自动化测试概述

```
1.自动化测试的产生的背景
    1.重复、频繁的工作太多
    2.任务重,时间短
    3.创新性的工作太少
    4.测试结果有时需要精确到毫秒，手工做不到
2.概述:自动化测试就是借助于测试工具、依照测试规范，从局部或全部代替人工进行测试以及提高测试效率的过程。
3.自动化测试的两大特征
    1.自动测试过程是通过模拟人工操作，完成对被测系统的输入，开且对输出进行检验的过程；
    2.自动测试是由软件代替人工操作，对被测系统的GUI发出指令，模拟操作，完成自动测试过程；
4.优势
    1.可重复
    2.可程序化
    3.可靠
    4.提高测试的精确度
    5.提升测试资源的的利用率
5.和手工测试比较的特点
    1.优化成本；
    2.可靠；
    3.快速；
    4.规范化；
    5.可重用。


```

手工测试VS自动化测试

	手工测试	自动化测试
谁发现的缺陷多？（测试质量的高低）	能够更多的发现缺陷	需要依赖脚本和代码的实现。如果脚本和代码没有让程序自动进行判断，工具程序永远不会发现bug
测试质量	能够更好的发现缺陷，甚至是用例中没有涉及的缺陷	程序中没有涉及到的测试点，不做测试，因此会漏掉一部分缺陷
与开发关系	一般和开发较为对立，不写代码，不知道开发有多难	因为写代码，所以更能体会和理解开发的难处；跟开发的关系也更为密切
各自的局限性	执行效率低,可靠性不高,准确度会受到情绪的影响,精确度不够高	执行效率高,可靠性高，全天候全时段运行,准确度高,精确度高,不易出错 .如果代码不对，就无法得出准确结果;如果被测试的程序界面修改了，代码也不能正常运行没有错误联想功能

自动化测试工具的分类

```
1.从软件使用的目的分类:
功能测试:QTP/UFT,Selenium
性能测试:Loadrunner, JMeter
测试管理:禅道
其他测试:Postman
2.从软件的获取分类:
自主研发
开源
商业

```

自动化测试工具的介绍

```
1、功能上自动化测试工具：基于界面（UI）/黑盒的测试工具
    1）web自动化测试：Selenium可以通过一个插件，实现可视化的操作，但是功能很单一；突出特点是通过Java或者Python进行脚本代码的编写，实现更自由的自动化测试
    2）移动端自动化测试：APPium，是一个可以实现可视化操作，重点也是通过代码完成对移动端APP的自动化测试
    3）QTP---UFT：Mercury公司首发，HP公司收购，Micro focus购买；功能很强大；非常专业的；收费很高

2、性能测试工具：需要代码基础不需要很高，但是业务的分析、专业性能术语要求较高
    1）HP Loadrunner。Mercury公司首发，HP公司收购，Micro focus购买；功能很强大；非常专业的；收费很高
    2）Apache Jmeter（阿帕奇）：开源，刚开始，只进行web服务的性能测试；后来扩展到很多功能

3、测试管理工具：在数据统计、任务分配上有很好的作用
    1）bugfree---禅道：开源版/商业版。功能齐全
    2）QC/ALM：Mercury公司首发，HP公司收购，Micro focus购买；功能很强大；非常专业的；收费很高
    3）JIRA：商业软件；费用也不低；功能也比较强大

4、接口测试工具：技术含量不高，但是需求较多
    1）Postman。界面美观，功能完善，但是只支持进行HTTP和HTTPS的协议的接口测试。商业软件，可以免费使用
    2）Apache Jmeter（阿帕奇）：开源，支持多种协议的接口测试
    3）Fiddler。抓包工具。
```

实施自动化测试-流程(重点)

```
一.流程:
1.自动测试决定
2.测试工具获取
3.自动化测试引入
4.制定测试计划（5w1H）、测试设计（测试用例：测试步骤、测试数据、预期结果）、测试开发（编写测试脚本、在工具中完成测试场景的开发）---最关键的一个环节
5.测试执行与管理（脚本的运行、过程监控、结果管理）
6.测试审评和评估
```

实施自动化测试-测试方案的选择

```
二.考虑因素:
1.项目影响
2.复杂度
3.时间
4.早期需求和代码的稳定性
5.编护工作量
6.覆盖率
7.资源
8.自动化测试的执行
```

实施自动化测试-场景

```
三.自动化测试实施场景
合适:
1.回归测试
2.更多更频繁的测试
3.手工测试无法实现的工作
4.跨平台产品的测试
5.重复性较强的操作
不合适:
1.软件版本不稳定
2.涉及与物理设备交换的测试
3.测试结果较容易通过人工判断的测试

```

实施自动化测试-工具的选择

```
四.自动化测试工具选择
1.只买对的，不买贵的
2.选择主流的测试工具
3.分阶段、逐步引入测试工具
4.选择技术支持完善的产品
5.如需多种工具，尽量选择一个公司的产品
```

自动化测试前景

```
1.自动化测试前景:那是相当的好
2.目前自动化测试工程师的待遇：测试和开发技术够好，年薪完全能拿到20w，30w，40w，甚至更多。
3.自动化测试是未来测试的普遍方式
```

Selenium概述(重点)

```
1.selenium是测试工具
2.开源的
3.可以写脚本调用selenium,然后selenium调用驱动去控制浏览器
4.Selenium3.0中的Firefox浏览器驱动独立了,现在也要下载浏览器驱动
5.不同的浏览器驱动不同,而且相差很大的版本的同一种浏览器驱动也不同

谷歌浏览器驱动
http://chromedriver.storage.googleapis.com/index.html
火狐浏览器驱动:
https://github.com/mozilla/geckodriver/releases
```

Selenium环境搭建(重点)

```
1.python+selenium，是需要自己搭框架的，搭完框架后需要自己去写脚本

第一步:
第一种方式:
1.把下载的驱动程序,放到一个英文命名的文件夹内
2.配置环境变量

第二种方式:
把驱动程序直接放到项目中

第二步:
下载selenium模块
第一种方式,使用pycharm中的设置里下载
第二种方式,终端输入:pip install selenium
第三种方式,代码中输入import selenum,如果没有会报红,鼠标点上,旁边有一个红色小灯泡,点击小灯泡,弹出对话框,选择install selenium

第三步:
写入下代码,然后运行,如果弹出浏览器,证明ok
from selenium import webdriver
dr=webdriver.Firefox()

```

页面元素定位介绍(重点)

```
selenium可以识别的元素,要求属性必须唯一
1)id  用id的值
2)xpath 路径
3)css selector 标签名+class属性的值
4)link text 使用链接文字
5)patial link text  使用子元素的链接文字
6)name 用name属性的值
7)class name 用class属性的值
8)tag name 用标签名
查找顺序是从html开始
e=dr.find_element(By.ID,'kw')-- 3.7以上版本的
from selenium.webdriver.common.by import By
By.ID
By.NAME
By.LINK_TEXT
By.PARTIAL_LINK_TEXT
By.TAG_NAME
By.CLASS_NAME
By.CSS_SELECTOR
By.XPATH

一次识别一个元素的api
• find_element_by_id()
• find_element_by_name()
• find_element_by_link_text()
• find_element_by_partial_link_text()
• find_element_by_tag_name()
• find_element_by_class_name()
• find_element_by_xpath()
• find_element_by_css_selector()

• Selenium一次识别多个元素的API
• find_elements_by_id()
• find_elements_by_name()
• find_elements_by_link_text()
• find_elements_by_partial_link_text()
• find_elements_by_tag_name()
• find_elements_by_class_name()
• find_elements_by_xpath()
• find_elements_by_css_selector()
```

find_element_by_id()函数(重点)

```
1.在id定位里，会返回第一个id属性匹配的元素，如果没有元素匹配，会抛出NoSuchElementException异常。
2.函数说明:
find_element_by_id("元素的id值") 精确查找到元素
返回的是元素
3.获取远程页面的元素
```

```
dr = webdriver.Firefox()
dr.get("完整域名")
4.获取本地html页面的元素
dr = webdriver.Firefox()
dr.get("file:///'+文件的绝对路径)

目的:把"自动化测试"这几个字输入到百度首页的搜索框中
#!/usr/bin/env python
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
#====#====#====#====

from selenium import webdriver
import time

#1.获取浏览器,并打开
dr=webdriver.Firefox()
#2.进入指定的网站
dr.get("https://www.baidu.com")
#3.暂停2秒,让页面充分加载
time.sleep(2)
#4.通过id的值获取搜索框元素
e=dr.find_element_by_id('kw')
#5.在搜索框中输入信息
e.send_keys("自动化测试")
#6.暂停3秒
time.sleep(3)
#7.关闭浏览器
dr.quit()



案例:找淘宝首页的搜索框元素,通过id获取元素,然后通过元素获取name的值
案例:自己写一个文本域,属性有id,和name,通过id获取元素,然后通过元素获取name的值
```

find_element_by_name()函数(重点)

```
1.在name定位里，会返回第一个name属性匹配的元素，如果没有元素匹配，会抛出NoSuchElementException异常。
2.函数说明:
find_element_by_name("元素的name值") 页面上的元素中name的值可以相同,所以这个方法不能精确查找元素
返回元素
#!/usr/bin/env python
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
```

```
#====#====#====#====

from selenium import webdriver
import time

#1.获取浏览器,并打开
dr=webdriver.Firefox()
#2.进入指定的网站
dr.get("https://www.baidu.com")
#3.暂停2秒,让页面充分加载
time.sleep(2)
#4.通过id的值获取搜索框元素
e=dr.find_element_by_name('wd')
#5.在搜索框中输入信息
e.send_keys("自动化测试")
#6.暂停3秒
time.sleep(3)
#7.关闭浏览器
dr.quit()




案例:去淘宝首页找有name属性的元素,通过元素获取其他属性值
#案例:自己去写一个html页面,页面上有一个输入框,找有name属性的元素,通过元素获取id属性的值

```

find_element_by_link_text()函数

```
链接文本定位,<a></a>标签
1.在超链接定位里，会返回第一个文本属性匹配的元素，如果没有元素匹配，会抛出NoSuchElementException异常。
2.函数说明:
find_element_by_link_text("链接文字") #链接文字在页面中一般都是唯一的,所以这个方式可以精确查找到元素
返回元素

目的:点击百度首页左上角的新闻链接
#!/usr/bin/env python
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
#====#====#====#====

from selenium import webdriver
import time

dr=webdriver.Firefox()
```

```
dr.get("https://www.baidu.com")
time.sleep(2)
#通过链接文字获取元素
e=dr.find_element_by_link_text('新闻')
#点击元素
e.click()
time.sleep(3)
dr.quit()

Python是3.7版本以上使用以下代码
from selenium import webdriver
import time
from selenium.webdriver.common.by import By
dr=webdriver.Firefox()
dr.get("https://www.baidu.com")
time.sleep(2)
#通过链接文字获取元素
e=dr.find_element(By.LINK_TEXT,'新闻')
#点击元素
e.click()
time.sleep(3)
dr.quit()



#点击百度首页左上角7个链接文字
from selenium import webdriver
import time
dr=webdriver.Firefox()
dr.get("https://www.baidu.com")
time.sleep(2)
mylist=['新闻','hao123','地图','贴吧','视频','图片','网盘']
for i in mylist:
    e=dr.find_element_by_link_text(i)
    e.click()
    time.sleep(2)

time.sleep(2)
dr.quit()



#案例:获取淘宝首页天猫的链接
#案例:自己写一个有超链接的页面,通过链接文字获取链接
```

find_element_by_partial_link_text()函数

```
链接文本定位,不只<a></a>标签，也可以模糊查询，链接文字可以是一部分
1.在超链接定位里，输入字符串的子串也会返回第一个文本属性匹配的元素，如果没有元素匹配，会抛出NoSuchElementException异常。
```

```
2.函数说明:
find_element_by_partial_link_text("链接文字")  #链接文字在页面中一般都是唯一的,所以这个方式可以精确查找到元素
返回元素

目标:点击百度热搜的第一个新闻标题
#!/usr/bin/env python
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
#====#====#====#====
from selenium import webdriver
import time

dr=webdriver.Firefox()
dr.get("https://www.baidu.com")
time.sleep(2)
# e=dr.find_element_by_partial_link_text('韩国人将集体减龄一至两岁')
# e=dr.find_element_by_partial_link_text('韩国人')
# e=dr.find_element_by_partial_link_text('一至两岁')
e=dr.find_element_by_partial_link_text('将集体减龄')
e.click()
time.sleep(3)
dr.quit()




案例:点击百度首页中的百度热搜里全部的文字链接

```

总结

```
1.获取浏览器,并打开
dr=webdriver.Firefox()

2.打开页面
dr.get("url")

3.定位元素    左边是python3.6以下,右边是python3.7以上
find_element_by_id("元素的id值")   find_element(By.ID,"元素的id值")
find_element_by_name("元素的name值")   find_element(By.NAME,"元素的name值")
find_element_by_link_text("链接文字")   find_element(By.LINK_TEXT,"链接文字")
find_element_by_partial_link_text("链接文字")  find_element(By.PARTIAL_LINK_TEXT,'链接文字')

4.操作
time.sleep(秒数) #暂停
e.send_keys("信息") #往元素中输入信息
```

```
e.click() #点击元素
dr.quit()#退出浏览器


```

打开本地页面

```
1.打开本地页面需要引入os模块

2.本地html内容如下:
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <a href="https://www.baidu.com">百度一下</a>
</body>
</html>

3.获取本地页面元素,并操作元素
#!/usr/bin/env python
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
#====#====#====#====
from selenium import webdriver
import time
from selenium.webdriver.common.by import By
import os

dr=webdriver.Firefox()
#打开本地页面
dr.get("file:///"+os.path.abspath("hello.html"))
time.sleep(2)
# dr.find_element_by_link_text("百度一下").click()
dr.find_element(By.LINK_TEXT,'百度一下').click()
time.sleep(2)
dr.quit()
```