阅读提示：

阅读本小节，需要读者具备如下前提条件：

1.  掌握一种编程语言基础，如java、python等。

2.  掌握一种单元测试框架，如java语言的testng框架、python的unittest框架。

3.  掌握目前主流的UI测试框架，移动端APP测试框架Appium，或者掌握Web端测试框架Selenium。

4.  如果是Appium框架，还需要掌握如何定位控件元素，原生页面可使用安卓SDK自带的uiautomatorviewer工具，webview页面可以使用Chrome浏览器入"chrome://inspect/#devices"，该方法需要翻墙或者修改hosts。

5.  如果是Web端，浏览器中打开F12即可。

1**APP端UI分层自动化测试框架的搭建**

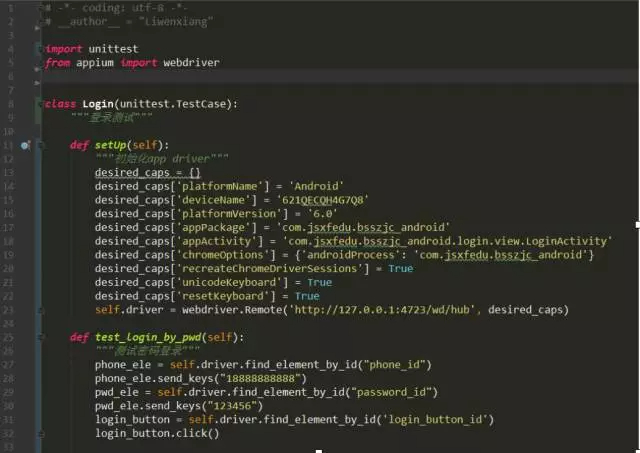
下面以Appium+python+unittest为例介绍UI自动化框架的搭建思路

**1. 最开始入门级别的代码**

以登录页面举例，如下：

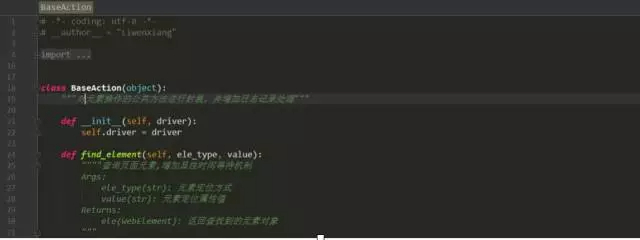


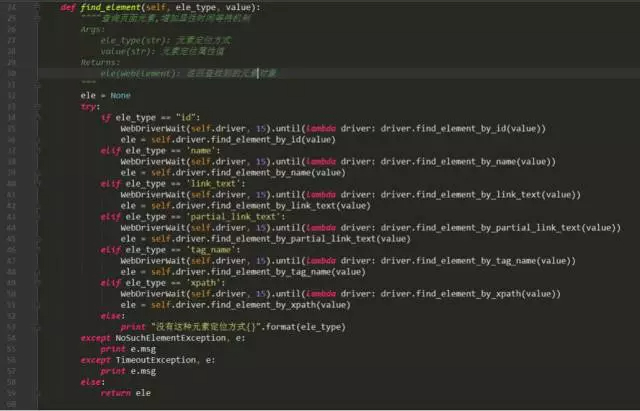
最开始的登录脚本：



**2. 对元素操作的公用方法进行封装**

按照上述代码写了很多行，发现每次操作元素都要写同样的代码，一是啰嗦，二是工作量大，这样就要将重复高的操作抽象出来进行封装成类，及把元素定位，元素点击，元素输入、页面滑动等方法进行封装，如下代码：



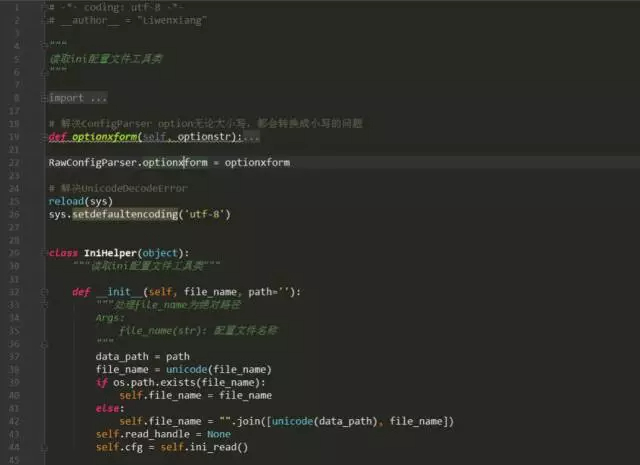


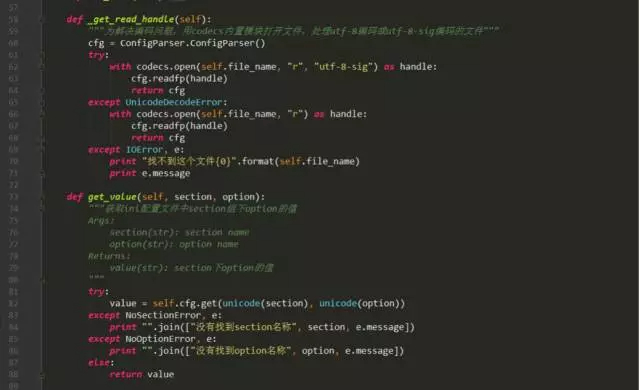
上述代码只是展示了查找元素的方法，在BaseAction类中封装元素对象的重用操作代码笔者就不贴出来，大家明白思路即可。

**3. 测试代码和测试数据分离**

开始测试数据还是写在测试代码中，如果用例多了，后期需要修改数据，这样只能在测试代码中修改，实际中没有这样维护代码的，所以需要将测试数据和代码分离，笔者选择ini配置文件存放数据。

那么问题来了，把数据放在ini配置文件中，在代码中怎样获取数据呢，这就需要将操作配置文件的方法封装成一个单独的类，即如下代码

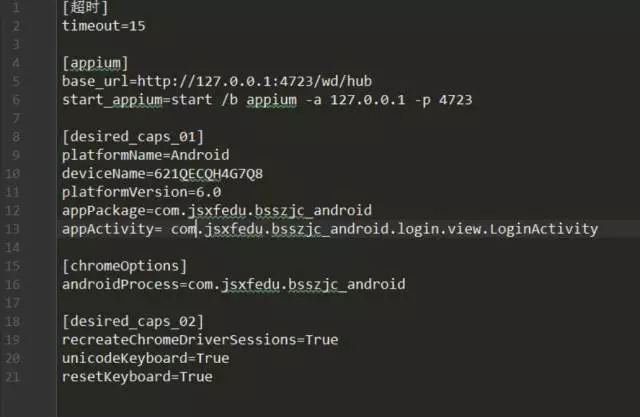




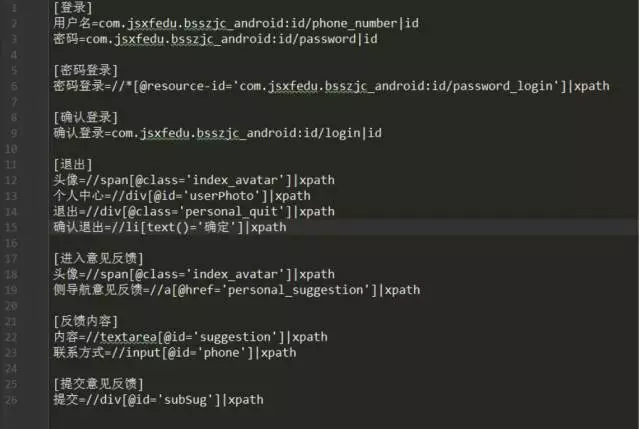
通过对元素操作公用方法的封装和测试代码和测试数据的分类，优化的脚本如下：

存放数据的ini配置文件：

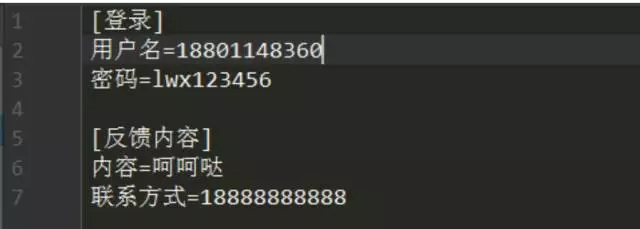
app配置：



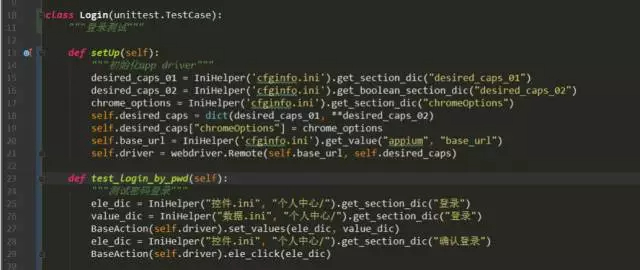
存放元素控件：



存放元素输入的数据：



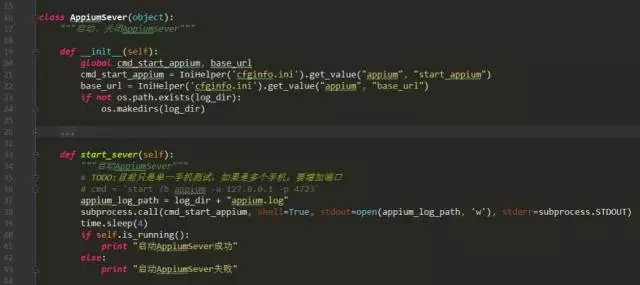
测试脚本：



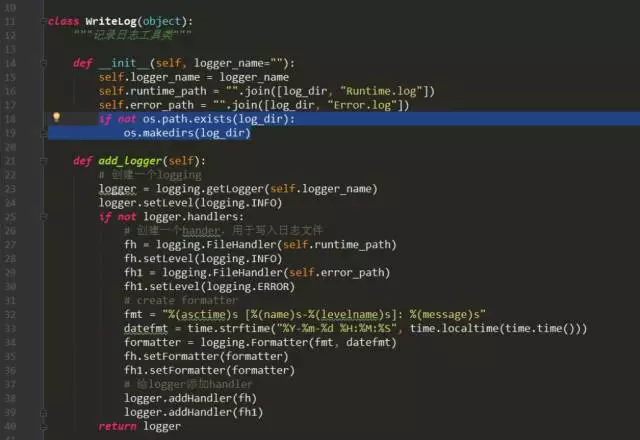
**4. 通用的公用类封装**

以上的脚本中，其实还缺少一些东西，如启动appiumsever，日志记录，发送邮件，数据库操作等，这几种操作都是框架公用的，所以单独封装成类，代码如下：

appiumsever类：



log工具类：



还有发送邮件类和数据库操作类笔者就不贴出来了，重在大家理解思路即可。

**5. 笔者框架分层分享**

现在在测试界吹嘘的分层自动化，意思就是把各种代码实现的类根据不同的功能进行分层，自动化测试框架包括的元素大致如下：

基于Appium框架运用python语言以及unittest单元测试框架，搭建的APP的UI自动化框架如下：

AppAuto/:

  apk: 存放app的apk安装包

  config: 存放配置文件以及测试数据，把所有的项目的配置均放在                                                            这里，用python支持较好的配置文件格式如ini等进行配置。

          实现配置和数据与代码分离。

  screenshot: 存放测试用到的图片以及测试时用例失败截图

  log: 存放日志文件,包括运行时日志runtime.log、error.log和appium.log日志

  report: 存放生成的html格式的测试报告文件

  src:源码层

      common层: 框架级公用方法库

          chche.py: 缓存

          dbsever.py: 数据库操作封装公用类

          emailsever.py: 发送邮件服务封装公用类

          confparser.py: 配置文件解析器

          log.py: 日志记录工具

          appiumsever.py: appium服务公用类

          initappdriver.py: 初始化driver类

          adbconnect.py: adb连接操作

          ...

          (如果还有框架级别的公用方法，还可以在该层封装成类，通过面向对象的方式调用即可)

      functions层: 用例级公用方法库(元素操作公用方法封装，基于PageObject模式对控件公用方法封装，常用业务操作封装)

          eleaction.py: 封装元素操作的一些公共方法

          login.py: 登录操作

          ...

          (该层主要是封装用例层面的公用方法，常用的操作步骤，针对PageObject思想对不同类型的页面控件元素的操作封装等)

      testcase层: 测试用例层

          basecase.py: 测试用例基础类

          testcase1.py: 测试用例1

          ...

      runner层: 测试套件层

          testrunner.py: 各种加载测试用例的方法封装，以及生成报告

  run.py: 执行器，整个框架运行该文件即可

下面是笔者搭建的一个小框架，仅供参考：

