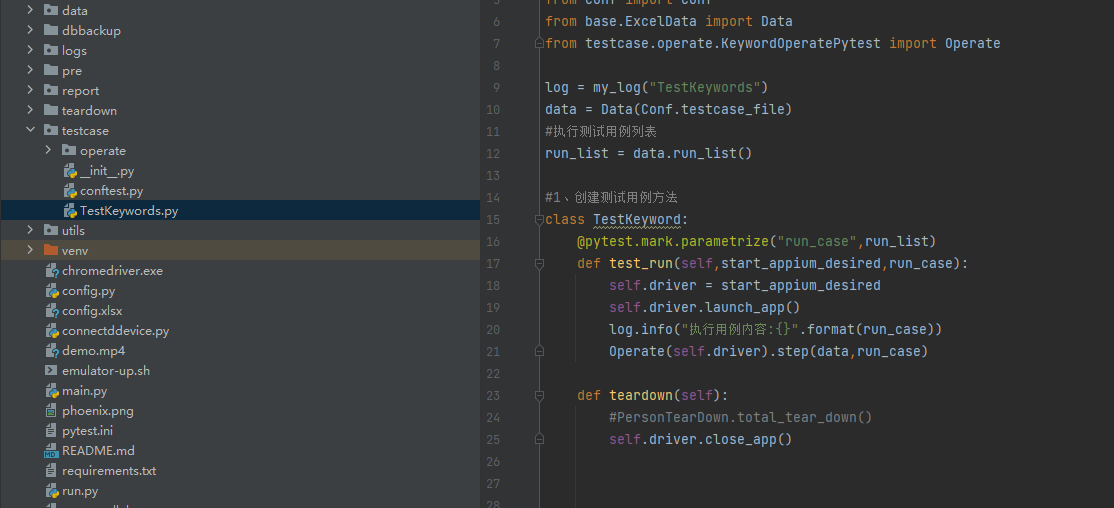
这里主要介绍一下我这个测试框架

1

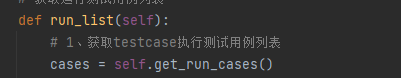


启动传入运行参数

框架会调用testCase层次的以Test打头的文件

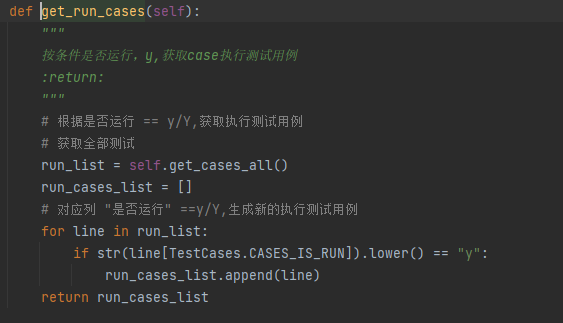


通过data.run\_list()传入读取Excel中的信息作为参数传入

然后调用

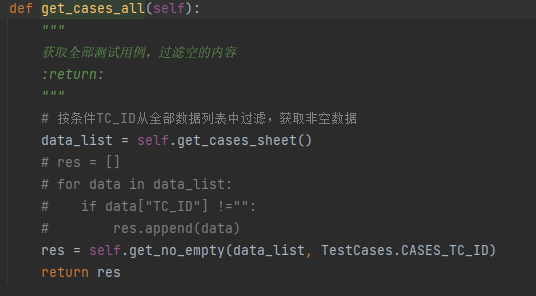
self.get\_run\_cases()

这个方法的逻辑是

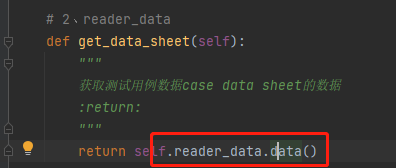


先调用get\_cases\_all()

方法后面判断是否标注需要执行 如果需要执行添加到待执行用例列表中

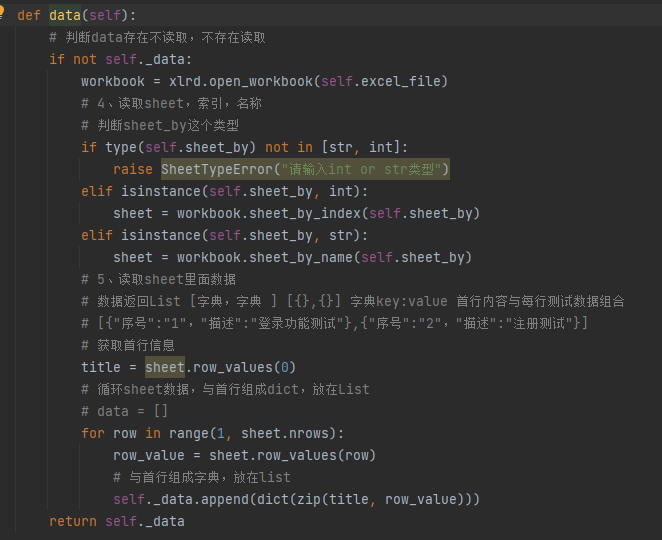


他的逻辑是获取Excel中的数据 通过get\_no\_empty 方法去掉空行



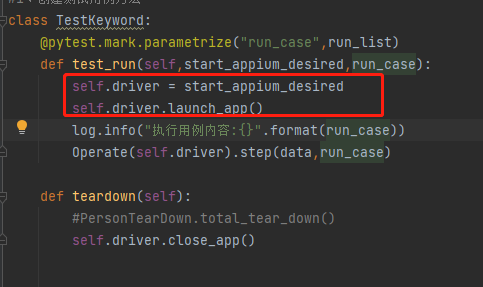
调用reader\_data.data()

这个逻辑是



判断传入的标签页是否是整数或者是文字 如果是文字按sheet\_by\_name定位标签页如果是整数 按照sheet\_by\_index查找如果都不是报错

找到后把数据通过{“字段名”:”字段值”} 方式组装后返回

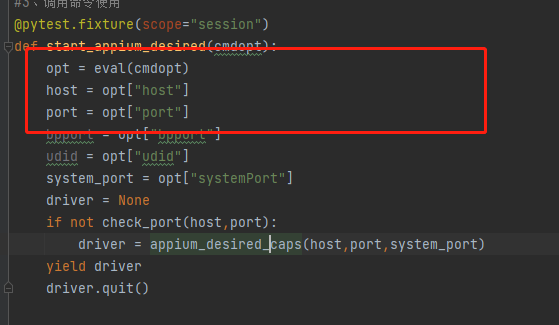


调用启动Appium的方法用于获取Appium Driver 且这个方法是放在conftest 方法中

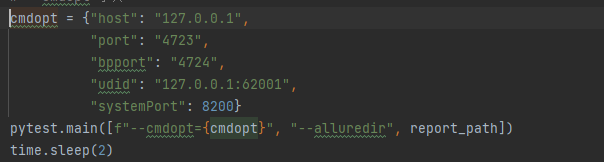
start\_appium\_desired



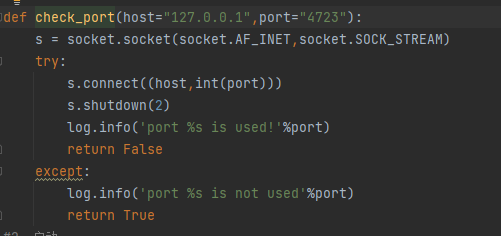
在pytest 框架中conftest专门存放前置流程所需要的方法

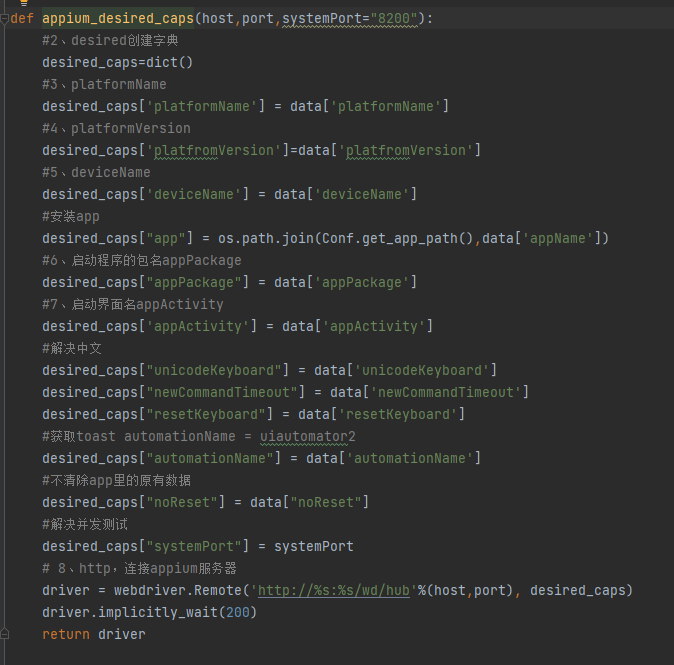


从run里面cmdopt中获取运行参数



Checkport方法用于检查是否端口被占用





通过这个方法传入手机运行的参数 生成Driver

这里就要提一下 怎么搭建Appium测试环境

1 到 <https://github.com/appium/appium-desktop/releases/tag/v1.22.3-4>

下载 Appium Driver

2 安装selenium

安装好python环境后 pip install selenium

3 安装appium\_python\_client

pip install Appium-Python-Client

4 安装appium\_inspector

<https://github.com/appium/appium-inspector/releases>

更多的环境搭建请参阅

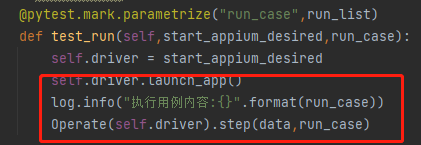
[APP自动化测试神器之Appium-Python环境部署及脚本实战 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/144737398)

特别注意安装好后要配置JAVA\_HOME

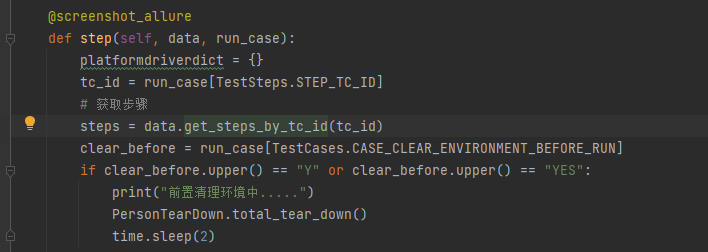
ANDROID\_HOME

至于怎么配置 网上自己百度一下

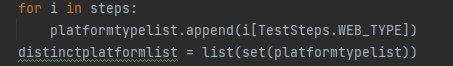
appiumInspector用于元素的识别



然后把driver传入 Operate.step

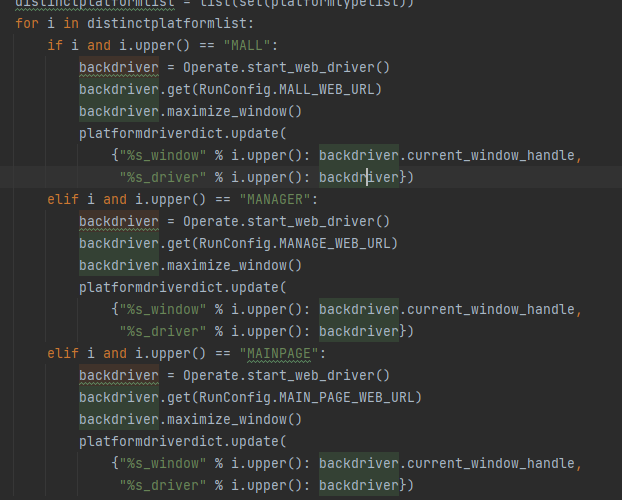


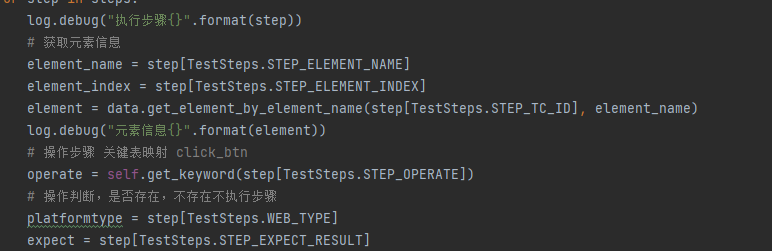
这部分是根据第一个标签页的 TestCaseId 在已经读取出来的TestStep中获取这个用例的步骤并判断是否在执行和这个步骤是否做前置环境清理



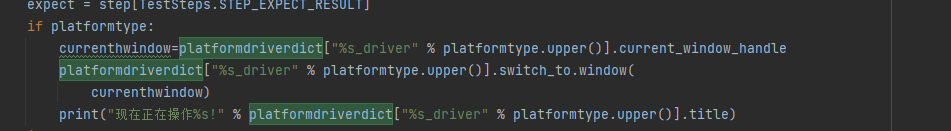
如果需要Web端联测 会标注MALL 表示 商场 MANAGE 管理平台 MAINPAGE表示企业后台

获取后台标签并去重后 如果set后不为空分别调用Web浏览器打开对应的页面并把driver（不同的平台不同的driver避免相互干扰）和对应的窗口句柄以Dict方式保存





根据这段代码获取元素和预期结果和操作类型关键字以及是否为Web端操作，如果需要Web端操作获得需要操作那个后台 并切换到对应窗口 并提示你正在操作那个页面

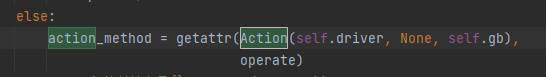


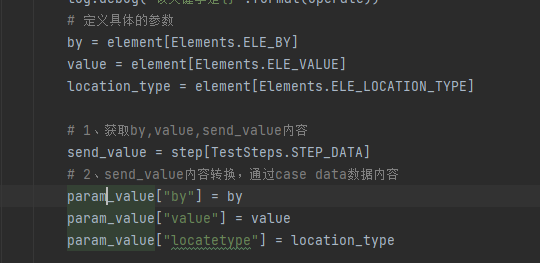


从测试步骤获得发送的JSON

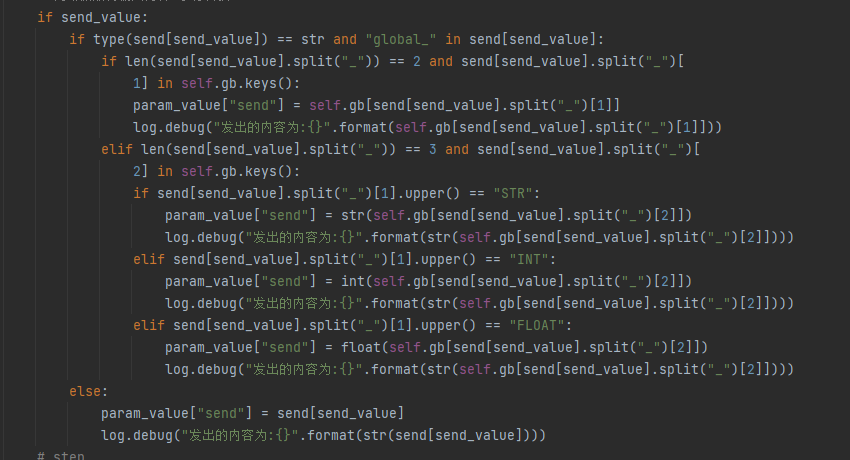
并传入全局变量和JSON 通过str\_to\_dict方法转换为实际的值（支持写死，调用动态方法和全局变量都是通过这个方法）转换为实际值

最终通过Python的getAttr（）反射出operate变量中的方法名称的方法实例

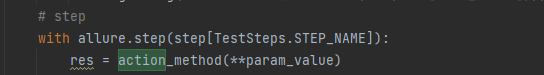




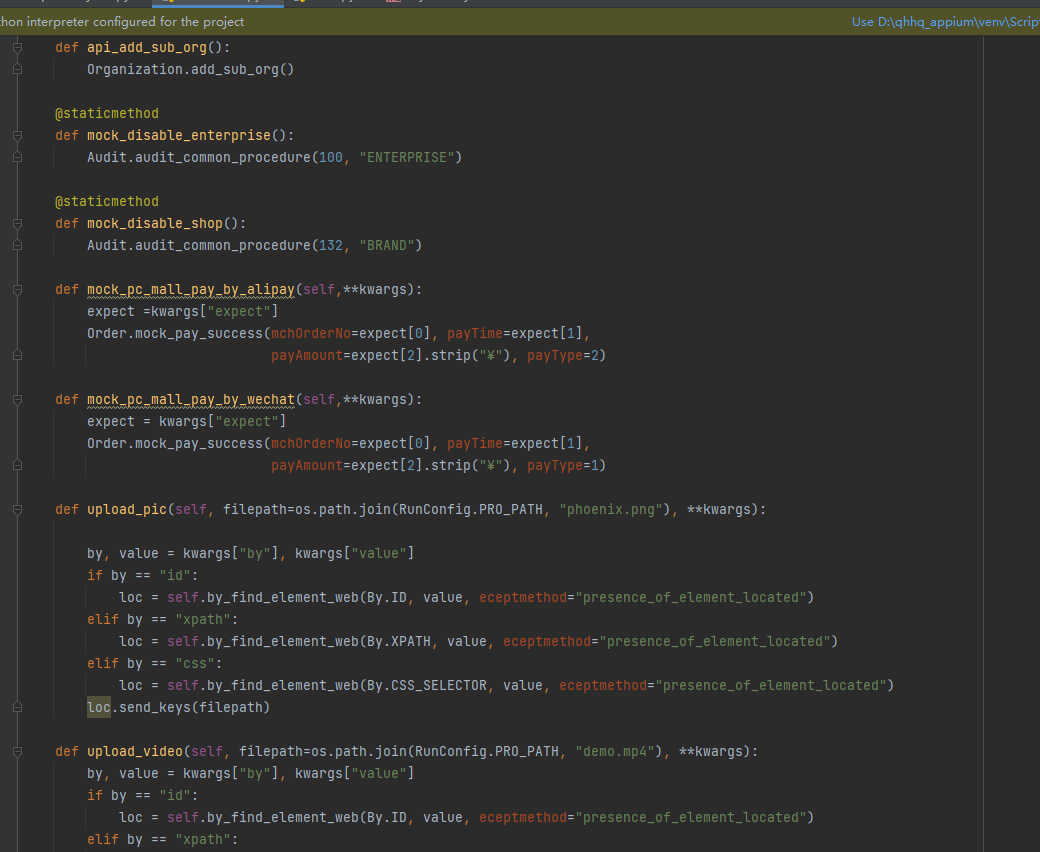
把相关的Element放入param\_value这个dictionary里面



判断是否需要发送信息 如果是并且是从全局变量里面取则处理，否则也把send内容放在param\_value里面



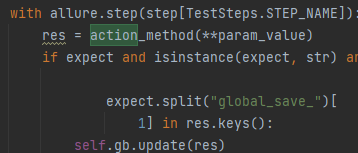
最终把param\_value传入方法的实例



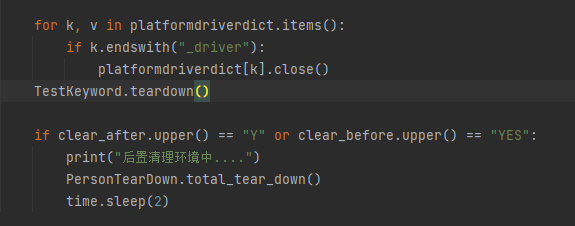
在具体方法在Action中定义会从param\_value 中获取值并调用不同的方法一般后台的方法以\_web结尾（Chrome） App方法不带尾标 API方法会放在



并以api\_打头（调用接口方法以便环境准备或用例前置条件太复杂如果完全通过UI维护成本太高 比如用例的前置条件是要新建然后三级审核如果全部都通过UI走步骤会很长而且相对于接口UI稳定性会更低 所以用例前置条件一般都通过接口准备）



最终把上述获得的param\_value 传入反射出来的方法实例并把结果更新到global中供给后续使用



最后Chrome关闭driver

Appium关闭driver

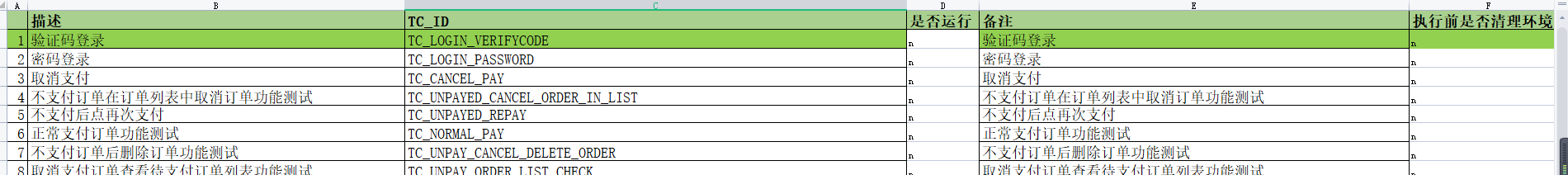
如果需要后置清理 调用一下后置清理流程

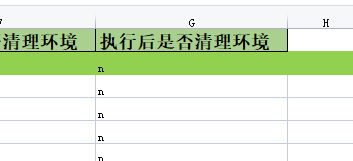
总体就是这么个流程

Excel中分为四个标签页



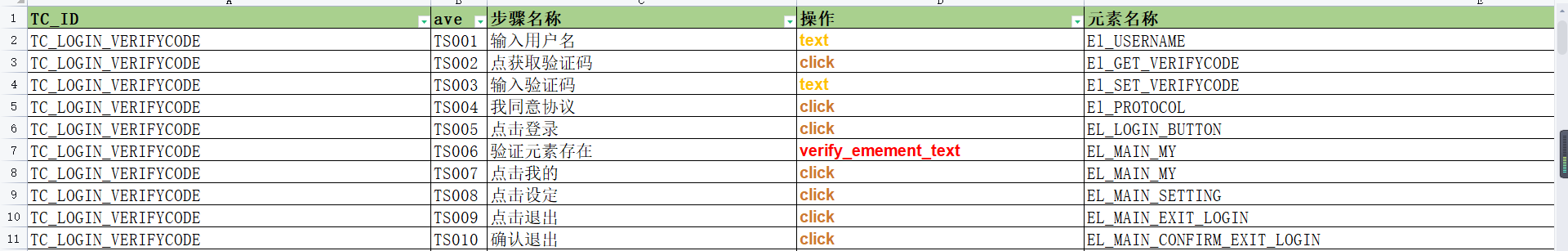
1测试用例

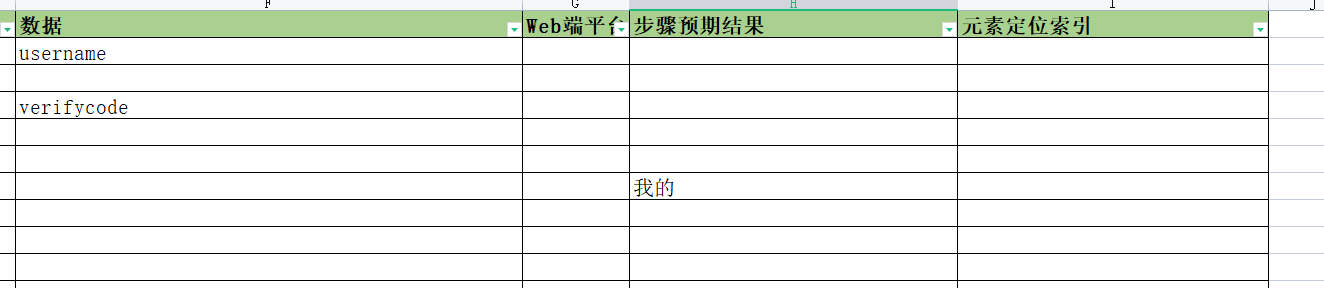




关键字段是TC\_ID 不能重复 以及是否需要前后置环境清理

2 测试步骤



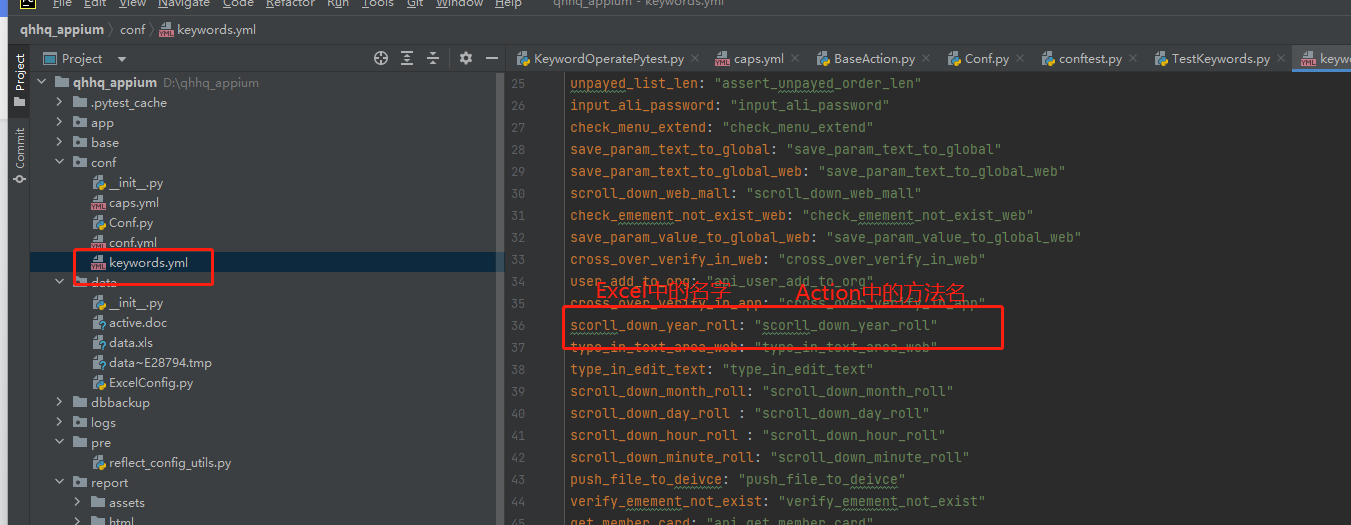


TC\_ID要和第一个标签页的匹配 才能让测试用例和用例步骤关联起来

操作是关键字段 告诉代码你操作步骤要对后面的元素名称所对应的元素干嘛

一般我以\_web结尾表明是后台的操作不带的是APP操作

这个名称通过这个方法



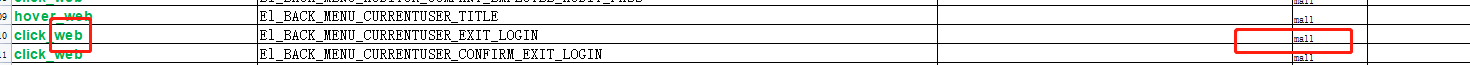
Mapping

并通过



转化为operation 并反射出方法实例

如果是后台操作

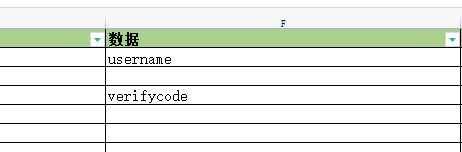


会标注一个平台标识，代码会根据平台标识切换到对应的界面并操作Element指定的页面元素注意带web的元素是在Chrome通过F12 获取的 而不是Appium Inspactor 获取的

验证的方法

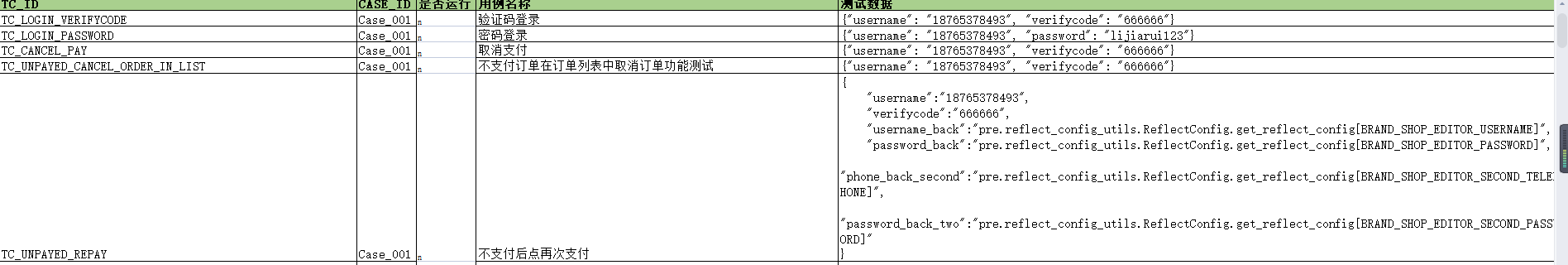


一般都是红色并填写预期结果 同样预期结果也支持写死 动态元素和从上步骤获取的全局变量 往往是上一步APP操作完把页面获取的元素存入全局变量



数据为第三个标签页的dict的key

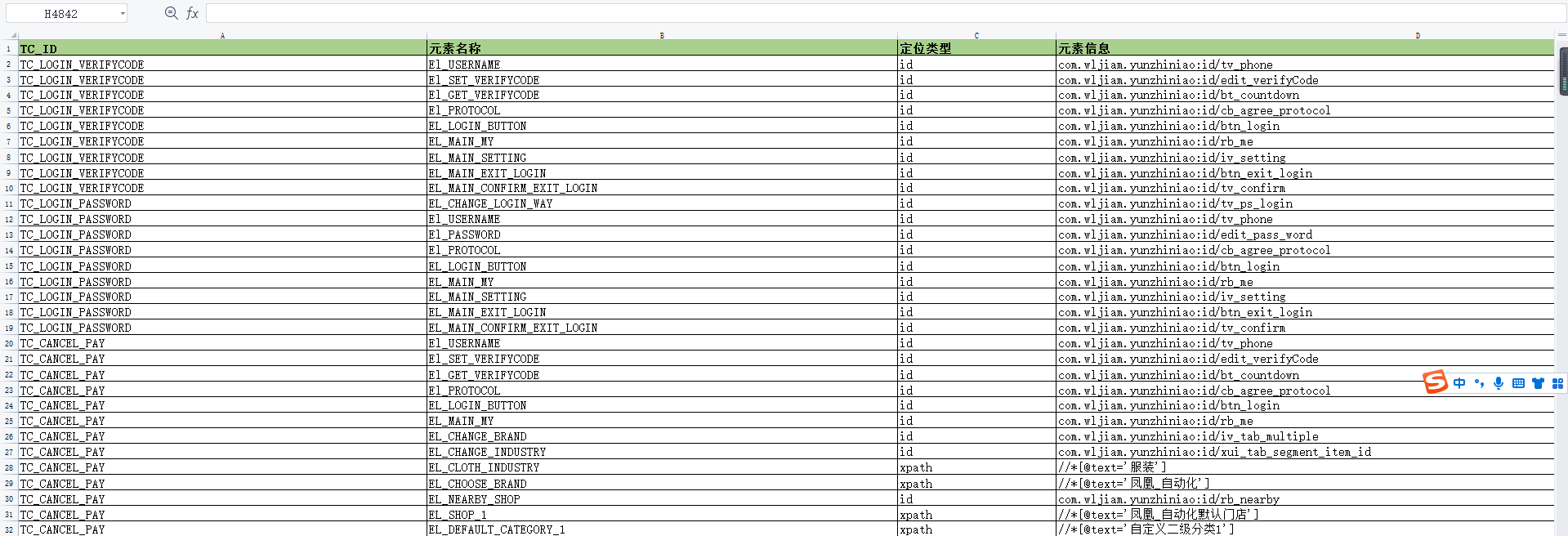
3 请求数据



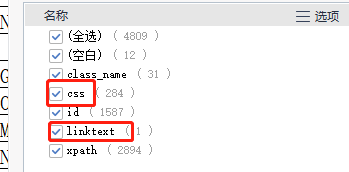
同样TC\_ID是关键字段 是否运行要和第一个标签页是否运行一致，发出的数据通过Dict方式保存value支持固定值，动态方法调用；而且支持上一步骤产生的数据如果要在下一步骤用通过从res更新到send dict后下一步使用



4 元素定位信息



同样最重要的就是TC\_ID字段 表明是哪个用例的Element 元素名称 要对应Step中的元素名称 定位方式支持id ,xpath ,css(仅支持web) ,class\_name ,linktext(仅支持web)



Web端元素识别请参考上节课说的Web端元素的识别 或者参考上机课给个PDF教程

这4个标签页通过TestCaseId贯穿

这里说一下APP端元素的识别

需要注意使用Appium Server

Appium会往你手机里面安装服务端 这个需要Root 权限 如果你的手机在保不建议Root否者会影响保修 可以使用虚拟机方式

建议安装夜神

安装地址https://www.yeshen.com/

安装好后建议选择Andorid7 因为超过9 元素识别你就只能用Appium Inspector 而不能用uiautomatorviewer 因为高版本Android 不支持

以下以淘宝的APP讲解一下怎么样通过Appium进行元素识别

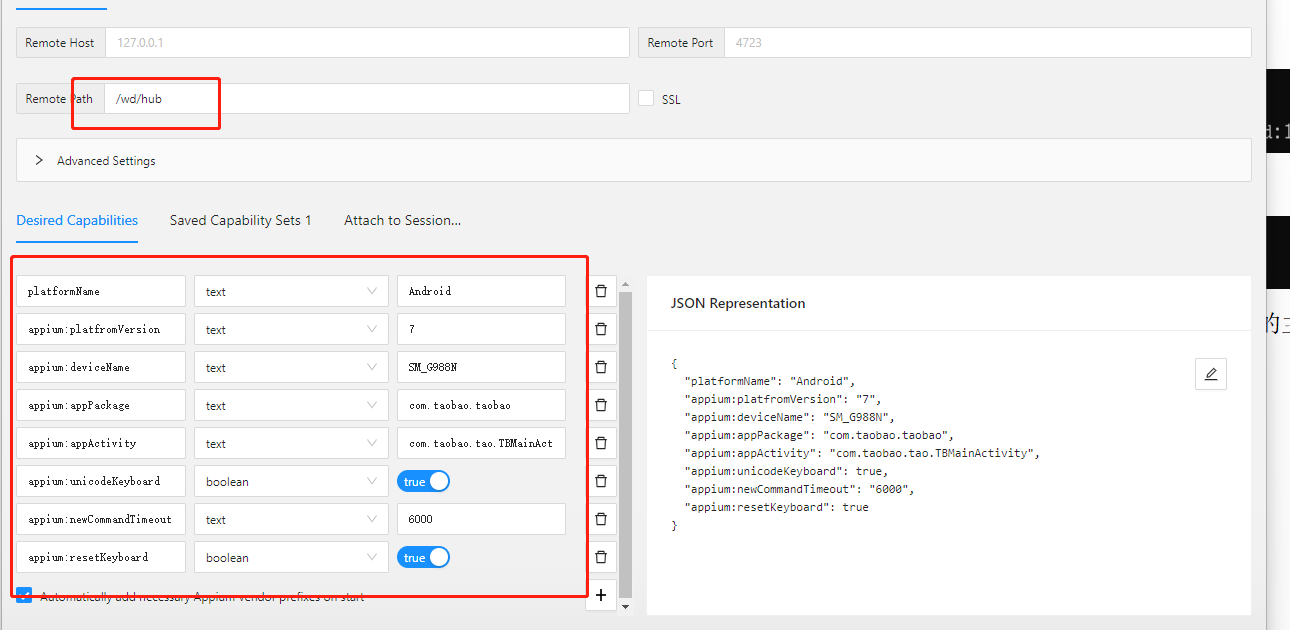
1 查询设备信息



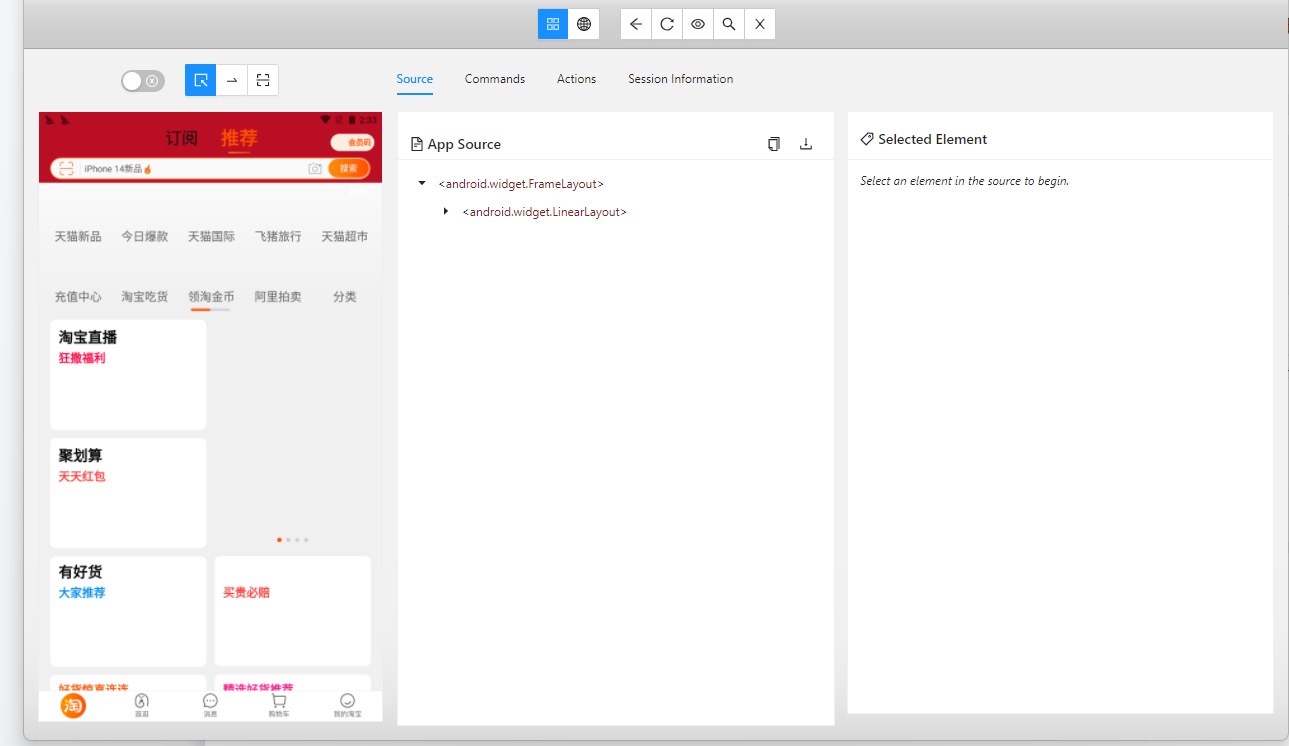
2 查询APP Package



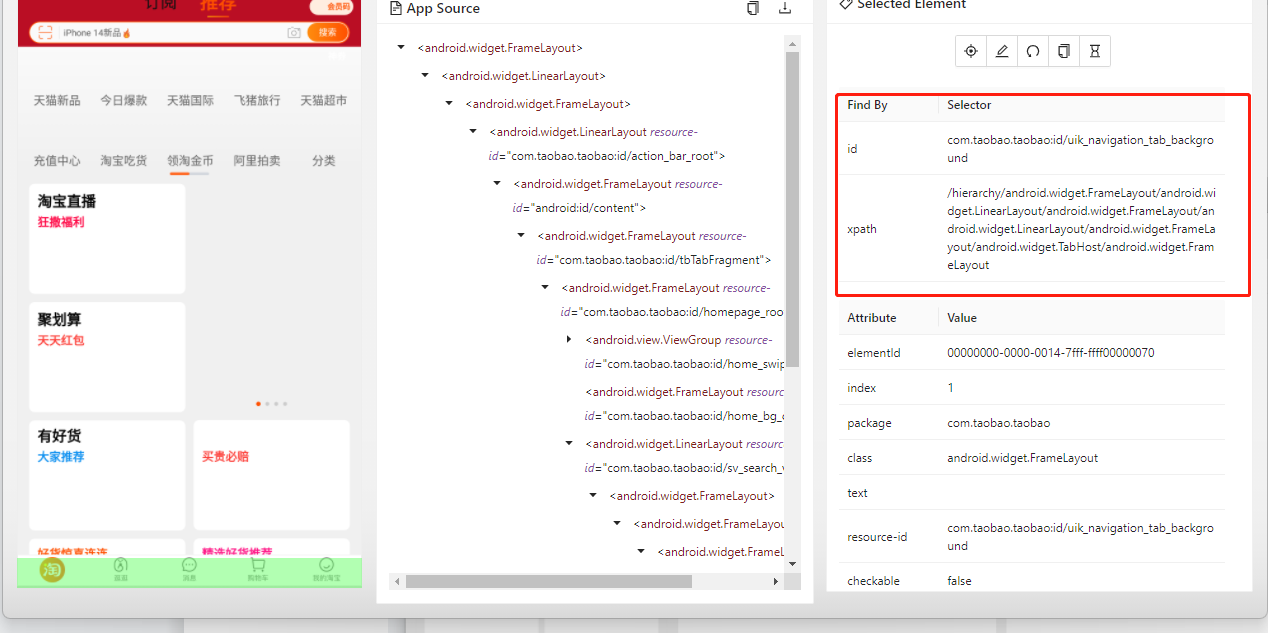
前面的是package name 后面是ActivityName ActivatyName 你可以理解为一个APP的主页等同于Web端的登录界面 一切操作从这个页面开始



点start\_session



比如我的淘宝



Id 就填第一个

Xpath就填第二个