**接口自动化框架设计**

**—基于python与unittest的实现**

**文档所有者**

|  |  |  |
| --- | --- | --- |
| **姓名** | **职位** | **部门** |
|  |  |  |
|  |  |  |

**批准人列表**

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | **职位** | **批准时间** | **备注** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**参考文档**

|  |  |  |
| --- | --- | --- |
| **文档名称** | **作者** | **内容主题** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**术语和名词解释**

|  |  |
| --- | --- |
| **术语** | **定义解释** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**文档修订摘要**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **版本日期** | **修改者** | **修改内容** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 简介

## 目标

该文档主要介绍此框架的设计、运行环境安装配置等相关内容。

该文档是动态更新的，根据需求，会对某些章节做内容增加、删除、修改等。

该文档目标读者为项目经理、开发人员、测试人员等。

## 范围

该文档主要覆盖以下内容：

* 整体框架
* 目录设置
* 环境搭建
* 命名规范

涵盖文件名、方法名、变量名、类名等

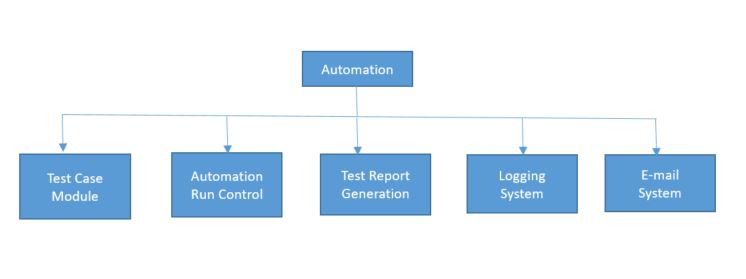
## 假设

对本文档的用户做了以下假设：

* 熟悉Python语言
* 了解Python unittest框架
* 了解数据驱动测试及python数据驱动包ddt
* 有一定的接口自动化开发经验

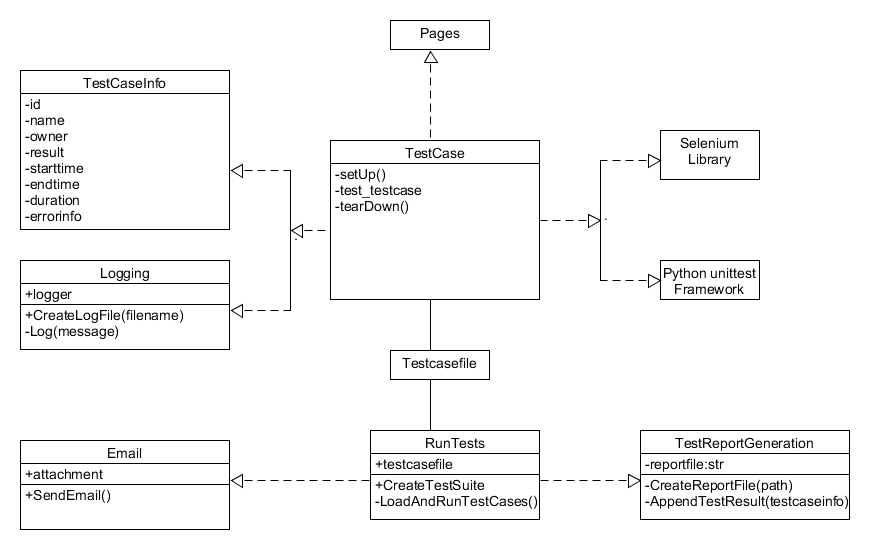
# 整体框架

该框架主要基于Python结合unittest与ddt实现的。通常，一个典型的自动化测试框架一般包括用例管理模板、自动化执行控制器、报表生成模块、日志模块和通知模块（如邮件通知）等。



* Test Case Module：包括用例的增、删、改等操作单元。这些单元也会涉及到用例书写的模式，测试数据的管理（分离），可复用库等。
* Automation Run Control：是自动化用例执行的组织模块，主要包括待执行用例的组织及以什么方式执行待执行的用例。（该框架中使用Python自带的unittest框架作为自动化执行控制模块。）
* Test Report Generation：主要负责执行完用例后生成报表，报表一般以HTML格式居多，内容主要涵盖用例的执行情况及相应的总结报告。（该框架中使用HTMLTestRunner模块（三方模块）作为报告生成模块，以生成HTML格式报告。）
* Logging System：主要用来记录用例的执行过程的情况，以便于更高效地查看失败信息及追踪用例执行情况。（该框架使用Python自带的logging模块实现日志记录。）
* 通知模块：主要用来及时地将用例执行结果推送至项目相关人员。此模块乃泛指——可是以短信方式，其他通讯工具如微信、QQ等——具有类似通知功能的模块。（该框架支持腾讯企业邮箱、QQ邮箱、网易163邮箱及Outlook邮件通知。）

基于上述典型自动化测试框架需包含的内容，以及业界常用的对象维护方式，该框架总体设计（概要设计类图）如下：

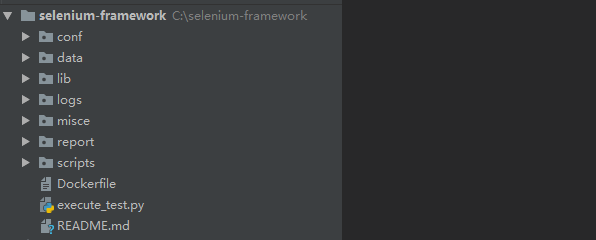


# 目录设置

对于做自动化测试的新手来说，遇到的一个瓶颈就是虽然能够写脚本，但是不知道怎么去组织代码，从而增强代码的可复用性、可维护性和降低耦合性。该框架基于上述的整体框架设计给出了一种目录设置方案，主要包含以下一些目录：

* conf：放置配置文件，把项目相关的配置均放置该目录，用python支持较好的配置文件格式如ini或yaml等进行配置，实现配置与代码分离。
* data：放置数据文件，把所有测试用例参数化并将参数文件放置在该目录，一般可用xls、csv、xml等格式，实现数据与代码分离。
* misce：放置框架相关的东西，如框架设计、使用手册等文档。（该目录可要可不要，这里只是作为存储目录使用，与脚本运行无关）。
* lib：公共库目录，放置浏览器driver、三方包、公共函数及页面对象。
* logs：所有脚本运行中生成的日志均放置在该目录，可将日志分类，如运行时日志、错误日志等（参考python的logging模块手册）。
* report：放置程序运行生成的报告，如该框架最终生成的html报告。
* scripts：放置实际的测试用例脚本。

框架目录如下截图所示：



# 环境搭建

## Windows

1. 安装Python：<https://www.Python.org/>

请下载安装3.5 或以上版本

添加Python的安装目录到系统环境变量Path

添加$Python\Scripts目录到系统环境变量Path

2. 安装浏览器

建议使用Google Chrome、firefox、edge

3. 安装最新的浏览器驱动

根据将使用的浏览器，下载浏览器对应的驱动，浏览器版本与驱动对应关系，请参考<http://blog.51cto.com/taoismli/2157024>博文中第二小节内容，并将下载的driver放置到框架的lib目录下

4. 安装框架所需的三方工具包

框架所需的三方工具包已整理至框架根目录下的requirements.txt文件中，可直接用pip一次性安装或用IDE安装

>pip install -r path\requirements.txt

## Linux

1. 安装python(linux系统一般默认已安装python)

请检查是否为3.0以上版本，如果不是请另行安装，且注意安装后配置PATH，保证默认启动的是安装的版本（或者在启动脚本中指定执行脚本的解释器）

2. 安装浏览器（Linux安装浏览器可网上找相关教程）

建议使用Google Chrome、firefox

3. 安装最新的浏览器驱动

根据将使用的浏览器，下载浏览器对应的驱动，浏览器版本与驱动对应关系，请参考<http://blog.51cto.com/taoismli/2157024> 博文中第二小节内容，并将下载的driver放置到框架的lib目录下

4. 安装框架所需的三方工具包

框架所需的三方工具包已整理至框架根目录下的requirements.txt文件中，可直接用pip一次性安装或用IDE安装

>pip install -r path\requirements.txt

## Linux虚拟桌面执行脚本

在Linux测试服务器上，可用xvfbwrapper模块实现虚拟桌面以执行脚本：

安装命令： pip install xvfbwrapper

# 命名规范

模块名：module\_name

包名：package\_name

方法名：method\_name

函数名：function\_name

实例名：instance\_var\_name

方法参数名：function\_parameter\_name

本地变量/全局变量名：local\_var\_name.

异常名：ExceptionName

类名：ClassName

常量名：GLOBAL\_VAR\_NAME