytest.mark.parametrize("a,b,e",data,ids=ids)
def test_maker(a,b,e):
    print("test_maker")
    print("a=%d,b=%d,e=%d"%(a,b,e))
    assert add(a,b)==e



if __name__=='__main__':
    pytest.main(['1pytest参数化.py','-s'])
```

Allure介绍

```
在当前市面上所有第三方或者自研的测试报告系统中，Allure 是最全面，且支持的测试框架最多的一个测试报告系统。它是开源的测试报告框架，它旨在创建让团队每一个人都清楚明了的测试报告。

```

Allure安装

```
1.Allure 的安装和配置
在该网站下载最新版本或使用我提供的
https://repo.maven.apache.org/maven2/io/qameta/allure/allure-commandline
解压文件，获取到allure-2.22.0文件夹，把该文件夹的bin添加到环境变量中

2.安装allure-pytest
你也可以直接通过如下方式安装：
1.pycharm的设置中下载

2.终端中:pip install allure-pytest

注意，如果你安装过 Allure 2.0 之前的版本，你需要先将之前的版本卸载。

```

Allure使用

```
代码:
#!/usr/bin/env python
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
#====#====#====#====
import allure

@allure.feature("功能名称")
def test_01():
    print("test_01")
    assert 1

@allure.story("子功能名称")
def test_02():
    print("test_02")
    assert 1

@allure.step("步骤细节")
def test_03():
    print("test_03")
    assert 1




第一步:
生成json文件
终端运行:pytest 文件名.py --alluredir=json文件存储的位置 --clean-alluredir
说明:
--alluredir:指定json文件存储的位置,如果有这个文件夹,那么就直接存储,如果没有这个文件夹,就生成这个文件夹,然后再存储
--clean-alluredir:清除上一次的文件

第二步:
第一种方式打开测试报告
allure serve ./生成的json文件夹
这时会调用系统默认的浏览器,打开页面
注意:如果pycharm用不了allure,那么就要进入生成json文件夹的上一层目录中,打开cmd,输入命令

第二种方式打开测试报告
1.把json文件生成为html文件
生成html报告:allure generate ./json的文件夹 -o ./html文件存储的位置 --clean
html文件存储的位置:如果有这个文件夹就把html文件直接存储到这个文件,如果没有就生成这个文件夹,然后在把生成的html文件存储到这个文件夹
2.打开html报告:allure open -h 127.0.0.1 -p 8883 ./生成html文件夹的名字

```

Allure实战

```
1.功能上加@allure.feature("功能名称")
2.子功能上加@allure.story("子功能名称")
3.步骤上加@allure.step("步骤细节")
4.联测试用例（可以直接给测试用例的地址链接）
@allure.testcase("https://www.baidu.com","测试用例链接")

#!/usr/bin/env python
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
#====#====#====#====
import allure
import pytest
import time
from selenium import webdriver

@allure.testcase("https://www.baidu.com")
@allure.feature("百度搜索")
@pytest.mark.parametrize("data",["allure",'pytest','unittest'])
def test_maker(data):
    with allure.step("打开网页"):
        dr=webdriver.Firefox()
        dr.get("https://www.baidu.com")
        dr.maximize_window()
        time.sleep(2)

    with allure.step(f"输入搜索词:{data}"):
        dr.find_element_by_id("kw").send_keys(data)
        time.sleep(2)
        dr.find_element_by_id('su').click()
        time.sleep(2)

    with allure.step("保存图片"):
        dr.save_screenshot("./baidu.png")
        allure.attach.file("./baidu.png",attachment_type=allure.attachment_type.PNG)

    with allure.step("关闭浏览器"):
        dr.quit()




生成json文件
终端运行:pytest 文件名.py --alluredir=json文件存储的位置 --clean-alluredir

打开测试报告
allure serve ./生成的json文件夹
```

```
作业:
在京东首页输入,软件测试,自动化测试,接口测试,使用allure生成测试报告,然后打开
```

接口测试概述(重点)

```
1.什么是接口?
    生活中的接口:插座,usb接口,网线接口,鼠标键盘接口
    测试中的接口:api接口(函数,重点),GUI接口(图形用户界面,如有些软件有微信登录,QQ登录,支付宝登陆)
2.接口测试:
接口测试是测试系统组件间接口的一种测试。
接口测试主要用于检测外部系统与系统之间以及内部各个子系统之间的交互点。

主要的测试内容是:
1.检查数据的交换
2.数据的传递和控制管理过程
传递:要传3个参数,但我就传2个参数,会怎么样?
3.系统间的相互逻辑依赖关系
比如:登录(输入用户名和密码)->服务器中的验证<-数据库
```

为什么要做接口测试及适用环境和好处

```
1.为什么要做接口测试
可以提效率,降成本,自动化,
接口测试是一个完整的体系，也包括功能测试、性能测试、安全等


2.接口测试适用环境
多系统,为其他系统提供服务
平台越复杂，系统越庞大，接口测试的效果越明显

3.进行接口测试的好处
①可以发现很多在页面上操作发现不了的bug
②检查系统的异常处理能力
③检查系统的安全性、稳定性
④前端随便改，接口测好了，后端不用变
```

接口测试的目标和分类

```
1.接口测试的目标:
稳定,持续,提效率和体验,降成本

2.接口测试的分类:
```

```
1.业务功能的测试:和手工功能测试一样
2.边界分析测试:比如手机缴费,金额这个是以一个参数传递了服务器中,参数我是1千万行不行,我交100.0001?
3.参数组合测试:正交法,选一部分合适的
4.异常情况测试:归纳为业务中的异常场景
5.性能测试:响应的时间
6.安全测试:没有加密就传递过去了,如密码.


接口测试就是通过测试不同情况下的输入参数与之相应的输出参数信息来判断接口是否符合或满足相应的功能、性能、安全性要求。
与界面处功能测试相比：
接口测试没有页面；
它是通过接口规范文档上的调用地址、请求参数(请求的方法、请求头部
、数据)，进行请求信息拼接；
然后发送请求，检查返回结果；
只需测入参（请求）和出参（响应）就行
```

接口文档的阅读和分析(重点)

```
接口文档应该包含的内容
1  接口说明
2  调用url
3  请求方法（get/post）
4  请求参数、参数类型、请求参数说明
5  返回参数说明

接口文档的获取
1.标准化的接口文档
2.询问开发人员,一般这个接口文档也是开发写的
3.测试人员自己抓包获取数据和信息

```

接口测试必备的常用知识

```
1.常见的接口传输协议
http/https
ftp
jdbc --java连接数据库的协议
...

2.常见的接口数据组织形式
html
json
String
XML --标记语言,和html类似,后面详细讲

```