

软件测试方法与技术

自动化测试

本章目标

- 1)充分理解自动化测试的核心技术与实施价值；
- 2)充分理解 GUI 自动化测试技术的核心原理；
- 3)熟悉自动测试过程
- 4)熟悉自动化测试工具

什么是自动化测试？

- 自动化测试是把以人为驱动测试行为转化为机器执行的一种过程。通常，在设计了测试用例并通过评审之后，由测试人员根据测试用例中描述的规程一步步执行测试，得到实际结果与期望结果的比较。此过程中，为了节省人力、时间或硬件资源，提高测试效率，便引入了自动化测试的概念。
- 自动化测试，通常指功能自动化和性能自动化测试。这里仅只功能自动化测试。

自动化测试核心要素？

- 1) 定义某个功能点的测试步骤及期望结果；
- 2) 自动化的操作或调用被测对象来驱动测试执行；
- 3) 将实际结果与期望结果进行比较进而得出测试结论。

自动化测试实施过程？

- 1) 分析

需求分析、投入产出比分析、工具与技术分析、测试团队分析、风险分析

- 2) 设计

规范设计（ 流程、编码、版本控制 ）、 框架设计、用例设计

- 3) 实现

- 4) 执行

- 5) 维护

自动化测试可行性分析？

编号	待分析内容	分析结果		备注
1	自动化测试的目标客户是否明确？	是	否	目标客户群是谁？
2	目标客户的预期目标是否明确？	是	否	预期目标是什么？
3	自动化测试范围是否明确？	是	否	项目有多大？
4	公司管理层是否重视此事？	是	否	老板是否支持？
5	是否有足够资金投入并长期坚持？	是	否	老板是否肯掏钱？
6	能为公司带来直接收益吗？	能	不能	投入产出比是多少？
7	能提升测试团队的工作效率吗？	能	不能	能节省多少时间？
8	是否有合适的自动化测试工具？	是	否	什么工具？
9	是否有免费的自动化测试工具可用？	是	否	什么工具？开源吗？
10	是否可以不用培训就立即开展自动化？	是	否	如需要，培训什么？
11	是否需要从外部招聘自动化测试人员？	是	否	招聘要求是什么？
12	是否有技术上的问题没有解决的？	是	否	一一列出？
13	目前测试团队在测试方面经验是否丰富？	是	否	有哪些项目经验？
14	在实施时间上是否有要求？	是	否	实施时间是否太紧？
15	测试部门分工是否明确？	是	否	是否任何人做任何事？
16	被测产品的变更是否频繁？	是	否	界面级，业务级变更？
17	补测产品的研发周期长不长？	长	不长	对长期产品做投入？
18	是否有专门的团队支撑自动化测试实施？	是	否	测试的技术门槛并不低？

GUI自动化测试的价值或者为什么要学？

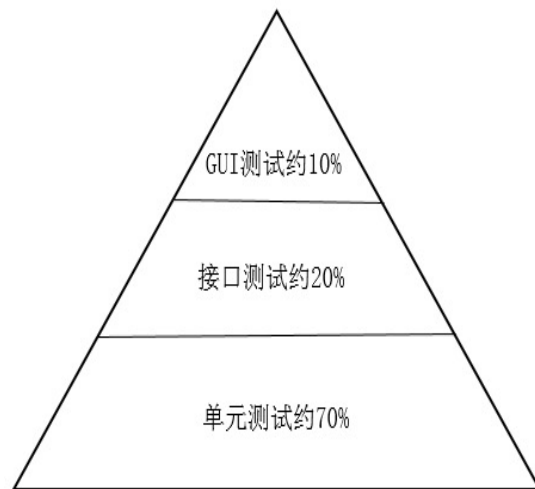
- 1) 行业趋势，大势所趋；
- 2) 将人从重复的工作中解放出来，让人来完成更加重要的测试设计工作而不是简单的执行；
- 3) 能做B/S、C/S和APP结果软件的UI自动化测试；
- 4) 可以节省人力、时间或硬件资源，提高测试效率；
- 5) 回归测试，快速监测新功能的引入或者缺陷的修复是否会影响到已有功能；
- 6) 兼容性测试，快速测试产品在不同平台，不同环境下的兼容情况；
- 7) 代替人完成手工无法执行的测试，比如多线程并发操作进行压力测试；
- 8) 减少不必要的调试时间和重复的测试工作；
- 9) 提升团队对于当前产品质量的信心，提升客户的信心。

GUI自动化测试与手工测试的关系？

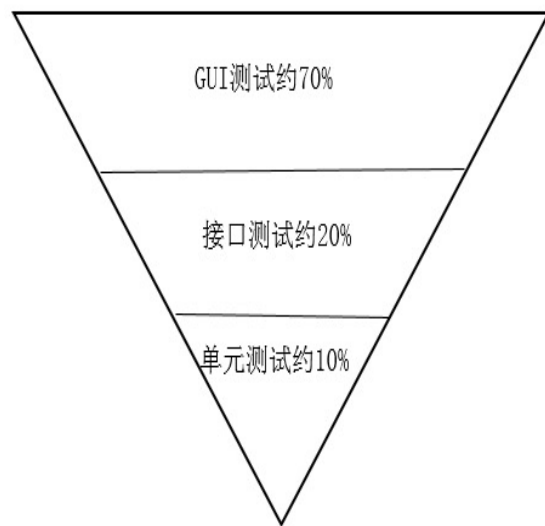
- 不能相互替代；
- 手工测试比自动测试发现的缺陷更多；
- 对产品质量的依赖性更强；
- 自动化测试的用例一般来源于手工测试用例；
- 自动化测试的目的不是为了发现BUG，而是为了验证没有BUG。

分层的自动化测试：GUI自动化测试所占比例？

- 理论比例：



- 实际比例：



自动化测试技术类别？

- 代码级自动化测试
- 协议级自动化测试
- 界面级自动化测试

GUI自动化测试几种工作原理？

- 1) 通过模拟按键操作和鼠标定位 (基于坐标)
- 2) 通过基于界面图像识别和定位 (基于图像)
- 3) 通过识别界面元素的核心属性 (基于元素)

什么样的项目适合做GUI自动化测试？

- 1) 测试需求明确，不会频繁变动；
- 2) 每日构建后的测试验证；
- 3) 比较频繁的回归测试；
- 4) 软件系统界面稳定，变动少；
- 5) 需要在多平台上运行的相同测试案例、组合遍历型的测试、大量的重复任务；
- 6) 软件维护周期长；
- 7) 项目进度压力不太大；
- 8) 被测软件系统开发比较规范，能够保证系统的可测试性；
- 9) 具备大量的自动化测试平台；
- 10) 测试人员具备较强的编程能力。

GUI自动化测试通常都有哪些工具？

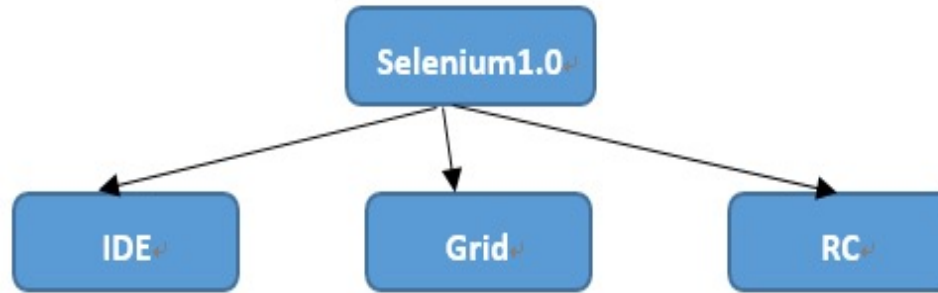
特性对比 工具名称	费用	适用软件架构	适用软件规模	支持语言	备注
sikulix	免费	B/S, C/S	小项目、简单项目	Python、js、Java	
katalon	免费	B/S	小项目、简单项目	Python、Java、C#等	
selenium	免费	B/S	几乎所有项目	Python、Java、C#等	推荐
uiautomation	免费	C/S	几乎所有项目	Python、Java、C#等	推荐

Selenium:简介

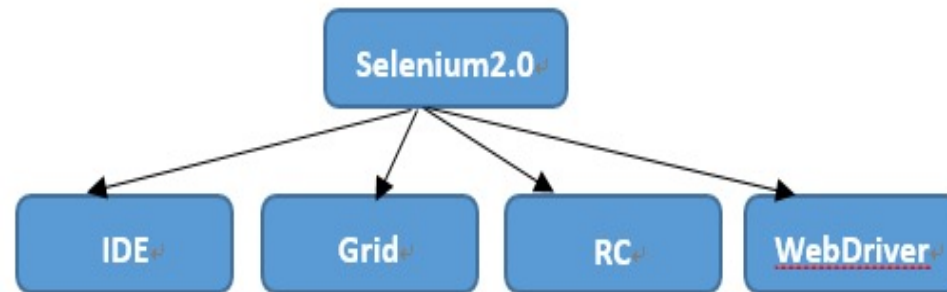
- 开源, 免费
- 多浏览器支持: Firefox, Chrome, IE, Opera
- 多平台支持: Linux, Windows, MAC
- 多语言支持: Java, Python, Ruby, php, C#, JavaScript
- 简单(API简单), 灵活(用开发语言驱动)

Selenium:版本区别

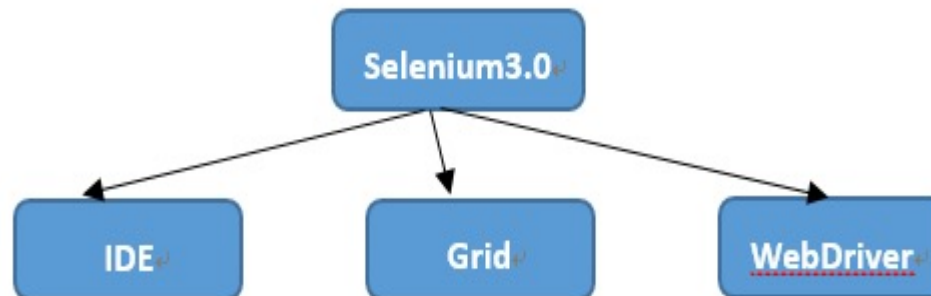
- V1.0:



- V2.0:



- V3.0:



Selenium:为什么要学习Webdriver?

- 动态网页中, 通常只会更新局部的Html元素, webdriver会很好的帮助用户快速定位这些元素
- 最终目的是通过提供精心设计的面向对象API来解决现代高级网页中的测试难题

Selenium:Webdriver的实现原理

- Webdriver启动目标浏览器, 并绑定到端口. 该启动的浏览器实例, 做为WebDriver的Remote Server
- Client端通过Commandexecutor发送HttpRequest给Remote Server的侦听端口(通信协议: The Webdriver Wire Protocol)
- Remote Server需要依赖原生的浏览器组件(如: chromedriver.exe, geckodriver.exe, IEDriverServer.exe), 来转化浏览器的Native调用

Selenium:安装配置

- 方式1：在线安装

`pip install selenium`

- 方式2：离线安装

通过提前下载好的selenium包解压到指定目录(例C:\Python36\Lib\site-packages)，然后运行命令`python setup.py install`即可使用。略

Selenium:浏览器原生驱动的使用

- 下载地址：<https://www.cnblogs.com/nancyzhu/p/8589764.html>
- 使用方式1：放置环境变量路径

例如将驱动文件直接放置到已配置好的python环境变量根路径。

```
dr = webdriver.Chrome()
```

```
dr = webdriver.Firefox()
```

```
dr = webdriver.Ie()
```

- 使用方式2：指定绝对路径

```
dr = webdriver.Chrome(executable_path="C:\\driver\\chromedriver.exe")
```

```
dr = webdriver.Firefox(executable_path="C:\\driver\\geckodriver.exe")
```

```
dr = webdriver.Ie(executable_path="C:\\driver\\IEDriverServer.exe")
```

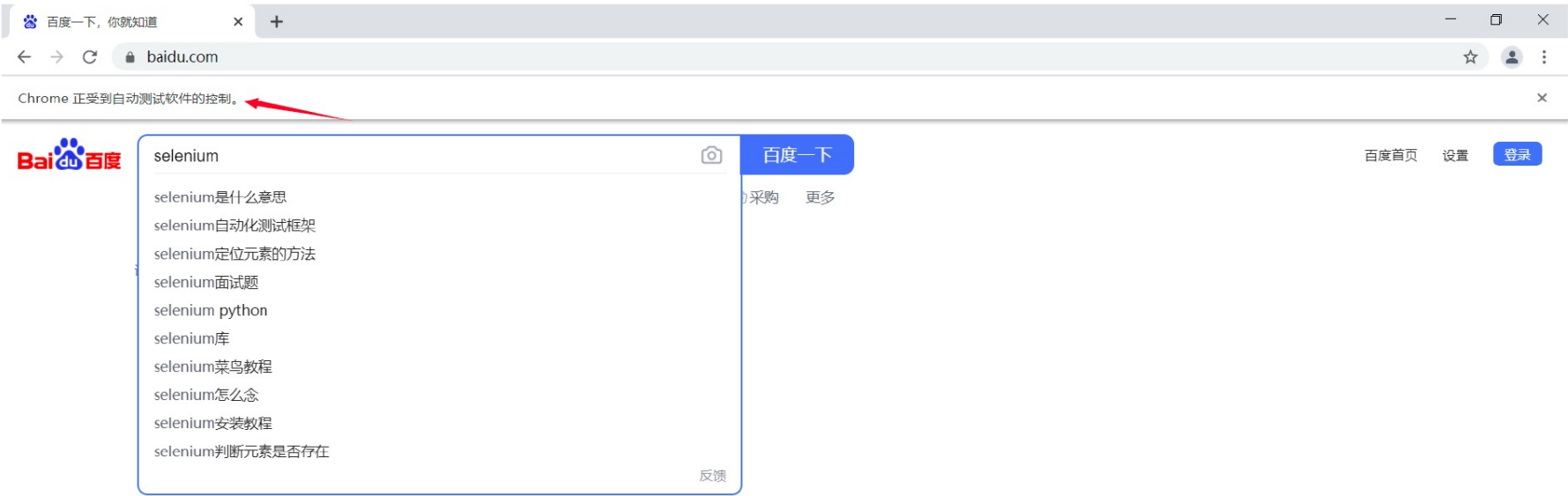
注：可用于浏览器兼容性测试。

Selenium:代码演示

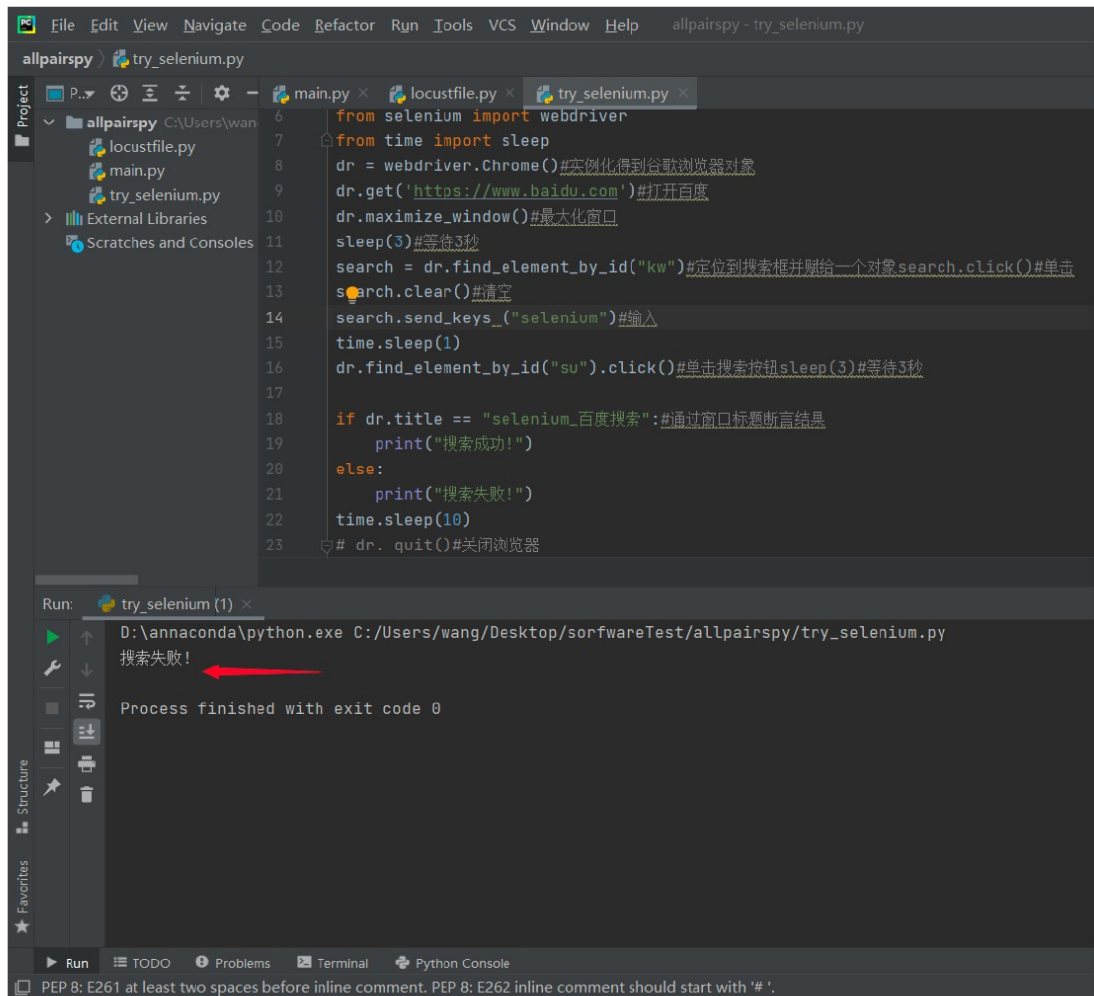
```
from selenium import webdriver
from time import sleep
dr = webdriver.Chrome()#实例化得到谷歌浏览器对象
dr.get('https://www.baidu.com')#打开百度
dr.maximize_window()#最大化窗口
sleep(3)#等待3秒
search = dr.find_element_by_id("kw")#定位到搜索框并赋给一个对象
search.click()#单击
search.clear()#清空
search.send_keys("selenium")#输入
dr.find_element_by_id("su").click()#单击搜索按钮
sleep(3)#等待3秒
if dr.title == "selenium_百度搜索":#通过窗口标题断言结果
    print("搜索成功！")
else:
    print("搜索失败！")
dr.quit()#关闭浏览器
```

测试过程与测试结果

测试过程



测试结果



The screenshot shows an IDE with a Python script named `try_selenium.py` and its execution output. The script attempts to open Baidu, search for 'selenium', and verify the page title. The output shows the search failed.

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help allpairsy - try_selenium.py
allpairsy > try_selenium.py
Project
  P...
  C:\Users\wan
  allpairsy
    locustfile.py
    main.py
    try_selenium.py
  External Libraries
  Scratches and Consoles
main.py x locustfile.py x try_selenium.py x
6 from selenium import webdriver
7 from time import sleep
8 dr = webdriver.Chrome() #实例化得到谷歌浏览器对象
9 dr.get('https://www.baidu.com') #打开百度
10 dr.maximize_window() #最大化窗口
11 sleep(3) #等待3秒
12 search = dr.find_element_by_id("kw") #定位到搜索框并赋给一个对象search.click() #单击
13 search.clear() #清空
14 search.send_keys_("selenium") #输入
15 time.sleep(1)
16 dr.find_element_by_id("su").click() #单击搜索按钮 sleep(3) #等待3秒
17
18 if dr.title == "selenium_百度搜索": #通过窗口标题断言结果
19     print("搜索成功!")
20 else:
21     print("搜索失败!")
22 time.sleep(10)
23 # dr.quit() #关闭浏览器

Run: try_selenium (1) x
D:\annaconda\python.exe C:/Users/wang/Desktop/sorftwareTest/allpairsy/try_selenium.py
搜索失败!
Process finished with exit code 0
```



Selenium:常用API之操作浏览器

#打开

```
dr = webdriver.Chrome()
```

```
dr.get('https://www.baidu.com')
```

#关闭

```
dr.close()#关闭所有打开的页面
```

```
dr.quit()#关闭当前页面
```

#窗口设置

```
dr.maximize_window()#最大化
```

```
dr.minimize_window()#最小化
```

```
dr.set_window_size(400,600)#设置指定大小
```

#后退、前进和刷新

```
dr.back()
```

```
dr.forward()
```

```
dr.refresh()
```

Selenium:常用API之元素定位

- 几种定位方式：

`dr.find_element_by_id()`

`dr.find_element_by_name()`

`dr.find_element_by_tag_name()`#标签名

`dr.find_element_by_link_text()`#链接文本

`dr.find_element_by_partial_link_text()`

`dr.find_element_by_class_name()`

`dr.find_element_by_css_selector()`

`dr.find_element_by_xpath()`

注：1.确保唯一属性的情况下，定位推荐使用顺序id-name-xpath-other；

2.定位一组具有相同属性的元素，例如：`dr.find_elements_by_name()`；

3.有时即便有id也不能通过id定位，因为它可能是动态id；

4.由于selenium使用xpath定位时采用遍历页面的方式，在性能上采用CSS选择器的方式更优。xpath虽然性能指标较差，但是在浏览器中有比较好的插件支持，定位元素比较方便，对于性能要求严格的场景，可考虑通过xpath改写css的方式进行替换。

Selenium:常用API之元素定位xpath补充

● xpath基础语法：

/：从根开始

//：从任意节点开始找

*：代表任意节点名称

@：代表属性

text()：节点文本

[]：通过下表定位

.：当前节点

..：父节点

Selenium:常用API之元素定位xpath补充

- xpath几种定位方式：

- 1) 全路径

`/html/body/table/tbody/tr[2]/td[2]/div/div[1]/div[2]/input`

- 2) 节点属性

`//input[@value="新建"]`

- 3) 节点文本

`//option[text()="AgileoneDemo"]`

- 4) 关联属性定位

`//input[@value="新建" and @type="button"]`

- 5) 父节点定位

`//table[@class="table-admin"]/thead[@class="bottomhr"]`

- 6) 通过下标定位

`//*[@id="dtrow_6"]/td[2]`

- 7) 模糊匹配属性

`//tr[contains(@id,'dtrow')]/td[2]`

Selenium:常用API之元素操作

`dr.find_element_by_name("tj_trnews").text`#获取文本

`dr.find_element_by_id("kw").click()`#单击

`dr.find_element_by_id("kw").send_keys("selenium")`#输入内容

`dr.find_element_by_id("kw").clear()`#清空输入内容

`dr.find_element_by_id("kw").get_attribute("name")`#获取属性值

`dr.find_element_by_id("kw").is_displayed()`#是否显示

`dr.find_element_by_id("kw").is_enabled()`#是否可用

`dr.find_element_by_id("kw").is_selected()`#复选框是否被选中

Selenium:常用API之元素操作

- 下拉列表选项处理

方法1：关联定位，也叫二次定位

```
dr.find_element_by_id("newitem").find_element_by_xpath("//*[@value='proposal']").click()
```

方法2：Select

```
from selenium.webdriver.support.select import Select
```

```
Select(dr.find_element_by_id("newitem")).select_by_value("proposal")#根据value选择
```

```
Select(dr.find_element_by_id("newitem")).select_by_visible_text("※ 需求提案 ※")#根据text选择
```

```
Select(dr.find_element_by_id("newitem")).select_by_index(3)#根据索引选择
```

思考：如何随机选择一个下拉列表值？

Selenium:设置等待时间

- 强制等待

```
from time import sleep
```

```
sleep(3)
```

- 隐式等待：针对所有元素

```
dr.implicitly_wait(30)
```

- 显示等待：针对指定元素

```
from selenium.webdriver.support.wait import WebDriverWait
```

```
element = WebDriverWait(dr,30,0.5).until(lambda dr: dr.find_element_by_id("msg"))
```

```
content = element.text
```

Selenium:窗口切换

- 自定义弹窗切换：可直接定位弹窗上的元素进行操作，无需切换
- 多页面(windows句柄)切换：

```
nowhandle = dr.current_window_handle#获得当前句柄
```

```
dr.find_element_by_xpath("//*[@id='1']/h3/a").click()#打开官网
```

```
allhandle = dr.window_handles#获得所有句柄
```

```
for handle in allhandle:
```

```
    if handle != nowhandle:
```

```
        dr.switch_to.window(handle)
```

```
        dr.find_element_by_id("poplogin").click()#点击登录
```

```
        dr.close()
```

Selenium:窗口切换

● alert、confirm和prompt弹窗切换

alert弹窗：

```
# alert("hello")
```

confirm弹窗：

```
# if(confirm("是否？")){ }
```

prompt弹窗：

```
# var info = prompt("请输入：")
```

```
# alert = dr.switch_to.alert#切换弹窗
```

```
# content = alert.text#获取弹窗内容
```

```
# alert.accept()#点击确定
```

```
# alert.dismiss()#点击取消
```

```
# alert.send_keys("你好！")#输入内容
```

● 切换frame窗口

```
dr.switch_to.frame(dr.find_element_by_id("FreeTextBox1_editor"))#方式1:如果没有frame id时可使用
```

```
dr.switch_to.frame("FreeTextBox1_editor")#方式2:推荐
```

```
dr.switch_to.default_content()#返回原页面
```

Selenium:文件上传

- 方式1：通过send_keys()

```
driver.find_element_by_id("batchfile").send_keys('D:\\woniue\\秦超\\教学\\UI自动化\\PiCiDaoRu.xls')
driver.find_element_by_xpath("//input[@value='确认导入本批次商品信息']").click()
```

- 方式2：通过PyKeyboard，需要依次安装pyHook和PyUserInput

```
from pykeyboard import PyKeyboard
```

try:

```
    driver.find_element_by_id("batchfile").click()#用firefox不行，chrome可以
except Exception as e:
```

```
    driver.find_element_by_xpath("//*[@class='col-lg-5 col-md-5 col-sm-5 col-xs-5']")[2]).click()
    sleep(3)
```

```
k = PyKeyboard()
```

```
k.type_string("E:\\study\\PycharmProjects\\python_3issue\\GUI\\PiCiDaoRu.xls")#不支持中文
```

```
k.press_keys([k.alt_key,'o'])#alt+o组合键点击确定
```

```
sleep(1)
```

```
driver.find_element_by_xpath("//input[@value='确认导入本批次商品信息']").click()
```

- 方式3：使用sikuli的jar包
- 方式4：其它，比如AutoIt

Selenium:鼠标事件

```
from selenium.webdriver.common.action_chains import ActionChains
```

```
right = dr.find_element_by_id("barcode") #定位到元素
```

```
ActionChains(dr).click(right).perform()#单击
```

```
ActionChains(dr).context_click(right).perform()#对定位到的元素执行鼠标右键操作
```

```
ActionChains(dr).double_click(right).perform()#双击
```

```
ActionChains(dr).move_to_element(right).perform()#鼠标悬停在一个元素上
```

```
ActionChains(dr).click_and_hold(right).perform()#按下鼠标左键在一个元素上
```

```
element = dr.find_element_by_name("xxx") #定位元素的原位置
```

```
target = dr.find_element_by_name("xxx") #定位元素要移动到的目标位置
```

```
ActionChains(dr).drag_and_drop(element, target).perform()#拖动
```

ActionChains(dr) : dr: webdriver 实例执行用户操作。ActionChains 用于生成用户的行为；所有的行为都存储在 ActionChains 对象。通过 perform()执行存储的行为。

perform() : 执行所有 ActionChains 中存储的行为。perform()同样也是 ActionChains 类提供的方法，通常与ActionChains()配对使用。

Selenium:键盘事件

```
from selenium.webdriver.common.keys import Keys

driver.find_element_by_id("barcode").send_keys("123456")

driver.find_element_by_id("barcode").send_keys(Keys.BACK_SPACE)#单击回删键

driver.find_element_by_id("barcode").send_keys(Keys.SPACE)#单击空格

driver.find_element_by_id("barcode").send_keys(Keys.ENTER) #通过回车键盘来代替点击操作

driver.find_element_by_id( "barcode" ).send_keys(Keys.DOWN) #单击向下键

#。 。 。 。 。 。

driver.find_element_by_id("barcode").send_keys(Keys.CONTROL,'a') #ctrl+a 全选输入框内容

driver.find_element_by_id("barcode").send_keys(Keys.CONTROL,'c')

driver.find_element_by_id("barcode").send_keys(Keys.CONTROL,'v')

driver.find_element_by_id("barcode").send_keys(Keys.CONTROL,'x')

#。 。 。 。 。 。
```

Selenium:JS注入

● 移动滚动条

#方式1：没有ID的滚动条不支持

```
js="var q=document.documentElement.scrollTop=10000"
```

```
# js_="var q=document.documentElement.scrollTop=0"
```

```
driver.execute_script(js_)
```

#方式2：拖动到指定元素

```
# target1 = dr.find_element_by_xpath("//*[text()='页顶']")
```

```
# dr.execute_script("arguments[0].scrollIntoView();", target1)
```

● 修改属性

```
dr.execute_script("document.getElementById('barcode').readOnly=true;")
```

```
dr.execute_script("document.getElementById('barcode').removeAttribute('readonly');")
```

思考：如何绕过Woniusales会员管理模块的出生日期限制进行手动输入测试？

● 其它

Selenium:验证码处理

- 方式1：屏蔽验证码
- 方式2：设置万能验证码
- 方式3：图像识别技术
- 方式4：添加cookie

```
driver.get("http://qzx:8080/WoniuSales1.4/")
```

```
driver.add_cookie({'name':'username', 'value':'admin'})
```

```
driver.add_cookie({'name':'password', 'value':'admin'})
```

```
driver.get("http://qzx:8080/WoniuSales1.4/")
```

- 方式5：其它

Selenium:截图

- 方式1：

```
import time
```

```
now = time.strftime("%Y-%m-%d-%H-%M-%S", time.localtime())
```

```
dr.get_screenshot_as_file(".\\screen\\" + now + "error_png.png")#推荐
```

- 方式2：

```
dr.save_screenshot(".\\screen\\" + now + "error_png.png")#遇到alert弹窗截图会报错，不推荐
```

- 方式3：

```
s = dr.get_screenshot_as_base64()#保存的是base64()格式的文件值，html测试报告里插入图片会用到
```

```
print("base64()格式:%s" %s)
```

- 方式4：

```
s = dr.get_screenshot_as_png()#保存二进制数据
```

```
print(s)
```