接口自动化测试框架说明

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Desc.** | **author** | **date** |
| V0.1 | Init version | zhaoyue | 2017/9/4 |

目录

[一 概述 3](#_Toc492260399)

[二 框架结构 3](#_Toc492260400)

[三 流程 4](#_Toc492260401)

[四 features 5](#_Toc492260402)

[五 数据集格式说明 5](#_Toc492260403)

[六 邮件和报告 6](#_Toc492260404)

[七 需要完善 6](#_Toc492260405)

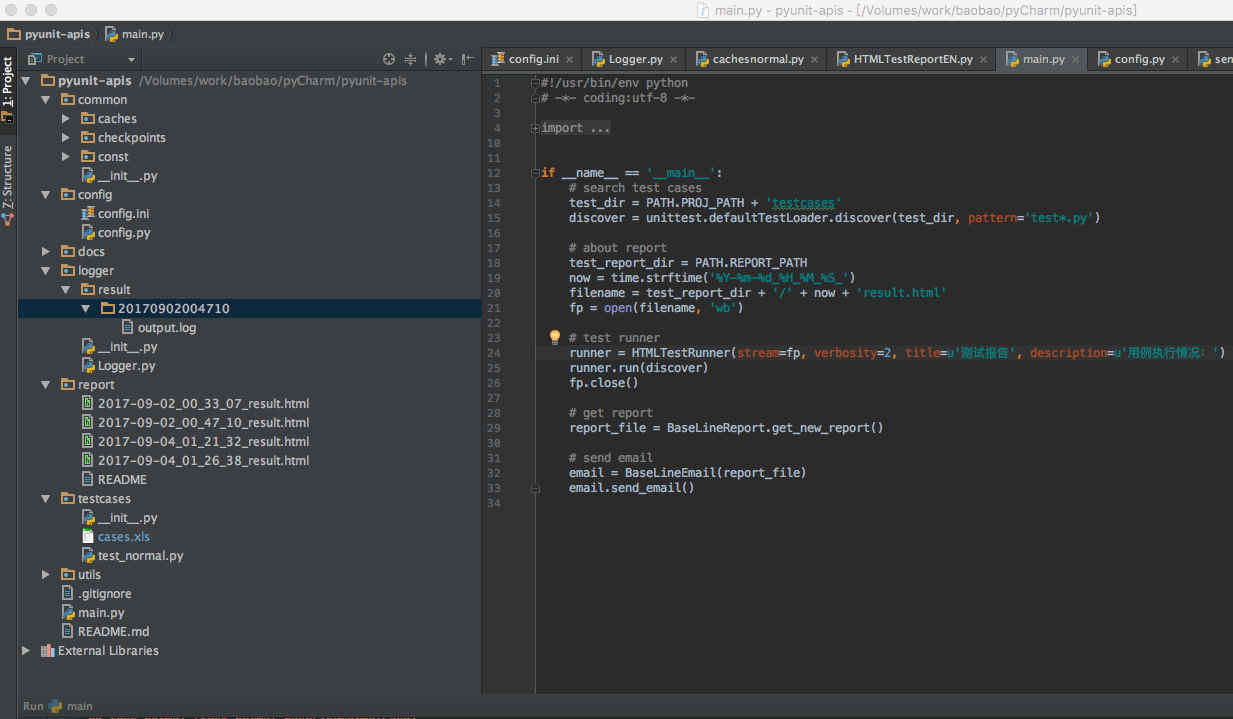
## 一 概述

在自动测试中，数据驱动是比较常用的一种测试框架，指的是测试用例采用数据和代码相分离的方式，使测试套件和测试数据尽量减少耦合。数据驱动的测试框架的优点在于：

1. 测试框架提供通用的测试逻辑，减少工作量，提高测试覆盖率；
2. 代码和测试数据相分离，可维护性强；
3. 测试数据集相对简单，提高团队的合作测试效率。

本框架以数据驱动的形式，采用pyunit来实现。

## 二 框架结构



1. common: 封装通用的接口类，比如checkpoints，const(常量)，lowerapi等；
2. config: 工程配置文件，server ip／port， email相关等配置
3. logger: 日志记录类操作接口；测试生成的日志文件
4. testcases: 测试用例，所有的测试用例都应该放在该目录及其子目录
5. utils: 项目用到的一些工具了类，多数为第三方库的再次封装
6. docs: 设计和说明文档
7. report: 测试报告所在目录
8. main.py 测试总入口

## 三 流程

1. **加载测试套件：**

discover = unittest.defaultTestLoader.discover(test\_dir, pattern='test\*.py')

Testloader会根据pattern规则取测试用例目录查找符合条件的文件，如果文件中有load\_tests函数，则会执行该函数，否则load文件中继承自unittest.TestCase类中以test\_开头的测试用例。

1. **load\_tests：**

如 test\_normal.py文件，因为我们要读取excel文件，所以指定load\_test.py来加载所有测试用例到test\_suit中。

**def load\_tests**(loader, tests, pattern):  
 **from** utils.xlshandler **import** BaseLineXls  
 **from** config.config **import** BaselineConfig  
 **from** common.const.const **import** PATH

首先创建TestSuit对象，然后获取配置文件，创建http实例，所有的测试用例均持有相同的一个http对象，以便来完成全局缓存和参数化列表的支持：

suite = unittest.TestSuite()  
config = BaselineConfig(PATH.CONFIG\_INI\_FILE)  
bl\_http = BaseLineHttp(config.get\_host(), config.get\_port())

之后通过exlhandler来解析excel表格，将数据以参数的形式来创建testcase加载到suit中，将suit返回：

caseXls = BaseLineXls(PATH.CASES\_XLS\_PATH + 'cases.xls')  
cases = caseXls.get\_xls('api\_cases')  
**for** case **in** cases:  
 **if not** isinstance(case, list **or** tuple):  
 **raise** TypeError  
 **if** case[8]:  
 suite.addTest(BaseLineNormalCase(api\_data=case, http=bl\_http))  
  
**return** suite

1. **创建testRunner，执行测试用例：**

创建testRunner变执行测HTMLTestRunner试套件，HTMLTestRunner对unittest.TestRunner进行了封装，传入测试报告的文件句柄，HTMLTestRunner解析unittest.TestResult, 把结果进行整理形成测试报告:

runner = HTMLTestRunner(stream=fp, verbosity=2, title=u'测试报告', description=u'用例执行情况：')  
runner.run(discover)

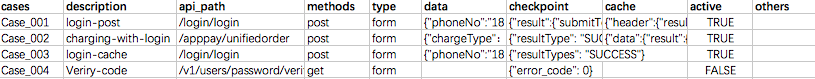
1. **发送邮件：**

# send email  
email = BaseLineEmail(report\_file)  
email.send\_email()

## 四 features

1. 支持post和get请求
2. 支持数据缓存和列表化参数
3. test suit所包含的test cases可配置
4. 支持data数据格式：json
5. checkpoint自动断言，断言类型支持json
6. 支持日志记录
7. 支持清晰的报告展示和邮件通知
8. 采用模块化，易于扩展，比如增加selenium模块可以进行web的自动化测试；增加Uiautomator模块可以进行android app自动化测试。

## 五 数据集格式说明



1. cases: 测试用例ID，最多支持8个字符
2. description:测试用例描述，最多支持30个字符，是否支持中文待验证
3. api\_path: 接口路径
4. methods: get/post/put/delete/head etc.
5. type: form/json/file/sql
6. data: 与type相对应的数据类型；注意：数据引用格式为

{"appidandbillno":"appid=${appid}&billno=${billNo}"}

其中appid和billNo是变量引用

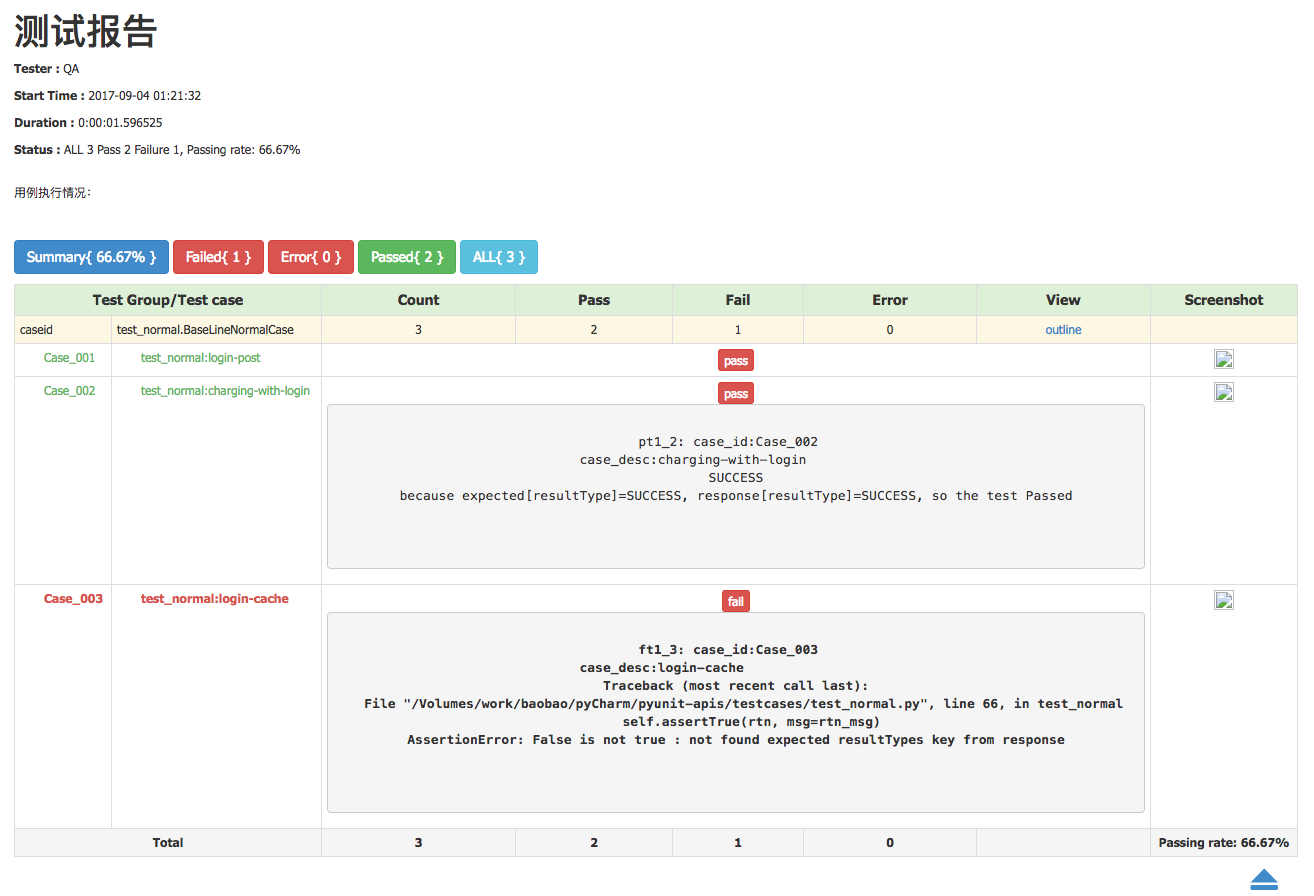
1. checkpoint: 断言点，注意json格式的多级嵌套
2. cache: 数据缓存，支持header和data的缓存

{"header":{"result":{"submitToken":"Submit\_token", "userToken":"OPERATOR\_TOKEN"}}}

最外层的键为header或者data，最内层键所对应的值为要缓存的键，即从结果中查找submitToken的value，然后赋给Submit\_token进行缓存。

1. active: 该case是否激活
2. others:预留

## 六 邮件和报告



邮件会展示case id和case description，成功原因和失败原因都会在详细中列出，后期app 测试和web测试会增加screen short。

## 七 需要完善

1. 增加put/delete/head 等请求支持，增加cookie支持；
2. 增加支持data数据格式：file，sql；
3. 增加断言格式支持：sql；
4. 增加日志记录等级配置以及http 请求超时配置；
5. 增加数据库支持；
6. 对于覆盖不了的测试用例，需要编写单独的脚本实现。

## 八 接口协议规范

1. 接口协议和http协议两者的错误代码和错误描述避免使用相同的字段；
2. 接口的error\_code和error\_msg应该能正确描述错误信息，没有错误也应该有相对应的error\_code和error\_msg;
3. 接口的error\_code和error\_msg一般应该放在json的root层；
4. 考虑到自动化测试，预留测试用字段