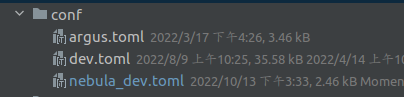
# 01 configObj配置文件对象

# configObj概述

* configObj是配置文件的对象, 用于在代码开发中, 获取\*.toml配置文件中的内容.
* 在apitest中, 每次执行前,会指定唯一的一个toml配置文件, 将其加载, 转化成configObj
* 在测试开发中, 可能会用到的常量数据, 可以抽出来, 存储在toml配置文件中,比如url信息, 用户信息,设备信息
* 根据被测环境的不同, 比如线上环境或线下环境, 可以灵活的切换指定的配置文件, 而不需要频繁的修改代码

# 配置文件 Toml

配置文件存放目录apitest/conf/  如下图:



在配置文件中, 我们可以根据需要, 添加配置数据

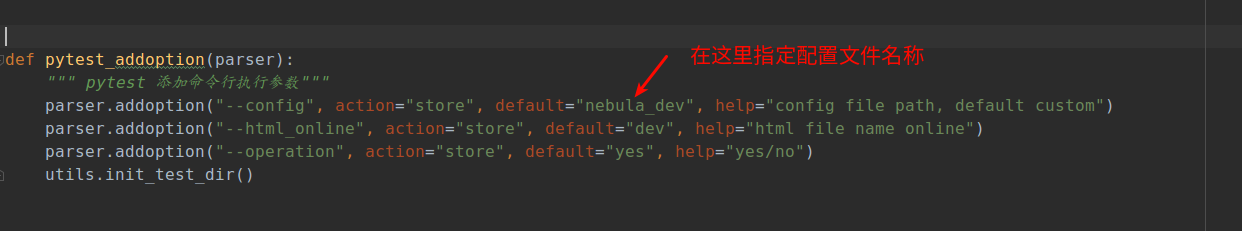
# 指定配置文件名称

方法一: 命令行添加参数–config, 如下:

pytest --config=nebula\_dev

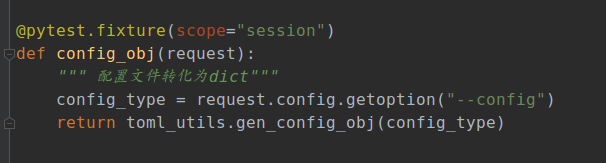
方法二: 代码中修改(适用于使用ide开发过程中修改配置文件)

apitest/conftest.py::pytest\_addoption函数, 如下图:



# configObj的作用范围

configObj在pytest里被定义为一个fixture, 其名称为config\_obj, 它是在全局conftest文件中定义的,且scope为session级别, 因此**在conftest和测试用例中皆可使用该fixture**



# configObj使用举例

配置文件如下:

[ConfigMap]

client\_info = [

'Clients.Aiot.ec1', #205

# 'Clients.Aiot.ec2', #206

]

camera\_info = [

'Clients.SubDevice.Camera\_hikvision',

# 'Clients.SubDevice.NVR2'

]

[EnvInfo]

[EnvInfo.NebulaCloud]

Service = "http://nebula.staging.sensetime.com"

[Clients]

Service = 'https://%s:5043'

[Clients.Aiot] # aiot的端

[Clients.Aiot.ec1]

ip = '10.151.5.205'

port = '22'

[Clients.Aiot.ec2]

ip = '10.151.5.206'

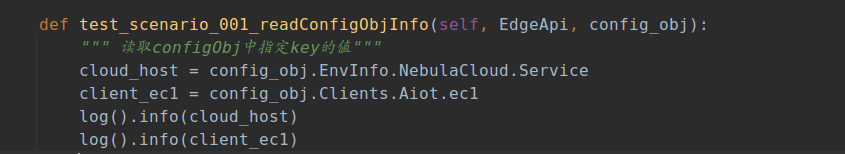
port = '22'

(...略)

案例1: 在项目级conftest中使用,如下图:



案例2: 在测试用例中使用,如下图:



# configObj进阶用法: ConfigMap

在以往的测试中,遇到过这样的参数化测试的场景:

        验证范围包含: 不同型号的端设备(型号类型数量设为N), 以及不同品牌的摄像头设备(品牌数量设为M)

如果想把所有类型都验证一遍的话, 需要运行测试用例N\*M次

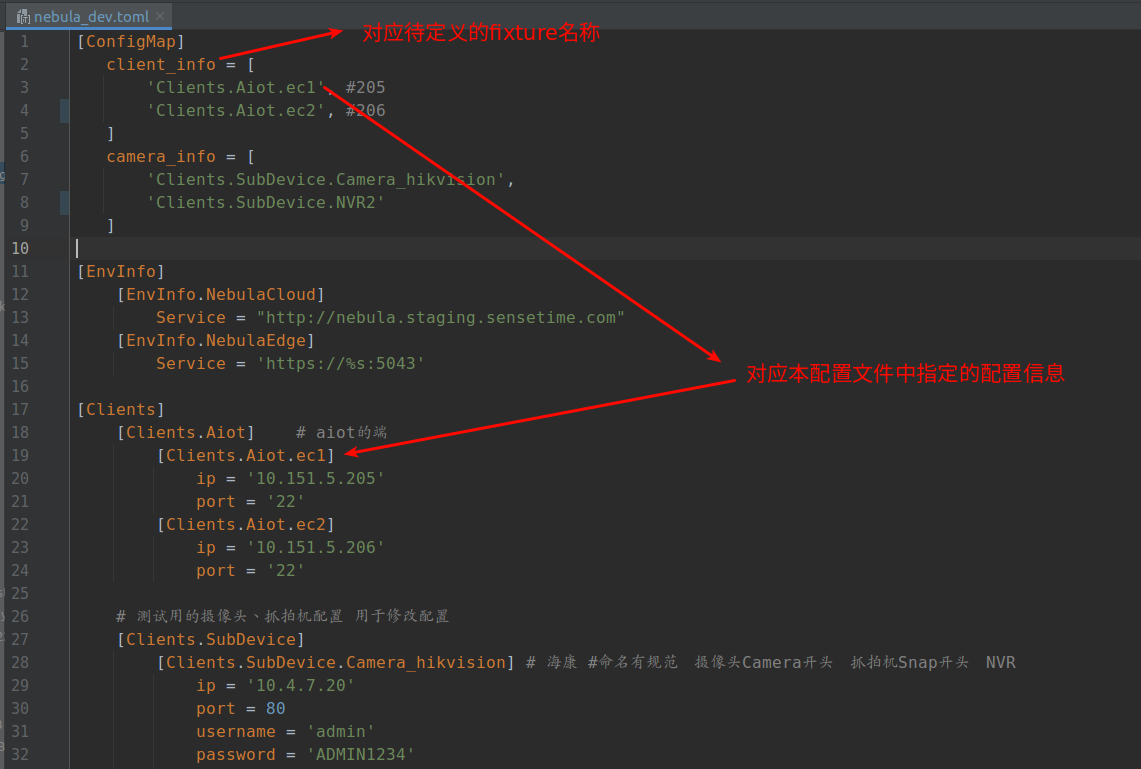
当时的做法是使用pytest的参数化装饰器(@pytest.mark.parametrize)实现,

缺点是需要在每个测试用例上都添加参数化装饰器, 并且写死在测试用例中.

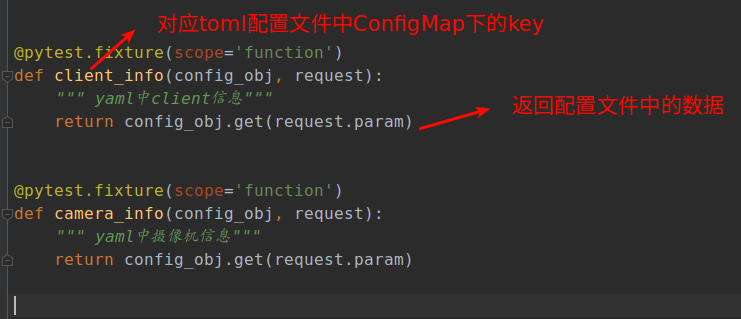
后来, 我思索出了一种解决方法:

基于configObj 以及pytest提供的动态参数化方法, 简化的这一流程,  不再需要在代码中添加过多的装饰器代码,  只需2步即可实现以上场景的测试

第一步: 在toml配置ConfigMap, 如下图:

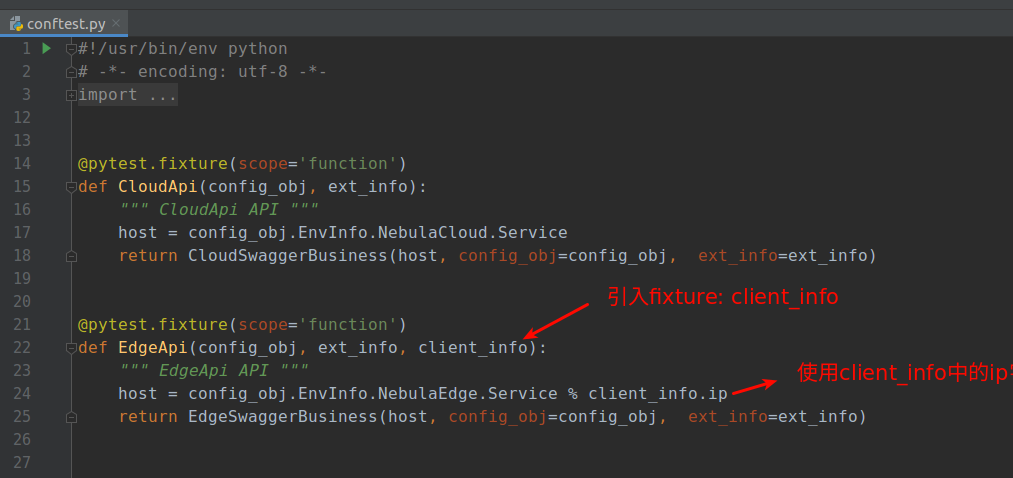


第二步: 为ConfigMap下的key, 定义相应名称的fixture, 建议在项目级conftest文件中定义.如下图:

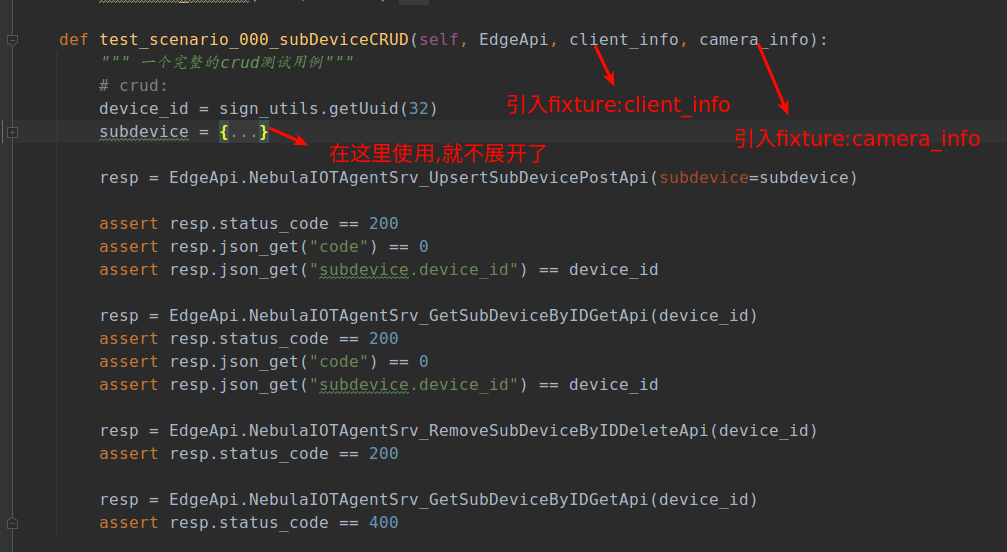


在代码中应用:

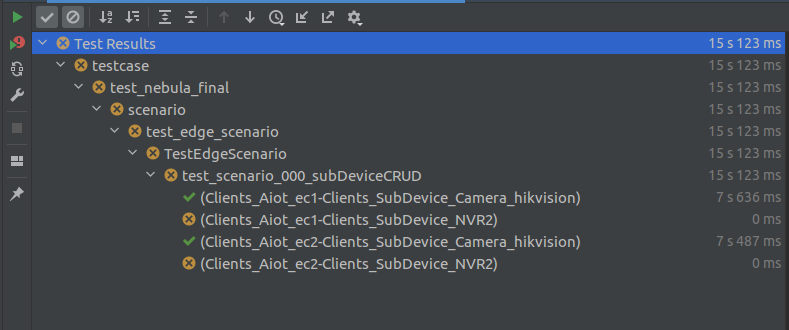
举例1: 在conftest中应用, 为端接口对象提供端的ip信息, 如下图:



举例2:在测试用例中应用, 实现前文中提到的场景, 测试不同设备, 不同摄像头下子设备的crud过程:



该例子运行结果如下, 可以看出, 在2个设备, 2个摄像头的情况下执行了4次



至此, configObj的说明结束