自动化测试模型

**概念**

### 自动化测试模型可以看作自动化测试框架与工具设计的思想。自动化测试不仅仅是单纯写写脚本运行就可以了，还需要考虑到如何使脚本运行效率提高，代码复用、参数化等问题。自动化测试模型分为四大类：线性模型，模块化驱动测试、数据驱动、关键词驱动。

本地Web测试站点搭建：

* 工具：帝国CMS
* 下载地址：<http://www.phome.net/download/>

**- 安装步骤**

* 1、解压安装包；
* 2、将安装包的“EmpireServer”目录复制到D盘根目录；(D:\EmpireServer目录名不可更改)
* 3、双击“D:\EmpireServer\一键安装.bat”；(用vista或windows7以上则要鼠标右键以管理员身份运行)
* 4、至此，运行环境及帝国CMS全部安装完毕；
* 5、在浏览器打入：<http://localhost/e/admin> 后回车，进入帝国CMS后台登陆界面。
* 前台地址：<http://localhost>
* 后台地址：<http://localhost/e/admin> (登录用户名、密码与认证码均为admin)
* 搭建完成之后注册一个测试账号 51zxw 密码：123456

### 线性模型

线性脚本中每个脚本都相互独立，且不会产生其他依赖与调用，其实就是简单模拟用户某个操作流程的脚本。

案例：在帝国软件主页自动登录和退出操作

from selenium import webdriver

from time import sleep

driver=webdriver.**Firefox**()

driver.**get**("http://localhost")

#输入用户名

driver.**find\_element\_by\_name**('username').**clear**()

driver.**find\_element\_by\_name**('username').**send\_keys**('51zxw')

#输入密码

driver.**find\_element\_by\_name**('password').**clear**()

driver.**find\_element\_by\_name**('password').**send\_keys**('123456')

#点击登陆

driver.**find\_element\_by\_name**('Submit').**click**()

**sleep**(3)

#退出

driver.**find\_element\_by\_link\_text**('退出').**click**()

**sleep**(2)

driver.**switch\_to\_alert**().**accept**()

**sleep**(3)

driver.**quit**()

### 模块化驱动测试

线性模型虽然每个用例都可以拿出来独立运行，但是用例之间重复代码很多，开发、维护成本高。其实把重复的操作代码封装为独立的公共模块，当用例执行时需要用到这部分，直接调用即可，这就是模块驱动的方式。比如登录系统、退出登录、截图函数等等。

from selenium import webdriver

from time import sleep

class **Login**():

def **user\_login**(self,driver):

driver.**find\_element\_by\_name**('username').**clear**()

driver.**find\_element\_by\_name**('username').**send\_keys**('51zxw')

driver.**find\_element\_by\_name**('password').**clear**()

driver.**find\_element\_by\_name**('password').**send\_keys**('123456')

driver.**find\_element\_by\_name**('Submit').**click**()

def **user\_logout**(self,driver):

driver.**find\_element\_by\_link\_text**('退出').**click**()

**sleep**(2)

driver.**switch\_to\_alert**().**accept**()

**if** \_\_name\_\_=='\_\_main\_\_':

driver = webdriver.**Firefox**()

driver.**get**("http://localhost/")

driver.**implicitly\_wait**(10)

**Login**().**user\_login**(driver)

**Login**().**user\_logout**(driver)

调用登录模块

**from** LoginClass import \*

driver = webdriver.Firefox()

driver.get("http://localhost/")

driver.implicitly\_wait(10)

Login().user\_login(driver)

Login().user\_logout(driver)

### 数据驱动测试

模块驱动的模型虽然解决了脚本的重复问题，但是需要测试不同数据的用例时，模块驱动的方式就不很适合了。 数据驱动就是数据的改变从而驱动自动化测试的执行，最终引起测试结果的改变。 装载数据的方式可以是列表、字典或是外部文件（txt、csv、xml、excel），目的就是实现数据和脚本的分离。

from selenium import webdriver

from time import sleep

class **Login**():

def **user\_login**(self,driver,username,password):

driver.**find\_element\_by\_name**('username').**clear**()

driver.**find\_element\_by\_name**('username').**send\_keys**(username)

driver.**find\_element\_by\_name**('password').**clear**()

driver.**find\_element\_by\_name**('password').**send\_keys**(password)

driver.**find\_element\_by\_name**('Submit').**click**()

def **user\_logout**(self,driver):

driver.**find\_element\_by\_link\_text**('退出').**click**()

**sleep**(2)

driver.**switch\_to\_alert**().**accept**()

数据驱动调用——实现多个账户登录

**from** LoginClass\_Para import \*

**from** selenium import webdriver

**from** time import sleep

driver=webdriver.Firefox()

driver.get("http://localhost")

driver.implicitly\_wait(10)

Login().user\_login(driver,"51zxw",'123456')

sleep(3)

Login().user\_logout(driver)

Login().user\_login(driver,"51zxwPro",'123456')

sleep(5)

Login().user\_logout(driver)

### 关键字驱动测试

通过关键字的改变引起测试结果的改变叫关键字驱动测试。 selenium IDE也是一种传统的关键字驱动的自动化工具，Robot Framework 是一个功能更强大的关键字驱动测试框架

