接口自动化测试框架帮助文档

1. 设计理念：为了我们更好更简单的去维护测试框架，我们需要把一些公共参数、公共方法抽离出来，实现数据的分离，我们只需要将精力花在测试用例的设计上。
2. 测试框架：python3+unnitest+requests库
3. 测试环境：python3
4. 需要安装的依赖包：requests、urlib3、xlrd、xlutils、xlwt、pyyaml
5. 框架目录讲解：

test\_interface

|--base（基础方法）

|--bin（运行主入口）

|--config（配置文件）

|--data（数据文件）

|--log（日志）

|--report（报告）

|--tests（测试用例）

|--utils（公共方法）

|--ReadMe.md（说明性的文件，需要的环境以及用法）

1. 框架内容讲解：
2. 首先我们将一些公共参数抽离出来，放在config目录的下的config.yml文件下，在utils目录下添加config.py和fileReader.py来读取config.yml文件的内容。
3. 然后我们将我们的测试数据与请求参数分离出来，在data目录下新建data.xls管理我们的测试数据，随后在data目录下新建requestData.json文件管理我们的请求参数。
4. 由于data.xls测试数据里的表头是固定的，我们可以对它进行封装，在utils目录下，创建excel\_data.py去管理我们的表头，这样我们只需要重点关注测试数据里的内容就可以了。
5. 封装xlrd模块，读取Excel，在utils目录下，创建operationExcel.py文件读取data.xls文件内容。
6. 读取json文件，在utils目录下，创建operationJson.py文件读取requestData.json文件中的数据，并通过get\_request\_data（）方法将data.xls和requestData.json文件关联起来。
7. 然后我们将接口测试经常用到的post，get方法进行了简单的封装。在base目录下创建method.py。
8. 然后我们对logging模块封装了一下，在utils中创建了log.py。
9. 然后我们对运行的错误信息进行了封装一下，在utils中创建了err.py。
10. 由于我们每个用例都需要用到断言，所以我们把断言写到一个公共方法里，在utils中创建了assertion.py。

10、然后生成HTMLTestRunner的测试报告。将HTMLTestRunner.py放在python安装目录的下的lib里。

11、然后在tests目录下创建我们的测试案例，因为我们采用的是unnitest框架来管理我们的测试案例，所以必须遵循unnitest的命名规则，以test开头，否则运行的时候不会执行。

12、最后我们在bin目录下，创建TestRun.py，运行我们所有的测试案例，并生成自动化测试报告。