|  |
| --- |
| **自动化框架使用手册** |
|  |

版本: 0.1

作者: 李于

版本日期: Friday, February 15, 2019

**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** | **版本** |
|  |  |  |
|  |  |  |
|  |  |  |

**术语和名称解释**

|  |  |
| --- | --- |
| **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** |
| 0.1 | 2019/02/15 | 李于 | 初稿 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 简介

## 目标

该文档主要介绍此框架的设计、运行环境安装配置及框架运用相关内容。

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

该文档将覆盖以下内容：

* 整体框架
* 目录设置
* 环境搭建
* 命名规范（文件名、方法名、变量名、类名等）
* 用例实例（基于信用重庆）
* 测试运行

## 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
* 熟悉Selenium
* 熟悉常用的页面元素定位方法
* 有一定的UI自动化开发经验

# 整体框架

Selenium是一个基于浏览器的自动化工具，它提供了一种跨平台、跨浏览器的端到端的web自动化解决方案。Selenium主要包括三部分：

* Selenium IDE： Firefox浏览器的一个插件（扩展），它可以进行录制回放，并且可以把录制的操作以多种语言（如JAVA、Python、C#等）的形式导出成测试用例。
* Selenium WebDriver： 提供Web自动化所需的API，主要用作浏览器控制、页面元素选择和调试。不同的浏览器需要不同的WebDriver。
* Selenium Grid： 提供了在不同机器的不同浏览器上运行selenium测试的能力。

这里的框架主要使用Python结合Selenium WebDriver库进行搭建的。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、日志模块和邮件发送模块等。

Automation

E-mail

System

Logging

System

Test Report

Generation

Automation

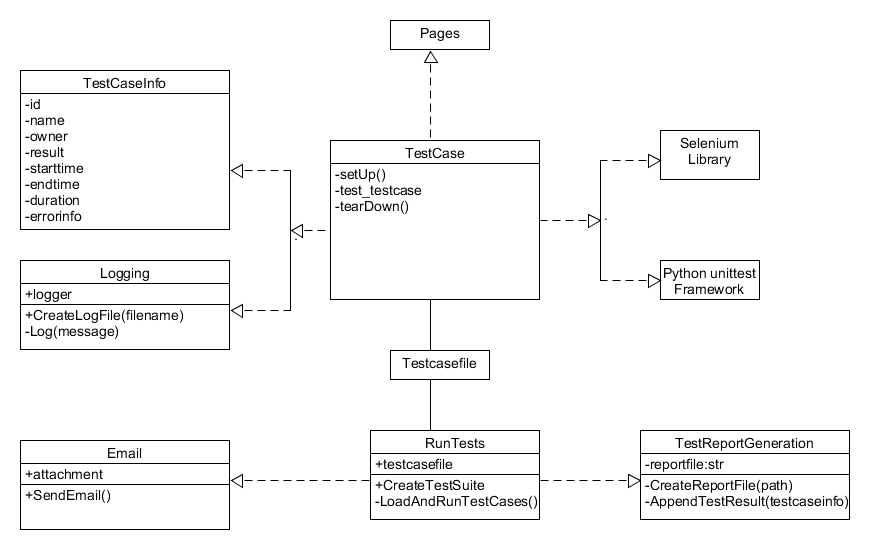
Run Control

Test Case

Module

* Test Case Module：包括用例的添加、修改、删除等操作单元。这些单元也会涉及到用例书写的模式，测试数据的管理，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。（该框架中运用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。（该框架中运用HTMLTestRunner第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。（该框架运用Python自带的logging模块实现日志记录。）
* E-mail System：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。（该框架以腾讯企业邮箱为例配置。）

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：

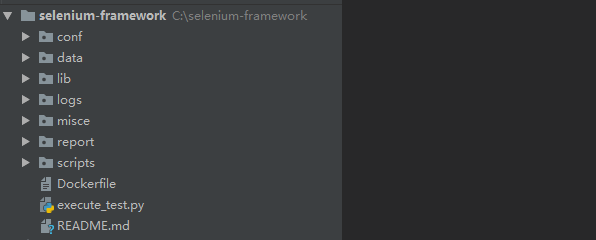


# 目录设置

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，把所有测试用例的参数化相关的文件放置该目录，一般可用xlsx、csv、xml等格式，实现数据与代码分离。
* misce：放置框架相关的东西，如使用手册、用例设计文档等。（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，比如该框架最终生成的html报告。
* scripts：放置实际的脚本。

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Selenium

cmd里运行“pip install selenium”。

3. 安装浏览器：建议使用Google Chrome或firefox。

4. 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$Python\Scripts目录下。

5. 安装HTMLTestRunner.py

请下载支持python 3.x的版本。

下载地址：<https://download.csdn.net/download/a136332462/9873996>

将下载的文件放置到$Python\Lib目录下。

## 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，如果不是请另外安装。注意安装后配置PATH，保证默认启动的是安装的版本。

2. 安装setuptools与pip

安装Python之后安装setuptools和pip非常方便，请参考<http://blog.51cto.com/taoismli/2157024> 博文中第一小节中的C小节。

3. 安装selenium

pip3 install selenium

4. 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$Python目录下：

#python3

>>>import sys

>>>sys.path

>>>exit()

#cp <driver\_name> <python\_path>

5. 安装HTMLTestRunner.py

请下载支持python 3.x的版本。

下载地址：<https://download.csdn.net/download/a136332462/9873996>，将下载的文件放置到$Python/site-packages/目录下

#python3

>>>import sys

>>>sys.path

>>>exit()

#cp HTMLTestRunner.py <$Python/site-packages/>

检查是否安装成功

#python3

>>>import HTMLTestRunner //不报错，则表示安装成功

## 安装第三方库（Windows & Linux）

由于已经安装pip，则安装第三方库将变得非常简单。本框架中和示例中使用到如下一些第三方库，请依次安装：

bs4

xlrd

xlwt

xlutils

chardet

xvfbwrapper //该模块用于linux系统虚拟桌面环境

安装命令： pip install PackageName

# 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

命名规范参考地址：

<http://www.cnblogs.com/zhanglianbo/p/5664997.html>

# 用例实例

|  |  |  |
| --- | --- | --- |
| **脚本名称** | **测试页面** | **备注** |
| cqnet\_publicity\_permission.py | 信用公示 – 行政许可  http://www.xycq.gov.cn/html/query/agree/list.html | 已实现：  检查页面元素、数据字符集判断  使用信用主体名称精确查询  使用统一社会信用代码精确查询  使用工商注册号精确查询  未实现：  点击主体名称检查许可详情页  使用信用主体模糊查询 |
| cqnet\_query\_credit.py | 信用查询 – 信用查询  http://www.xycq.gov.cn/html/query/credit/list.html | 已实现：  检查页面元素  使用企业名称精确查询  使用信用代码精确查询  使用法定代表人精确查询  未实现：  点击信用代码检查信用详情页  使用企业名称模糊查询 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## cqnet\_publicity\_permission.py

|  |  |  |
| --- | --- | --- |
| 方法名称 | 检测内容 | 未检测内容 |
| test\_preparation | 测试准备：  获取页面地址  获取页面加载时间限制  获取查询时间限制 |  |
| test\_page\_element | 定位行政许可页面的元素  判断是否所有元素已显示  数量是否匹配  页面数据字符集判断 | 未点击“信用主体”链接检查“许可详情” |
| test\_get\_valid\_data | 从每个分页里（总共10页）随机抽取某行数据作为精确查询的验证数据：  信用主体名称  统一社会信用代码  许可机关  许可时间  工商注册号 |  |
| test\_query\_exact\_by\_host | 精确查询：通过“信用主体”查询 |  |
| test\_query\_exact\_by\_credit\_code | 精确查询：通过“统一社会信用代码”查询 |  |
| test\_query\_exact\_by\_reg\_code | 精确查询：通过“工商注册号”查询 |  |
|  |  |  |
|  |  |  |

下面列出的每个部分都描述了脚本的一个方法，其中包含该方法的测试目标及实现内容。

### test\_permission\_page\_element

|  |  |
| --- | --- |
| **方法名** | test\_permission\_page\_element |
| **开发者** | 1. 胡天惠 |
| **检查内容描述** | 定位行政许可页面的元素，判断是否所有元素已显示，数量是否匹配：  定位页面标题“行政许可”  定位默认查询类别为“信用主体名称”  定位查询类别列表为4类  定位查询输入框  定位数据表为10行数据  定位页面按钮（可点击的）为9个  随机抽查某行的“信用主体”名称进行字符集检查 |
| **入参** | N/A |
| **出参** | True: 测试通过  False：测试失败 |

实现逻辑如下表：

|  |  |
| --- | --- |
| **步骤** | **描述** |
| 1 | 设置 is\_test\_pass 为True |
| 2 | 设置元素定位信息 |
| 3 | 定位各元素 |
| 4 | 将定位的元素数量、内容与期望对比，若其中任何有问题，设置is\_test\_pass 为 False |
| 5 | 返回 is\_test\_pass |
|  |  |

|  |  |
| --- | --- |
| **注意事项** | |
| 1 |  |

# 测试运行