4**Python+Requests接口分层自动化测试框架**

**使用python+requests编写接口自动化测试，需要了解以下几点内容：**

1.  requests第三方库

2. http协议

3.了解json对象

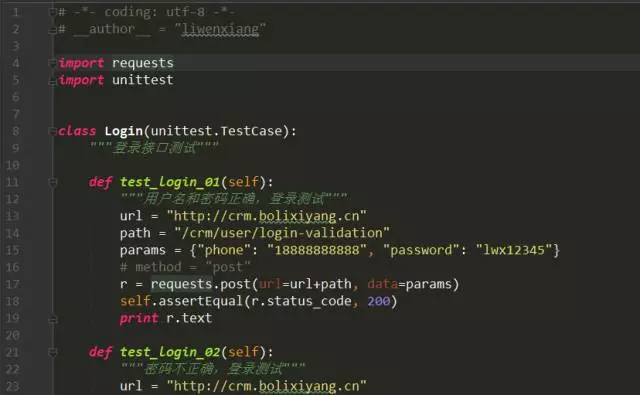
4. 熟悉接口测试，知道如何设计接口测试用例

5. 使用过接口测试工具进行过单个接口测试，如postman、jmeter等

下面详解笔者python+unittest+requests搭建接口分层自动化测试框架的过程：

**1. 最开始的简单代码**

最开始使用unittest框架写的**接口测试代码如下**：





开始时，接口数据都写在测试用例中，且仅仅比较了返回的状态码。

**2. 测试数据和测试代码分离**

在测试用例中维护数据，是很不好维护的，尤其是项目成熟以后，顾需要把测试数据和测试代码分开。笔者采用ini配置文件存放数据，当然大家也可以使用excel或者数据库存放接口数据。**优化后的代码如下**：



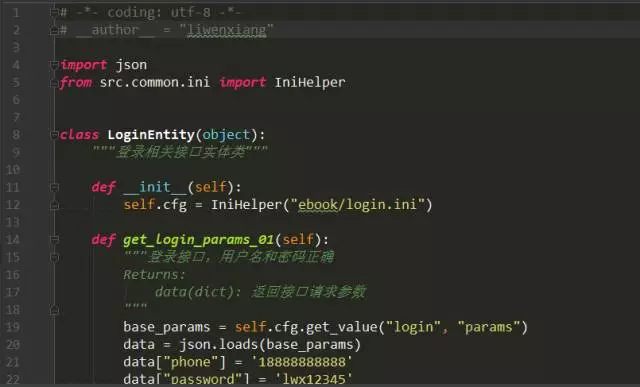


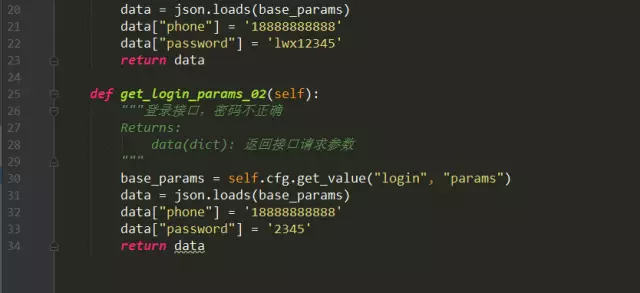
上面这种情况，只能说接口的参数信息放在了配置文件中，其实每个测试接口的方法数据还是需要在测试代码中写，那么有什么方法能让测试代码和数据彻底分开呢？

**3. 增加实体类使测试类和实体类分开**

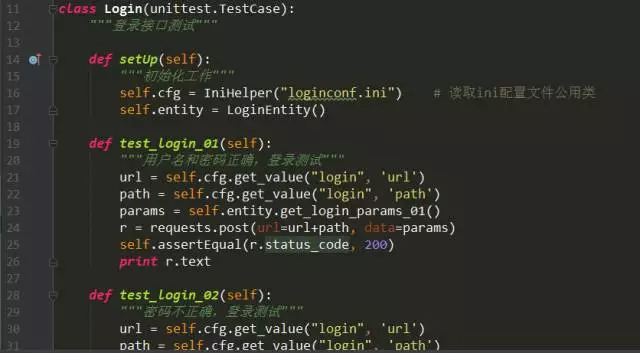
单独把提供接口测试数据的方法封装成一个实体类，为测试类代码提供测试数据，代码如下：

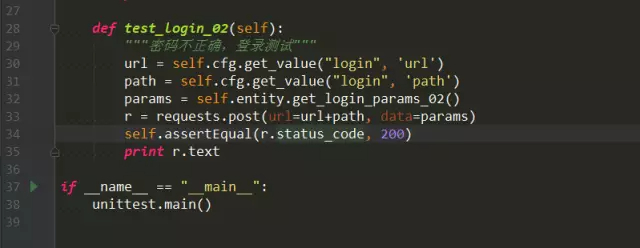
**实体类代码：**





**测试类代码：**





**4. 将实体类和测试类重复的代码抽象封装到各自的基类中**

截止上面的代码，还有很多方法中重复的代码，比如获取接口数据的url、path等，都抽取公用的代码，封装到基类中，测试代码继承基类。该操作的代码我就不贴出来了，感兴趣的可以自己动手敲敲代码去实现。

**5. 其他公用代码封装成公用类**

将接口实际返回的结果和预期结果对比的方法，封装成类或者测试基类中的公用方法

将数据库操作封装成类

将日志记录封装成类

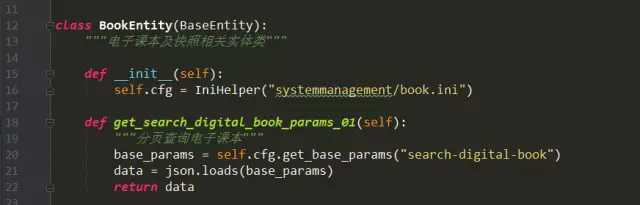
将读取配置文件的操作封装成类

将发送邮件的操作封装成类

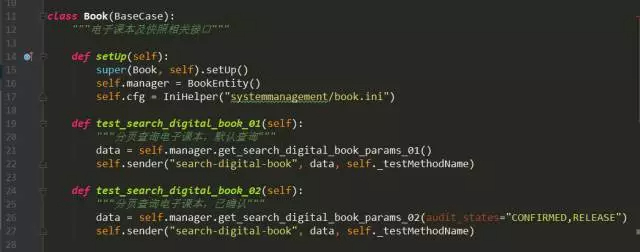
总之，就是把公用的方法单独封装成类对象，然后调用类中的方法即可，遵循面向对象的思想。

**6. 最终优化的实体类和测试类代码**

通过上面的不断优化，笔者**实体类代码如下**：



**测试类代码如下：**



其他的公用类方法就不贴出来了，重在大家理解思路即可。

**7. 接口分层自动化框架分享**

ApiAutoTest/

  --config/层: 存放配置文件以及API接口数据，把所有的项目的配置和接口数据均放在这里，用python支持较好的配置文件格式ini进行配置，实现配置和数据与代码分离。当然也可以使用excel或者数据库存放接口数据）

  --log/层：存放生成的日志文件，包括运行日志Runtime.log和错误日志Error.log等，logger名称为每个测试类的类名。(我是把所有的日志放在了一个文件中，当然根据具体需要，也可以每个测试类生成一个日志文件)

  --report/层：存放程序运行生成的html格式报告

  --src/源码层: 存放所有程序代码。其中还需要更进一步的分层：

      --common/ 公用类

          --cfg：读取配置文件ini公用类

          --db：数据库操作方法公用类

          --log：日志记录公用类

          --email：发送邮件公用类

          --cache：缓存，根据项目具体需求，可把运行时数据存放到缓存中

  --runner/：执行器，组织测试套件suite，生成html格式测试报告

  --testcase/：测试用例层

      --cases/：所有的测试用例类

      --entity/：所有的测试实体类，为测试用例类提供服务，如提供测试数据，也可以在该层增加一些其他公用方法

  --run.py：启动执行文件

