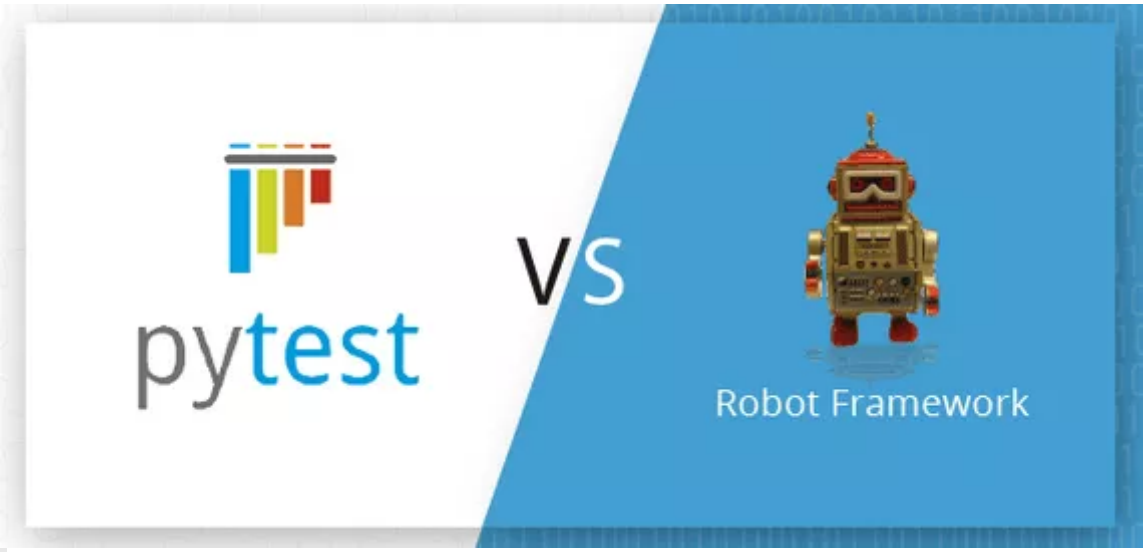
自动化测试框架为什么选择 Pytest，而不是 Robot Framework？

红客联盟 **3月11日**



> Python 自动化测试框架 的优缺点对比，欢迎留言一起交流，进阶学习文末加群。原文链接：https://testerhome.com/topics/17566

之前曾提问请教过 Pytest 和 Robot Framework 的优缺点对比，由于网上关于这方面的信息比较少，收到大家的反馈建议，十分感谢，现在是该总结一下了，欢迎大家一起交流探讨。

在对比框架优缺点之前，先清楚框架的意义是什么？什么是“好的测试框架”必备的特性？

什么是框架？

框架 (Framework) 是整个或部分系统的可重用设计，框架是用来解决代码的组织及运行控制问题的。在我们编写自动化脚本的时候，经常需要读取配置文件，读取数据文件，发送请求，记录日志，连接并对比数据库数据。每个脚本里都重写一遍各种方法不仅工作量大而且易错。所以我们需要把公共的方法提取出来，封装成单独的模块，放到公用方法包里。另外配置文件，数据文件，日志等我们也需要分类存到不同的文件夹下。 **这种对公共方法的封装及对脚本及配置文件怎么组织的设计就叫做框架。**

同时，一般框架除了完成对代码及配置文件的组织之外还要提供运行的控制功能。比如批量执行，分模块执行，生成报告，异常处理等等。

总结为以下 3 点：

- 封装公共方法
- 代码及配置文件的组织
- 执行控制

什么是测试框架？

一个完整的测试脚本（用例）一般包含以下几个步骤：

- 环境准备或检查

- 执行业务操作
- 断言结果
- 清理环境

而测试框架一般还要完成用例加载，批量执行，异常控制，结果输出等功能。基础的测试框架一般只提供执行控制方面的功能。

测试框架应具有的特点

- 易用性：编写用例，执行用例，生成报告及定位问题方便；
- 健壮性：稳定，比如 timeout 机制等；
- 扩展性：插件；
- 灵活性：用例组织或执行的灵活性，Fixture 功能（不同范围的 setUp 和 tearDown）等；
- 定制性：二次开发方便；

Pytest 与 Robot Framework 对比

Pytest 框架特性

Pytest 是一款强大的 Python 测试工具，它具有易于上手，功能强大，第三方插件丰富，效率高，可扩展性好，兼容性强等特点，实际上，**越来越多的项目开始放弃 Unittest 和 Nose 以及 Robot Framework，转而使用 Pytest，比如 Mozilla 和 Dropbox** . 因为 Pytest 可以提供更丰富功能，包括 assert 重写，第三方插件，以及其他测试工具无法比拟的 fixture 模型。

Pytest 可以自动找到测试用例执行，并汇报测试结果。它有丰富的基础库，可以大幅度提高用户编写测试用例的效率。它具备非常强的可扩展性，用户可以自己编写插件，或者安装第三方插件。Pytest 可以很方便地与其他工具一起使用，比如持续集成、web 自动化测试等。

Pytest 也具有 Robot Framework 所闻名的验收测试能力。Pytest 最好的特性之一是，它提供了测试用例的详细失败信息，使开发者可以快速准确地改正问题。它兼容最新版本的 Python，还兼容 unittest、doctest 和 nose，开箱即用。Pytest 还有各种可用插件来给现有测试技术和测试用例增加更多功能和多样性。事实上，在其活跃社区中有 300 多个可用的插件。该平台设计用于编写更简单，错误率更小的代码。你可以将 Pytest 与诸如 Selenium 和 Splinter 之类的图形用户界面一起使用，来使测试工作更轻松

Pytest 优于其他测试框架的地方：

- 简单测试可以很简单的编写
- 复杂测试也可以很简单编写
- 简单灵活，容易上手
- 支持参数化
- 能够支持简单的单元测试和复杂的功能测试，还可以用来做 selenium/appnium 等自动化测试、接口自动化测试（Pytest+requests）
- Pytest 具有很多第三方插件，并且可以自定义扩展，比较好用的如 Pytest-selenium（集成 selenium）、Pytest-allure-adaptor（生成漂亮的 allure 报告）、Pytest-rerunfailures（失败 case 重复执行）、Pytest-xdist（多 CPU

分发分布式执行）等

- 测试用例的 skip 和 xfail 处理
- 可以很好的和 jenkins 集成
- 测试代码可读性强
- 易于上手
- 断言失败仅使用原生关键字 assert，而不是 self.assertEqual()，或者 self.assertLessThan()
- Pytest 可以运行 unittest 和 nose 编写的测试用例
- 不依赖特定的 Python 版本，Python2 和 Python3 都可以使用最新版本
- 正在快速壮大的社区开发和维护

Robot Framework 框架（简称 rf）

优点

- 关键字驱动，自定义用户关键字。
- 支持测试日志和报告生成。
- 支持系统关键字开发，可扩展性好。
- 支持数据库操作。

缺点

- 测试用例写起来不简洁。
- 需要掌握特定语法，学习成本高
- 只支持 Python2，不支持 Python3，容易导致测试团队 Python 版本不统一
- 界面反应速度慢，经常卡死
- 导入测试库有时会异常
- 对于测试团队限制过多，不利于结合具体业务定制功能
- 问题调试困难
- 输出的日志结构复杂，多为英文，不容易看懂
- 维护困难，超过 2000 条用例就是噩梦
- 接口测试，rf 一般是顺序执行的，但在接口测试中需要并发测试，这种测试需要加入写入的并发关键字

```
*** Settings ***
Library    RequestsLibrary
Library    Collections

*** Test Cases ***
test_get_event_list    # 查询发布会（GET 请求）
    ${payload}=    Create Dictionary    eid=1
    Create Session    event
http://127.0.0.1:8000/api
```

```
    ${r}=    Get Request    event    /get_event_list/    params=${payload}
    Should Be Equal As Strings    ${r.status_code}    200
    log    ${r.json()}
    ${dict}    Set variable    ${r.json()}
    #断言结果
    ${msg}    Get From Dictionary    ${dict}    message
    Should Be Equal    ${msg}    success
    ${sta}    Get From Dictionary    ${dict}    status
    ${status}    Evaluate    int(200)
    Should Be Equal    ${sta}    ${status}

```

评判结果：不考虑，没人愿意这么写接口用例。