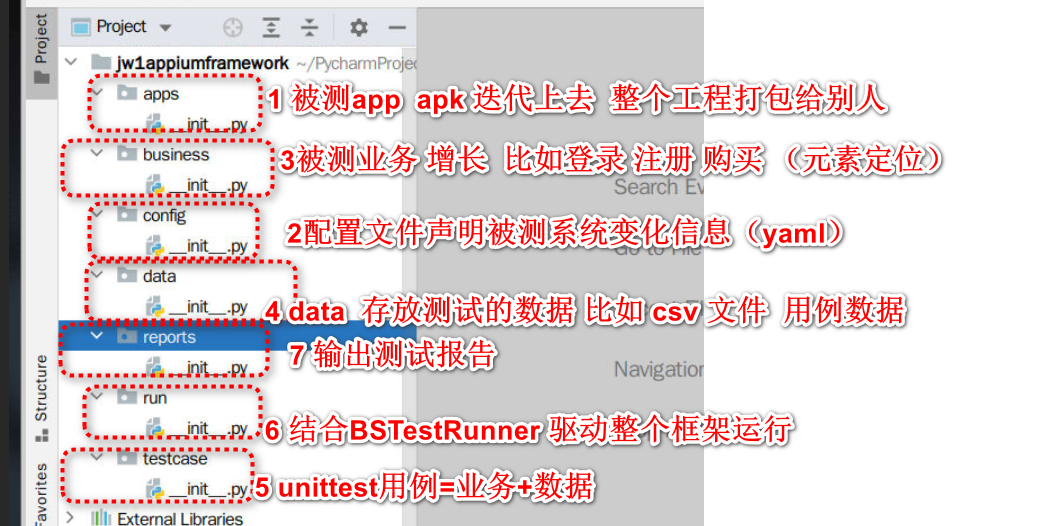
全程87期appium第五次（框架）

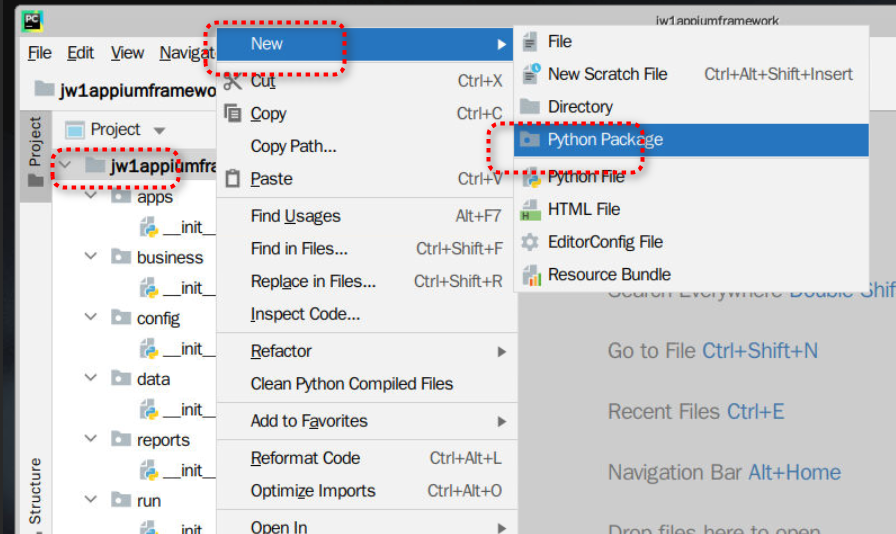
### 内容1： 框架---灵魂三问？

**什么是测试框架？**



框架 实现不同职责包和模块 满足测试需求

包：



**为什么测试框架**？

实际编写代码过程 满足如下需求：

**需求1：**

测试 app或者手机 更换的 Capability 信息频繁更换

解决： 测试项目 加一个文件（最好不是py文件）

定义容易变化配置部分比如 手机信息 app包名 活动名

**===》配置文件（涉及更改 格式 yaml ini 文件 ）**

**需求2：**

组织测试用例 用例中 进行测试断言 完成用例是否通过

解决 ： unittest 和 assert断言

**需求3：**

用例 设计到不同的用例数据 （CSV用例数据 ）

CSV 文件

**需求4 ：图形化报告**

解决 HTMLTestRunner （BSTestRunner）

**需求5： 分工（元素定位专门人 用例人）**

解决 ： 包 不同的人工作成果 放在不同的包

**如何掌握测试框架？**

**appium框架 （小 yaml）**

**1照搬照抄--模仿建议框架思想**

**2 按照包逐步实现框架（分清）**

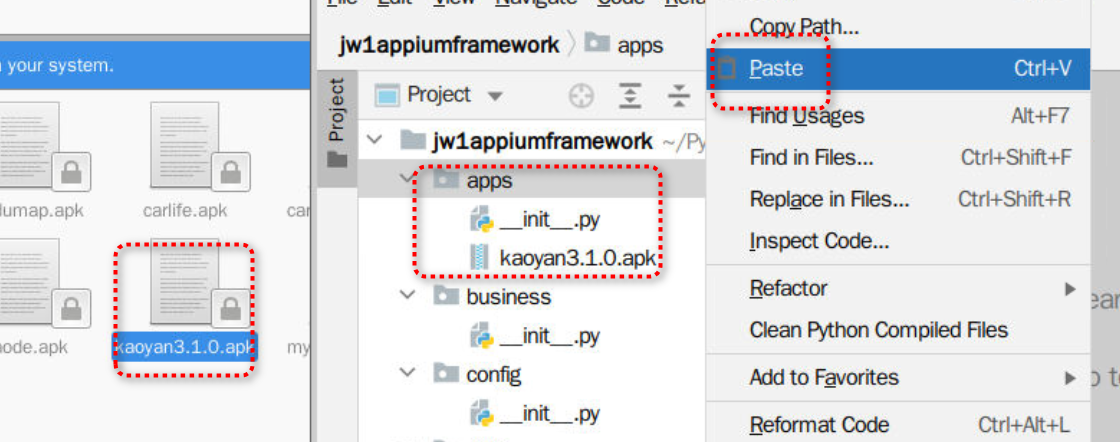
2.1 代码粘贴 （复用工具代码 比如 python 读yaml 文件 csv文件 复用）

2.2 测试代码（比如元素定位业务，测试数据 ，用例，驱动）

**3脱离老师 粘贴一遍 完整运行 +描述框架执行流程即可~**

### 内容2： 按照包逐步实现框架 ：

### 包1： apps 存放被测app （实际上多个版本app）



### 包2： config包

**2.1 yaml 目的区分源代码 py java**

表示信息 键值对表示配置~ github

配置格式语法：

name: 空格 zhangsan

name:

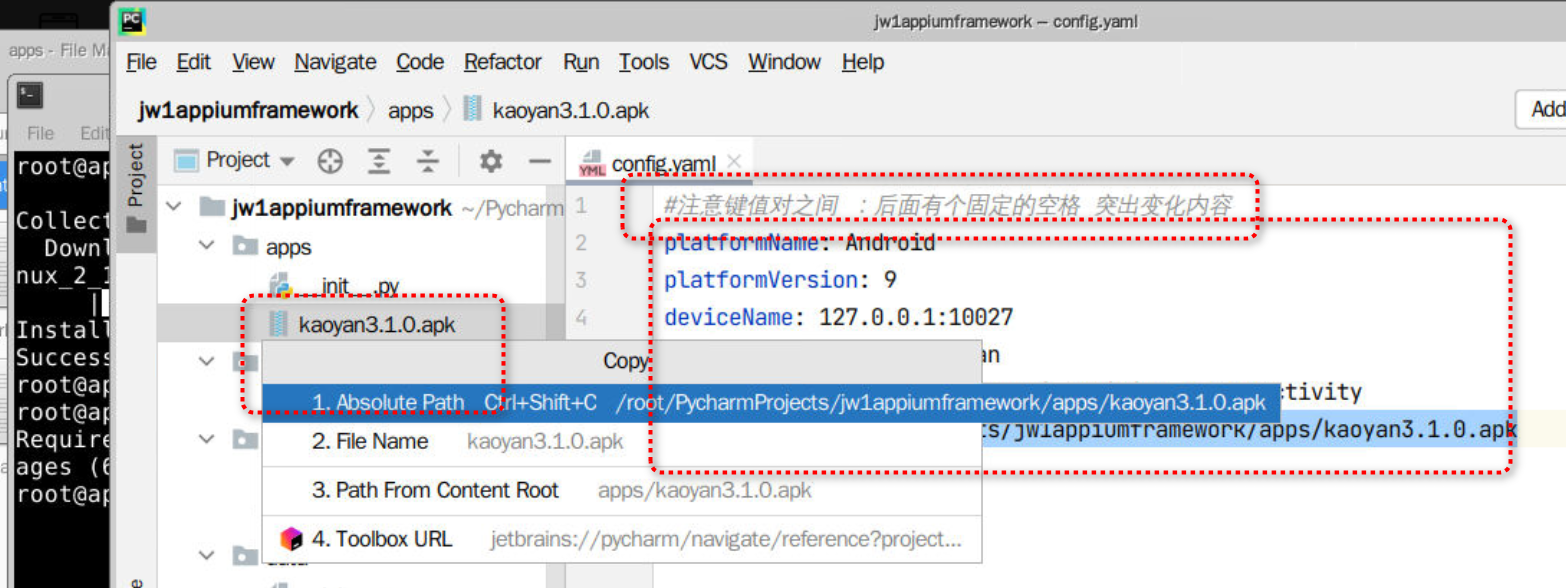
- zhansgan

- xiaojiang

- xiao

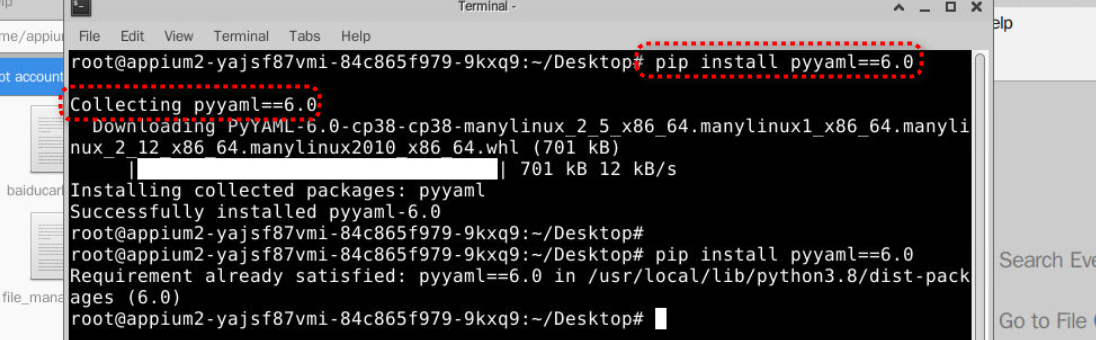
配置文件config.yaml方便以后修改~

|  |
| --- |
| #注意键值对之间 ：后面有个固定的空格  #注意键值对之间 ：后面有个固定的空格 突出变化内容  platformName: Android  platformVersion: 9  deviceName: 127.0.0.1:10027  appPackage: com.tal.kaoyan  appActivity: com.tal.kaoyan.ui.activity.SplashActivity  app: /root/PycharmProjects/jw1appiumframework/apps/kaoyan3.1.0.apk |



**2.2 python 读取yaml 配置信息 ，才可以执行**

需要安装一个库 pyyaml库 API



pip install pyyaml==6.0

**python 定位 yaml 中配置 根据yaml中的配置运行框架**

**还是打开yaml 文件 导入yaml 信息 ，根据键获取值~**

**（不会变化的 --公共使用的工具代码 ）**

|  |
| --- |
| import os  import yaml  from appium import webdriver  def appium\_desired():  # 1.获取appium capabilities  # 获取yaml文件的目录  yaml\_dir = os.path.dirname(\_\_file\_\_)  # 获取yaml文件的完整路径  yaml\_path = os.path.join(yaml\_dir,'config.yaml')  # 打开文件，读取内容  config\_file = open(yaml\_path,'r',encoding='utf-8')  # 解析yaml文件内容  # 说明跑的时候这一步会有警告可以忽略的  #data = yaml.load(config\_file)  # 注意新版 python 3.8 yaml 6.0必须使用如下形式!!!  data = yaml.load(config\_file,Loader=yaml.FullLoader)        desired\_caps = {}  desired\_caps['platformName'] = data['platformName']  desired\_caps['platformVersion'] = str(data['platformVersion'])  desired\_caps['deviceName'] = data['deviceName']  desired\_caps['appPackage'] = data['appPackage']  desired\_caps['appActivity'] = data['appActivity']  desired\_caps['noReset'] = False  desired\_caps['app'] = data['app']  # 中文问题处理  desired\_caps['unicodeKeyboard'] = True  desired\_caps['resetKeyboard'] = True  # 2.启动app(包含安装过程)  driver = webdriver.Remote('http://localhost:4723/wd/hub', desired\_caps)  driver.implicitly\_wait(10)  return driver  def cancel\_click(driver):  cancel\_btn = driver.find\_element\_by\_id('android:id/button2')  cancel\_btn.click()  def skip\_click(driver):  skip\_btn = driver.find\_element\_by\_id('com.tal.kaoyan:id/tv\_skip')  skip\_btn.click() |

小结:价值：

以前一个py 都要做一次 Capability

**现在 知道调用appium\_desired() 根据 yaml 生成了操作手机 的驱动即可~~**

**小结：**

**变化文件----》yaml**

**复用的代码文件 ---》 common.py中**

### 内容3： business 包（为了用例testcase）：

思考： 比如登录 比如 注册

注意 ： 编写业务脚本和 以前API脚本区别

1 考虑复用代码=======》 **appium\_desired()**

**2 业务能 接收用例数据 ====》形式参数 username password**

3 业务实际 结果 ===========》 有了实际结果才能和用例比对出测试结果

4 业务 自测 ========>才能整合

|  |
| --- |
| # 2 函数接受来自于 csv 用例数据  from time import sleep  from appium.webdriver.common.appiumby import AppiumBy  from config.common import appium\_desired  def login\_test(username,password):  # 1 capability 信息====》 appium\_desired()  driver=appium\_desired()  # 3.点击‘取消’按钮  sleep(2)  driver.find\_element(AppiumBy.ID, 'android:id/button2').click()  # 4.点击‘跳过’按钮  sleep(2)  driver.find\_element(AppiumBy.ID, 'com.tal.kaoyan:id/tv\_skip').click()  # 5.输入用户名和密码  # 用户名  sleep(2)  driver.find\_element(AppiumBy.ID, 'com.tal.kaoyan:id/login\_email\_edittext').send\_keys(username)  # 密码  sleep(2)  driver.find\_element(AppiumBy.ID, 'com.tal.kaoyan:id/login\_password\_edittext').send\_keys(password)  # 6.点击‘登录’按钮  sleep(2)  driver.find\_element(AppiumBy.ID, 'com.tal.kaoyan:id/login\_login\_btn').click()  # 7.验证是否登录成功  # 帐号下线提示  try:  element1 = driver.find\_element(AppiumBy.ID, "com.tal.kaoyan:id/tip\_commit")  except:  print("没有账号下线提示")  else:  element1.click()  # 增加新的广告处理  try:  element3 = driver.find\_element(AppiumBy.ID, "com.tal.kaoyan:id/view\_wemedia\_cacel")  except:  print("没有弹出广告提示２")  else:  element3.click()  # 检测是否有我知道了按钮  try:  element2 = driver.find\_element(AppiumBy.ID, "com.tal.kaoyan:id/task\_no\_task")  except:  print("没有弹出广告提示")  else:  element2.click()  # 根据界面是否有‘我’判断登录是否成功  try:  # 寻找下方4个tab中的‘我’  driver.find\_element(AppiumBy.ID, "com.tal.kaoyan:id/mainactivity\_button\_mysefl")  except:  print("登录失败")  return False #实际结果  else:  print("登录成功")  return True  if \_\_name\_\_ == '\_\_main\_\_':  #自测函数 业务编写成功 才能和用例整合～  login\_test("神龙大侠999","ksqb0177375") |

**小结 测试业务代码自己编写的**

**业务被测功能 越来越多 开发功能越来越多**

### 内容4：测试用例 -data包

4.1 csv文件 data.csv 格式 （自己准备）

数据之间,隔开 包含用例数据 +预期结果

|  |
| --- |
| zhangsan9999,12345678,true  zhangsan9999,11111111,false  lisi8888,22222222,false |

4.2 python 读取csv 文件 加载数据 （**工具代码**）

[get\_data.py](https://get_data.py)

|  |
| --- |
| import csv  def get\_csv\_data(csv\_file,line):  jw\_csv\_file= open(csv\_file, 'r', encoding='utf-8-sig')  # 注意读取函数必须是csv  reader = csv.reader(jw\_csv\_file)  # 参数2 :决定了下标位置的开始计数方式  for index, row in enumerate(reader, 1):  if index == line:  print(row)  return row  if \_\_name\_\_=="\_\_main\_\_":  get\_csv\_data("./data.csv",2) |

小结 get\_csv\_data 根据文件名和行号 获取指定行数据到列表

遇到return 导致函数立即结束

### 内容5： 测试用例 --unittest技术

回顾：

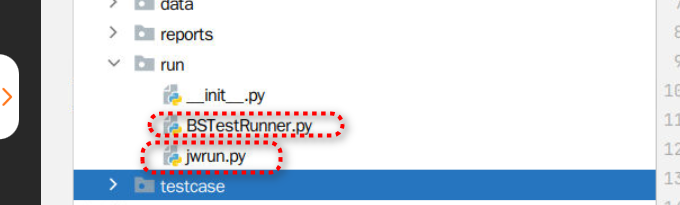
用例（test函数）= 业务函数+csv数据整合

|  |
| --- |
| import unittest  from business.login import login\_test  from data.get\_data import get\_csv\_data  class TestLogin(unittest.TestCase):  # 数据文件位置 属性  jw\_csv\_file = '../data/data.csv'  def test\_login\_111(self):  print("第一条用例")  #用例=csv数据+业务  # csv数据第一行数据 数据文件 行号  data=get\_csv\_data(self.jw\_csv\_file,1)  # 断言 比如实际和预期结果比对产生测试结果～  self.assertTrue(login\_test(data[0], data[1]))  def test\_login\_222(self):  print("第二条用例")  # 用例=csv数据+业务  # csv数据第一行数据 数据文件 行号  data = get\_csv\_data(self.jw\_csv\_file, 2)  # 断言 比如实际和预期结果比对产生测试结果～  self.assertFalse(login\_test(data[0], data[1]))  def test\_login\_333(self):  print("第三条用例")  # 用例=csv数据+业务  # csv数据第一行数据 数据文件 行号  data = get\_csv\_data(self.jw\_csv\_file, 3)  # 断言 比如实际和预期结果比对产生测试结果～  self.assertFalse(login\_test(data[0], data[1]))  if \_\_name\_\_ == '\_\_main\_\_':  unittest.main() |

练习：完成三条用例---3点59分继续~

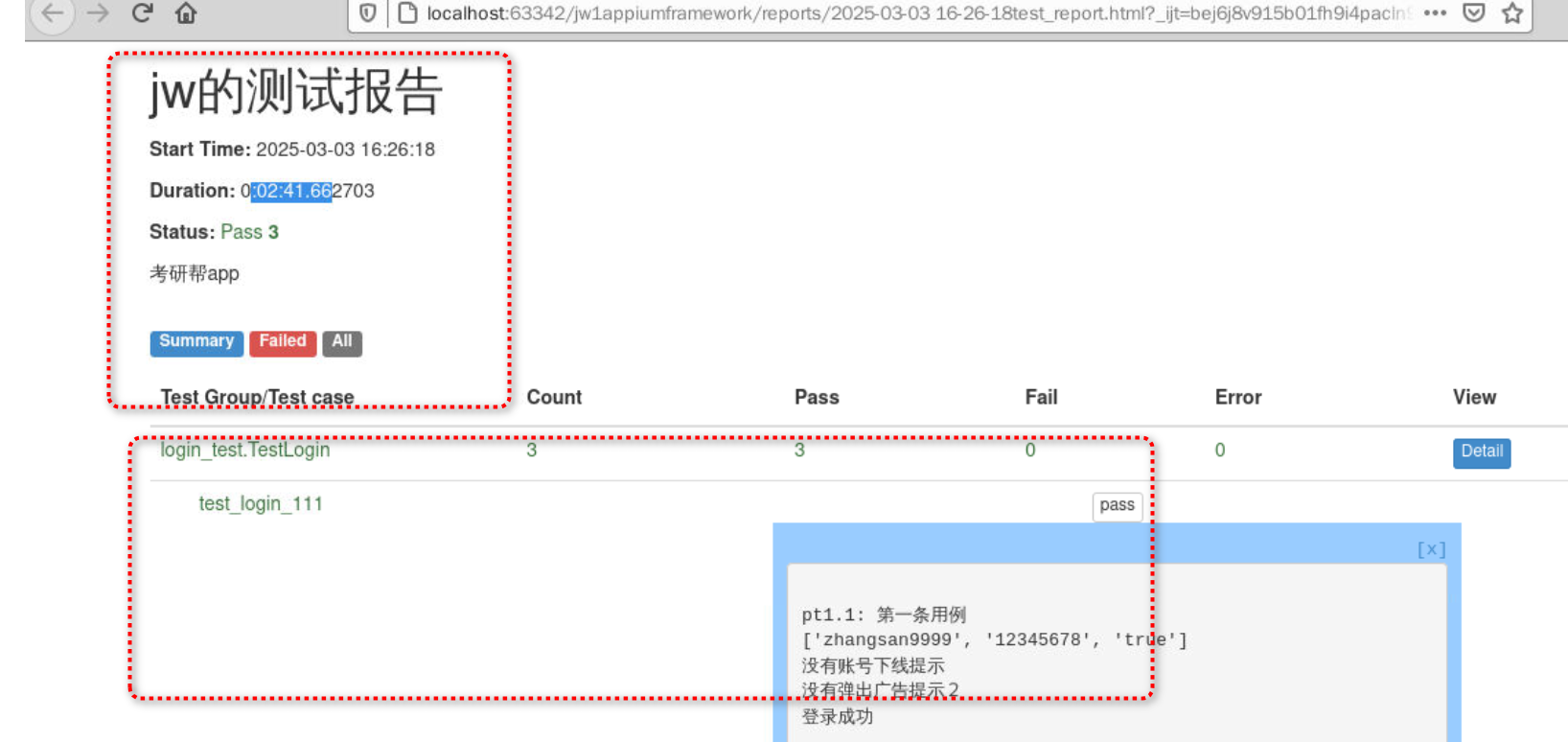
#### 内容6报告插件---html图形化报告





[jwrun.py](https://jwrun.py)

|  |
| --- |
| # 1.指定测试用例和测试报告的目录  # 2.给测试报告命名时间合理 说明 bug 报告产生时间～  # 3.筛选本次要执行的用例  # 4.执行用例、生成报告  import time  import unittest  from BSTestRunner import BSTestRunner  def runner():  # 1.指定测试用例和测试报告的目录  # 用例目录  testcase\_dir = '../testcase'  # 报告目录  report\_dir = '../reports'  # 2.给测试报告命名  now = time.strftime('%Y-%m-%d %H-%M-%S')  report\_name = report\_dir + '/'+ now +'test\_report.html'  # 3.筛选本次要执行的用例  # 路径 和 用例文件很多 都会被执行===》导入测试套～  discover = unittest.defaultTestLoader.discover(testcase\_dir,pattern='\*\_test.py')  # 4.执行用例、生成报告 open 忘了close  with open(report\_name,'wb') as f:  runner = BSTestRunner(stream=f,title='jw的测试报告',description='考研帮app')  runner.run(discover) #执行器执行测试套  # 自测程序  if \_\_name\_\_=='\_\_main\_\_':  runner() |



**小结：**

**1 自动化测试开始 小框架 照抄**

**2 明白每个包每个模块职责**

**3 测试人员+框架中固有的即可**