04 接口参数边界值测试用例开发

回顾

根据前文的步骤，我们已经完成了以下几个步骤:

- 生成了大部分代码, 拷贝到指定的目录下
- 配置了conf下的toml配置文件, 填入了一些数据参数
- 完善项目级conftest的代码, 使接口对象可以正常调用

接下来, 就可以正式开始测试

脚本开发

首先,我们先分析下参数测试用例生成的脚本中包含哪些内容, 如下图:

```python
    @pytest.mark.parametrize("invalidParam", [
        ('subdevice', 'invalidsubdevice'),
        ('subdevice', ''),
        ('subdevice', None),
        ('skip', 'invalidskip'),
        ('skip', ''),
        ('skip', None),
    ])
    def test_api_NebulaIOTAgentSrv_UpsertSubDeviceInvalidParam(self, invalidParam, config_obj, EdgeApi):
        """  UpsertSubDevice 更新/添加子设备, device_id不存在时添加，存在时更新。
该... """
        subdevice = None
        skip = None
        intef = EdgeApi.NebulaIOTAgentSrv_UpsertSubDevicePostApi(subdevice=subdevice, skip=skip, sendRequest=False)
        intef.update_body(invalidParam[0], invalidParam[1])
        resp = intef.request()
        assert resp.status_code != 200
```

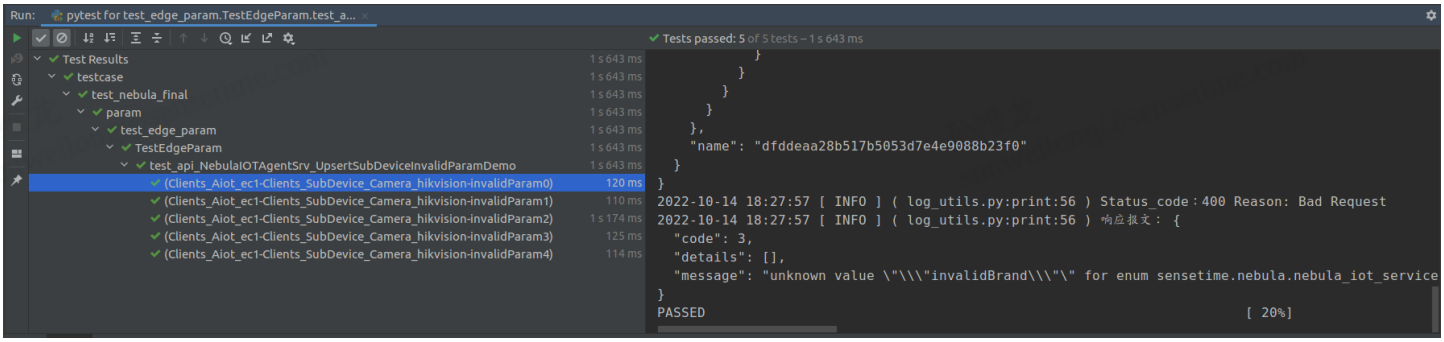
我们还以子设备添加举例:

- 首先可以看到方法定义中的参数中多了invalidParam, 实际是对应装饰器@pytest.mark.parametrize中的第一个参数, 这是pytest参数化的用法.
- 装饰器中的第二个参数是一个列表,每一项是一个元祖, 分别是入参的字段名称,以及测试的边界值
- 代码中,跟api单接口测试相同的是, 需要先填充正确的请求参数, 调用接口方法, 不同点是,接口方法的参数sendRequest=False, 意思是不会真正发送请求, 因为我们需要接下来,对请求报文中的其中一项进行修改,来测试它的边界值
- 通过intef.update_body(invalidParam[0], invalidParam[1]) 对请求报文中的某字段进行修改
- 之后, 调用intef.request()方法发送请求, 得到的resp就可以进行验证了

附上这个例子填充后代码的截图，截图中,我加了一些注释, 方便理解:

```python
# learn_6 边界值测试
@pytest.mark.parametrize("invalidParam", [
    ('subdevice.brand', 'invalidBrand'),
    ('subdevice.brand', ''),
    ('subdevice.brand', None),
    ('subdevice.device_id', ''),
    ('subdevice.device_id', None),
])
def test_api_NebulaIOTAgentSrv_UpsertSubDeviceInvalidParamDemo(self, invalidParam, config_obj, EdgeApi, client_info, camera_info):
    """  UpsertSubDevice 参数测试例子"""
    # 第一步: 构造正常可以通过的接口参数
    device_id = sign_utils.getUuid(32)
    subdevice = {
        "brand": camera_info.brand,
        "device_config": {...},
        "device_id": device_id,
        "extra_config": {...},
        "name": device_id,
        "device_kind": camera_info.kind
    }
    # 第二步: 获取接口对象,传入sendRequest=False,使接口不自动调用
    intef = EdgeApi.NebulaIOTAgentSrv_UpsertSubDevicePostApi(subdevice=subdevice, skip=None, sendRequest=False)
    # 第三步: 更新需要验证的参数, 准则是一次只验证一个参数
    # get, delete 只需要验证param,使用intef.update_params(invalidParam[0], invalidParam[1]) 更新
    # post put, 只需要验证body, 使用intef.update_body(invalidParam[0], invalidParam[1]) 更新
    # 如果对某个参数有特殊要求, 使用如下方法, 进行定制修改
    # if invalidParam[0] == 'subdevice.brand':
    #     intef.update_body(invalidParam[0], invalidParam[1])
    intef.update_body(invalidParam[0], invalidParam[1])
    # 第四步: 调接口,发送请求
    resp = intef.request()
    # 第五步: 验证返回值
    assert resp.status_code != 200
```

执行测试用例后的结果如下:



总结一下, 以上只是抛砖引玉，举了一个简单的例子, 更多的验证方法需要根据需求去写case, 代码写起来是比较灵活的.

这里再说一下, 生成代码这件事, 就是为了减少这些重复代码的编写, 参数边界值测试就是一个很好的例子, 往往为了测边界值, 要改参数,调接口,重复无数次, 而这些过程中,是有共性的,那么就一定能通过测试框架去解决这个问题.

因此, 我是希望测试同学能够写更少的代码, 而进行更多的测试及思考，把更多精力投入到业务逻辑测试中. 保证测试能在短时间内完成,并有很高的成效, 这也是我不断更新测试框架的动力.

至此, 参数边界值接口测试补充完成.