selenium===web UI自动化

课程安排

第一天：自动化测试理论，自动化测试环境搭建，selenium元素定位(8种方法)

第二天：元素操作（键盘输入，鼠标操作，浏览器操作，窗口切换，下拉框操作等）

第三天：unittest单元测试框架(作用：组织代码形成一条条的自动化用例，批量执行测试用例，断言，生产自动化测试报告)

1.自动化理论

1.什么是自动化测试?

手工测试：测试工程师手动操作被测试对象，执行测试用例，判断实际结果与预期结果，生成测试结果和报告。

自动化测试：测试工程师编写代码，让计算机运行代码，操作被测试对象，判断实际结果与预期结果(断言)，生成测试报告。

注意：自动化测试的思路与手工测试的思路是一样的，区别就是自动化测试是通过编写代码方式来完成。

2.为什么要做自动化测试?

1）提高测试的执行效率；

2）减少手工测试中的重复性工作；

3）可用于完成手工测试很难或无法实现的测试场景；

4）一定程度上节省企业的人力成本；

5）可用于在线产品的运行状态的监控；

6）可用于测试数据的生成；

3.自动化测试优点

1）提高测试效率，在较少的时间内执行更多的测试用例

2）极大的缩短回归测试时间

3）帮助完成一些手工测试完成不了的工作

4）更好利用资源：人力资源和时间资源。人力资源：自动化执行部分测试之后，就能够解放人力出来，去完成一些更要有创造性工作；时间资源：自动化代码可以在晚上，周末时间来执行。

5）可以重复执行，一致执行，可靠执行，避免人为错误。

4.自动化误区

1）自动化测试可以替代手工测试？不可以

（1）在项目初期，版本不稳定，不会做自动化测试，肯定都是先做手工测试；

（2）对于一些需求频繁变更的项目，不适合做自动化；

（3）对于易用性测试，需要主观判断的，是不能用自动化方式完成；

（4）自动化测试的成本较高，所以自动化测试会考虑覆盖率问题，不会对所有的功能都进行自动化，一般是选择主业务流程，主要功能，频繁使用功能，容易出错功能等进行自动化。UI自动化一般覆盖率30-50%之间，但是没有固定标准，各个公司不同，具体还是看公司投入。接口自动化一般覆盖率可以到90%以上。

2）自动化测试比手工测试厉害？不会

自动化测试跟手工测试的思路是一样的，区别只是通过代码方式效率会更高。

3）自动化测试能发现更多的bug？不会

自动化测试只是帮我们去确认之前测试正常的功能现在是不是仍然正常。

4）自动化测试适用于所有的功能？不是

验证码问题，在自动化测试中，如果出现验证码，一般常规的方式是请开发协助屏蔽测试环境的验证码或者设置完成验证码。当然如果绕不过验证码，对于一些图片验证码也可以是ocr技术来完成，但是有识别率问题，以及技术难度问题。

5.自动化测试分类

1）UI自动化测试

web ui自动化测试====selenium

app ui自动化测试====appium

2）接口自动化测试

1）requests===写代码

2）jmeter+ant+jenkins===工具，有界面

6.主流自动化测试工具（web UI自动化）

1）selenium

Selenium是专门为Web应用系统编写的一个验收测试工具。它的主要功能包括了2个方面：

- 兼容性测试——可以测试Web应用程序是否可以很好的工作在不同浏览器和操作系统上
- 功能测试——可以测试Web应用程序的功能，创建回归测试检验软件的功能和用户需求

特点：

- 轻量级（2M），开源免费
- 多浏览器支持：Firefox，Chrome，Edge，Opera，Safari.....
- 多平台支持：Linux/Windows/Mac
- 多语言支持：Java/Python/Ruby/JS/C++
- 简单、灵活

2）QTP(UFT)

商业收费工具，功能强大，重量级工具（2G），可以做web自动化，也可以做桌面程序（PC端的C/S的客户端），只支持VBScript语言。

3）RF

robot framework一个基于关键字驱动的框架。

7.自动化测试开展的条件?

什么项目适合做自动化?

对于需要项目迭代，版本比较稳定，需求变更不频繁项目，项目周期足够长

什么时候可以做自动化?

功能测试完成，项目比较稳定，没有太多bug，有时间就可以安排做自动化。

2.自动化环境搭建

1.代码编写环境===python语言

python3.7===编译器，翻译代码，让计算机可以执行

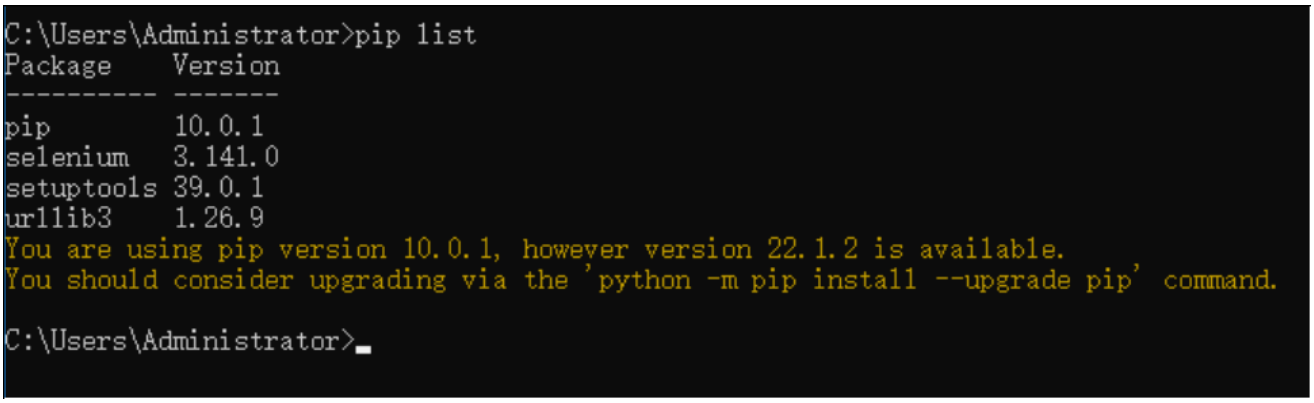
pycharm2017====开发工具，方便开发人员编写，调试运行代码

2.selenium安装

selenium是python一个第三方库，直接使用pip工具就可以在线安装。

在线安装方式：打开cmd，输入命令：pip install selenium==3.141.0 （安装完成后，如果没有红色报错就ok。）

验证是否安装成功：输入pip list，查看是否有selenium，如果有就是安装成功。如下：



3.浏览器驱动

selenium对于浏览器的操作，需要一个浏览器的驱动文件才可以进行，不同的浏览器的驱动文件是不同的。

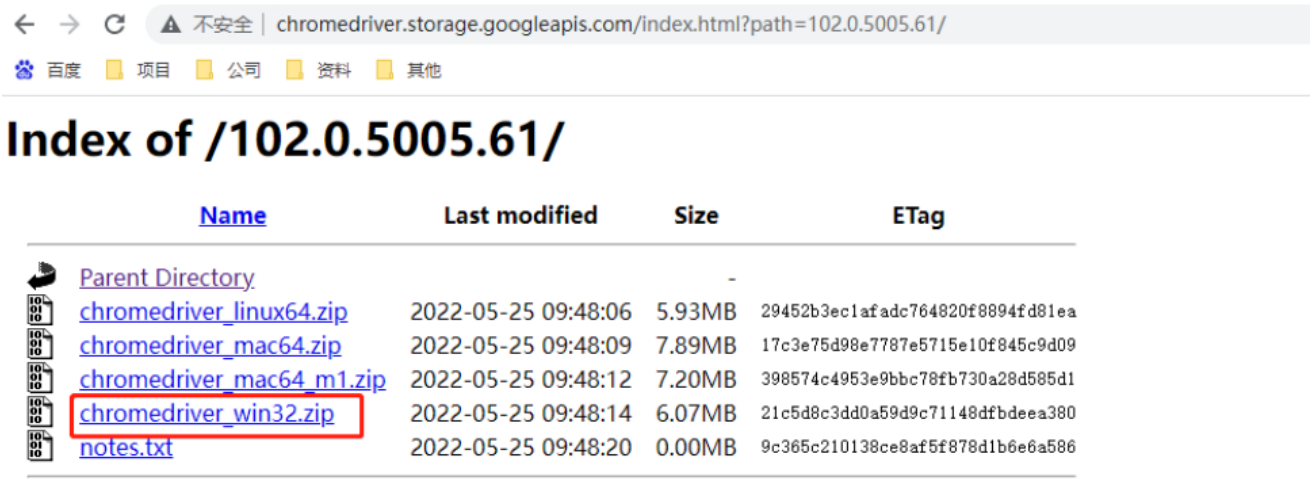
在自动化测试中，推荐是用chrome浏览器。

1.下载chrome浏览器驱动文件

chrome浏览器的驱动文件：chromedriver.exe

下载地址：http://chromedriver.storage.googleapis.com/index.html

下载驱动文件时注意：下载的驱动文件的版本需要与自己电脑上chrome浏览器的版本对应（大版本一致，小版本靠近就可以了），否则可能无法打开浏览器。



2.解压chromedriver_win32.zip，将里面的chromedriver.exe文件复制以下路径：

（1）如果是win10系统，复制到python安装目录。

在cmd，输入where python命令，可以查看python安装目录

（2）如果是win7系统，复制到chrome浏览器的安装目录。

在chrome浏览器桌面快捷方式上右键打开文件所在位置即可打开浏览器的安装目录。

4.使用pycharm编写第一个自动化程序

```
#导入 从我们自己安装的selenium库中导入webdriver模块(文件)==因为所有基于浏览器的操作的api都被封装在webdriver中
from selenium import webdriver
#导入 从python内置模块time模块中导入sleep函数==可以让代码暂停执行
from time import sleep
# 创建webdriver模块中Chrome类的对象driver==实际就是打开一个chrome浏览器
driver=webdriver.Chrome()
# 通过浏览器对象driver来调用类中get方法===访问一个网页
driver.get('http://www.baidu.com')
# 通过浏览器对象driver来调用类中find_element_by_id方法==通过标签的id属性值来去定位到对应的标签元素==百度页面上id='kw'的百度输入框
# 然后调用send_keys方法===模拟键盘输入
driver.find_element_by_id('kw').send_keys('hello')
# 通过浏览器对象driver来调用类中find_element_by_id方法==通过标签的id属性值来去定位到对应的标签元素==百度页面上id='su'的百度一下按钮
# 然后调用click方法===模拟鼠标点击
driver.find_element_by_id('su').click()
# 调用sleep函数 睡眠5秒  等待浏览器的执行
sleep(5)
# 通过浏览器对象driver来调用类中quit方法===关闭退出浏览器
driver.quit()
```

3.元素定位

在网页上定位到你想要操作的元素，是自动化的第一步，也是很容易出错的一步。

在selenium工具中，提供了8种元素定位的方法，我们需要根据元素自身特点来选择合适的方法。

在网页上利用开发者工具(F12)中elements页中查看网页的元素信息，可以利用小箭头先手动定位到元素，观察元素具体的特征来选择定位方法。

在selenium中定位方法有两大类:

find_element_by_XX() ==>只返回定位到的第一个元素，大部分情况是用这种

find_elements_by_XX()==>返回所有找到的元素，以列表形式返回

1）id定位

通过元素id属性来定位元素。

方法：find_element_by_id(元素id值)

```
#1.id定位   ==宝贝计划登录
#定位用户名输入框
driver.find_element_by_id('userName').send_keys('admin')
#定位密码输入框
driver.find_element_by_id('password').send_keys('admin11')
#点击登录按钮
driver.find_element_by_id('loginbtn').click()


#练习：1.在乐鲜，注册一个新用户
driver.get('http://172.30.63.23:8080/LexianMall/sc/register.html')
#输入注册信息
driver.find_element_by_id('phone').send_keys('13378901234')
driver.find_element_by_id('username').send_keys('小王')
driver.find_element_by_id('password').send_keys('123456')
driver.find_element_by_id('surepassword').send_keys('123456')
driver.find_element_by_id('submit').click()
```

如果元素有id属性情况下，首选用id定位，简单，精确。但是如果id是动态的（页面多刷新几次，每次id都不同，就是动态id），使用其他方法。

2）name定位

通过元素name属性来定位元素。

方法：find_element_by_name(元素name值)

```
driver.find_element_by_name('userName').send_keys('admin')
driver.find_element_by_name('password').send_keys('admin11')
driver.find_element_by_id('loginbtn').click()
```

注意：由于name属性可以重复，所以如果name有重复的情况下，建议使用其他方法。如果对于name重复的标签，可以使用find_elements_by_name()找到所有该name值的标签，返回为一个列表，再通过下标取值，找到需要找到的那个标签。

```
#重复name情况
driver.get("http://14.29.255.158:8081/LexianMall/sc/login.html")
#登录
driver.find_element_by_id('username').send_keys('13539984654')
driver.find_element_by_id('password').send_keys('123456')
driver.find_element_by_id('submit').click()
sleep(2) #添加等待，等待浏览器加载完成首页
#点击个人中心
driver.find_element_by_link_text('个人中心').click()
sleep(2) #等待页面跳转
#修改性别为女  由于性别的单选框name属性=sex 重复的，所以用find_elements_by_name
driver.find_elements_by_name('sex')[1].click()
sleep(1)
#保存
driver.find_element_by_id('sumbit').click()
```

3）class定位

通过元素class属性来定位元素。

方法：find_element_by_class_name(元素class值)

```
# 宝贝计划登录
# 说明：class='col-md-4 form-control' class值中有空格。表示复合类，需要将空格用.来代替
driver.find_elements_by_class_name('col-md-4.form-control')
[3].send_keys('admin')
driver.find_elements_by_class_name('col-md-4.form-control')
[4].send_keys('admin11')
driver.find_element_by_id('loginbtn').click()
```

注意：

1.由于class属性在页面中重复较多，所以一般使用不太多。如果遇到重复的class，可以使用find_elements_by_class_name()

```
driver.find_elements_by_class_name('col-md-4.form-control')[3]
```

2.当class是复合类时候，就将类名中的空格换为点 比如当class='col-md-4 form-control'

```
driver.find_elements_by_class_name('col-md-4.form-control')[3]
```

4）链接定位

使用链接的文本信息来进行定位。注意：此方法只能用来定位链接（a标签）。

方法：find_element_by_link_text(链接的文本)

```
driver.find_element_by_link_text('登录').click()
```

5）部分链接定位

使用的是链接的部分文本来进行定位。注意：此方法只能用来定位链接。

方法：find_element_by_partial_link_text(链接的部分文本)

```
driver.find_element_by_partial_link_text('登').click()
```

6）标签定位

使用标签名来定位元素。在实际工作中，此方法基本不太用，因为页面中重复元素太多。

方法：find_element_by_tag_name(元素的标签名)

```
driver.find_element_by_tag_name('select')
```

7）xpath定位

使用元素的xpath路径来进行元素定位。比较多用，因为基本绝大部分元素都可以定位到，适用性比较高。

xpath可以直接从浏览器中复制出来，使用方便。

方法：find_element_by_xpath(元素的xpath路径)

```
driver.find_element_by_xpath('//*[@id="userName"]').send_keys('admin')
driver.find_element_by_xpath('//*[@id="password"]').send_keys('admin11')
driver.find_element_by_xpath('//*[@id="loginbtn"]').click()
```

xpath：描述html这种树型结构文档中节点路径的描述语言

xpath有两种路径描述方法：

绝对路径：以/开头，从html根节点开始写起，一级一级往下写，不能跨级

/html/body/div[2]/form/div[1]/input 其中div[2]表示在同级中的第二个div标签

缺点：

1.比较长，可读性差；

2.对于前端的依赖性比较强，可能前端的改动会导致xpath失效，所以灵活性不好。

相对路径：以//开头，可以从任意节点开始写起，可以跨级

一般格式： //标签名[@属性='属性值'] ==》表示从任意级，找到指定的标签名，并且指定的属性=属性值

其中标签名可以使用通配符星号，表示找到任意的标签

//*[@id="userName"] ==》在任意级找到任意的标签，但是该标签的id属性值为userName

//*[@id="s-top-left"]/a[1] ==》找到任意级的任意标签，该标签的id='s-top-left'，然后再该标签的下一级找到第一个a标签

//a[@href='http://news.baidu.com'] ==》找到任意级的a标签，该标签的href='http://news.baidu.com'

//a[contains(@href,'news.baidu')] ==》找到任意级的a标签，该标签的href包含new.baidu

//*[contains(@id,'username')] ==》找到任意级的任意标签，该标签的id包含username ==》这种方法可以用来处理动态id的情况

8）css定位

通过css选择器来定位元素。比较多用，因为基本绝大部分元素都可以定位到，适用性比较高。比xpath好的地方，比xpath简洁，并且效率高，但是语法比xpath学习难度高。

css选择器可以直接从浏览器中复制出来。

方法：find_element_by_css_selector(元素css选择器)

```
driver.find_element_by_css_selector('#userName').send_keys('admin')
driver.find_element_by_css_selector('#password').send_keys('admin11')
driver.find_element_by_css_selector('#loginbtn').click()
```

总结：

一.如何选择元素定位方法?

1.如果元素有id属性，首选使用id定位，但是id动态生成的情况除外；

2.如果没有id属性，有name属性，可以使用name定位，但是注意name的情况；

3.如果是链接元素，可以直接使用链接定位，更加方便；

4.其他情况，可以使用xpath或者css定位。

二.元素定位失败可能有哪些原因?

NoSuchElementException 找不到元素

ElementNotInteractableException: Message: element not interactable 元素不可交互

1.检查代码中定位方法和传入的值是否一致，比如用id定位，需要传入id值；

2.如果使用id定位方法，检查id是否是动态id；多刷新几次页面，看id是否不同。

3.如果使用name或者class定位方位，检查是否有重复的name或者class；如果是重复的，使用find_elements来定位；

4.页面未加载处理，也会定位失败，需要添加等待；比如点击按钮页面跳转，比如点击按钮查询……

5.如果是元素在新的窗口，需要切换到新的窗口；

6.如果元素在框架中，需要先切换框架。

三.自动化测试如何从0到1?

1.自动化测试可行性评估（考虑项目周期，考虑项目需求是否确定，项目版本比较稳定，项目是否需要迭代，项目人力资源……）

2.编写测试计划，需要规定人员时间安排，测试范围，测试工具，编程语言，测试框架等；

测试范围：一般选择主业务流程，主功能，使用频繁功能等。

3.搭建自动化测试的框架；

对于web项目UI自动化框架：

python+selenium+unittest+HTMLTestRunner+PO模型（代码分层）+jenkins(持续集成)

4.准备符合要求的手工测试用例（主要是在编写自动化用例时参考）

5.根据测试计划，来分工完成自己负责的自动化用例(代码)；

6.将代码上传(svn，git)到代码服务器，批量执行，生成一份html测试报告；

7.集成jenkins，实现无人值守。

4.元素常用操作

1）键盘输入

方法：send_keys(输入的内容)

```
driver.find_element_by_css_selector('#userName').send_keys('admin')
```

2）鼠标点击(鼠标左键单击)

方法：click()

```
driver.find_element_by_css_selector('#loginbtn').click()
```

3）获取元素文本信息

方法：元素.text

示例：获取元素的text属性值（就是页面上看到的该标签的文本信息）保存在变量msg中备用

```
msg=driver.find_element_by_xpath('/html/body/form[1]/nav/div[1]').text
```

示例：

```
#获取元素文本内容
driver.find_elements_by_class_name('col-md-4.form-control')
[3].send_keys('admin')
driver.find_elements_by_class_name('col-md-4.form-control')
[4].send_keys('admin11')
driver.find_element_by_id('loginbtn').click()
sleep(2)#等待页面跳转到首页
#获取首页左上角的欢迎信息文本
msg=driver.find_element_by_xpath('/html/body/form[1]/nav/div[1]').text
#判断实际结果和预期结果是否一致
if msg=='admin'+', 欢迎来到宝贝计划在线教育后台管理! ':
# if 'admin' in msg: #登录的账号包含在左上角欢迎信息中
    print('测试用例通过')
else:
    print('测试用例不通过')
```

4）文本框清除

方法：clear()

```
driver.find_element_by_id('kw').clear()
```

5.其他操作

1.键盘操作

方法：send_keys(输入的内容)

键盘上除了普通字符键以外，还有一些功能键，这些功能键的输入需要导入Keys类。

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

```
driver=webdriver.Chrome()
driver.get("http://www.baidu.com")
#在百度输入框输入搜索关键字
driver.find_element_by_id('kw').send_keys('hello')
#输入回车键(enter)来完成搜索
driver.find_element_by_id('kw').send_keys(Keys.ENTER)

sleep(3)
#输入组合键  在百度输入框 输入ctrl+a(全选)
driver.find_element_by_id('kw').send_keys(Keys.CONTROL,'a')
```

练习：

```
#练习：1.在宝贝计划，输入用户名，输入密码，回车键登录
driver.get('http://172.30.63.23:8080/BabyPlan/login.jsp')
driver.find_element_by_id('userName').send_keys('admin')
driver.find_element_by_id('password').send_keys('admin11')
driver.find_element_by_id('password').send_keys(Keys.ENTER)
```

2.鼠标操作

鼠标的右键，双击，悬停操作，需要导入ActionChains类。

```
#需要导入ActionChains
from selenium.webdriver import ActionChains
```

1.双击操作

```
#1.双击操作
# (1)先定位到需要双击的元素
e1=driver.find_element_by_name('DoubleClick')
# (2)创建ActionChains类的对象ac,初始化对象时需要传入driver
ac=ActionChains(driver)
# (3)通过对象ac调用类中的双击方法double_click(),需要传入双击的元素
#最后调用perform()方法来完成执行
ac.double_click(e1).perform()

# 以上三句合并成一句,也可以写成
ActionChains(driver).double_click(driver.find_element_by_name('DoubleCli
ck')).perform()
```

2.悬停操作

```
#2.悬停操作
#1）定位需要悬停的元素
e2=driver.find_element_by_name('MoveTo')
#2)创建ActionChains类的对象
ac=ActionChains(driver)
#3)通过对象调用悬停方法move_to_element(),需要传入悬停的元素
#最后调用perform()方法来执行
ac.move_to_element(e2).perform()
```

3.右键操作

```
#3.右键操作
#1）定位需要右键的元素
e3=driver.find_element_by_id('box')
#2)创建ActionChains类的对象
ac=ActionChains(driver)
#3)通过对象调用右键方法context_click(),需要传入右键的元素
#最后调用perform()方法来执行
ac.context_click(e3).perform()
```

练习:

```
#练习：3.进入百度，在设置上悬停，点击搜索设置，选择搜索框提示为不显示，然后保存
driver.get('http://www.baidu.com')
#在设置上悬停
e=driver.find_element_by_id('s-usersetting-top')
ac=ActionChains(driver)
ac.move_to_element(e).perform()
#点击搜索设置
driver.find_element_by_xpath('//*[@id="s-user-setting-
menu"]/div/a[1]/span').click()
sleep(1)
#点击搜索框提示为不显示
driver.find_element_by_id('s1_2').click()
#点击保存设置
driver.find_element_by_xpath('//*[@id="se-setting-7"]/a[2]').click()
```

3.浏览器操作

浏览器最大化，最小化，设置窗口大小，刷新，前进，后退，关闭，访问网页等。

```
#打开浏览器
driver=webdriver.Chrome()
#1.访问网页
driver.get('http://www.baidu.com')

#2.窗口最大化
driver.maximize_window()

#3.设置窗口大小 宽500px 高300px
driver.set_window_size(500,300)

#4.刷新
driver.refresh()

#5.后退
driver.back()

#6.前进
driver.forward()

#5.退出
driver.quit()
```

获取浏览器标题和url，来完成判断

```
# 示例：在百度上搜索，判断搜索结果 ，给出测试用例结论
driver.find_element_by_id('kw').send_keys('hello')
```

```
driver.find_element_by_id('su').click()
sleep(2)
#获取当前浏览器的标题和url信息
#获取浏览器标题
a=driver.title
#获取浏览器的url
b=driver.current_url
if a=='hello_百度搜索' and 'hello' in b:
    print('测试用例通过')
else:
    print('测试用例不通过')

# 4.宝贝计划登录，判断标题和url，给出结果
driver.get('http://172.30.63.23:8080/BabyPlan/login.jsp')
#定位用户名输入框
driver.find_element_by_id('userName').send_keys('admin')
#定位密码输入框
driver.find_element_by_id('password').send_keys('admin11')
#点击登录按钮
driver.find_element_by_id('loginbtn').click()
sleep(2)
#判断浏览器标题和url
if driver.title=='管理员主页' and
driver.current_url=='http://172.30.63.23:8080/BabyPlan/admin/admin_index
.jsp':
    print('通过')
else:
    print('不通过')
```

4.多窗口切换

在自动化运行过程中，我们最开始打开的窗口时当前窗口，只能在当前窗口下操作，如果中间打开了新的窗口，需要到新的窗口操作，需要进行窗口的切换，切换到新窗口之后才可以操作。

句柄（handle）：操作系统分类给浏览器的每一个窗口一个唯一的编号，用来识别不同的窗口。

1.获取所有窗口句柄

```
#获取所有窗口句柄==以列表形式存储
handles=driver.window_handles
```

说明：在driver对象的window_handles属性存储了所有窗口的句柄，以列表的形式来存储的。

2.切换窗口

可以从所有窗口句柄的列表中通过下标取值来得到想要的窗口的句柄。

```
driver.switch_to.window(新窗口句柄)
```

示例：

```
driver.switch_to.window(handles[1])
```

```
driver=webdriver.Chrome()
driver.get("http://www.baidu.com")

#点击新闻链接
driver.find_element_by_link_text('新闻').click()
driver.find_element_by_link_text('hao123').click()
sleep(10)

#如果浏览器窗口太多，分不清是哪个句柄
for i in driver.window_handles:  #在所有句柄列表中循环，依次取出每一个句柄
    #切换到每一个句柄
    driver.switch_to.window(i)
    #切换之后判断页面是否时预期页面
    if driver.title=='百度新闻——海量中文资讯平台': #如果是预期的页面，就不继续切换，中断循环
        break
#切换完之后，打印页面标题
print(driver.title)
#在新窗口==新闻窗口上进行搜索
driver.find_element_by_id('ww').send_keys('hello')
driver.find_element_by_id('s_btn_wr').click()
```

5.弹框操作

对于浏览器的弹框是用javascript来定义的，不是html元素，需要做特殊的操作。

对于弹框操作：

1）获取弹框的文本信息

```
driver.switch_to.alert.text
```

2）点击弹框上的确定按钮

```
driver.switch_to.alert.accept()
```

3）点击弹框上的取消按钮

```
driver.switch_to.alert.dismiss()
```

4）弹框上的输入框输入内容

```
driver.switch_to.alert.send_keys(输入内容)
```

练习：

```
#练习：1.宝贝计划，错误密码登录，判断弹框上的文本内容，给出测试用例结果
driver.get('http://172.30.63.23:8080/BabyPlan/login.jsp')
driver.find_element_by_id('userName').send_keys('admin')
driver.find_element_by_id('password').send_keys('admin11123123')
driver.find_element_by_id('loginbtn').click()
#判断弹框内容
try: #加入异常处理，保证自动化代码健壮性，及时没有弹框弹出来，代码也不会报错
    msg=driver.switch_to.alert.text
    if msg=='用户名或密码错误！':
        print('通过')
    else:
        print('不通过')
    driver.switch_to.alert.accept()
except:
    print('不通过')
```

6.获取元素的属性信息

在测试过程中，可以获取元素的属性信息来做一些判断。

方法：元素.get_attribute(属性名)

```
driver=webdriver.Chrome()
driver.get(r'D:\work\我的课程\selenium\课件\index.html')

#选择排球复选框
driver.find_element_by_xpath('/html/body/table/tbody/tr[7]/td[2]/input[3
]').click()

#判断是否选中了
#可以判断复选框的checked属性的值
#获取复选框的checked属性值
msg=driver.find_element_by_xpath('/html/body/table/tbody/tr[7]/td[2]/inp
ut[3]').get_attribute('checked')
if msg=='true':
    print('通过')
else:
    print('不通过')
```

```
# 2.宝贝计划，空密码登录，判断密码输入框的validationMessage属性。给出测试用例结果
driver.get('http://172.30.63.23:8080/BabyPlan/login.jsp')
# #定位用户名输入框
driver.find_element_by_id('userName').send_keys('admin')
#定位密码输入框
# driver.find_element_by_id('password').send_keys('admin11')
#点击登录按钮
driver.find_element_by_id('loginbtn').click()
# 判断密码输入框的validationMessage属性    validationMessage属性的作用思存储了表单的校验结果信息
msg=driver.find_element_by_id('password').get_attribute('validationMessage')
if msg=='请填写此字段。':
    print('通过')
else:
    print('不通过')
```

7.切换框架

当元素在框架内，需要先切换进框架，才能对框架内html中的元素进行操作。

1）切换到框架

```
driver.switch_to.frame(框架的标识)
```

框架的标识可以使用frame/iframe标签的id值，name值，或者索引(0,1,2)，或者是frame标签对象本身。

```
#方法1：使用name属性来切换框架
driver.switch_to.frame('manage')
#方法2：使用索引来切换框架
driver.switch_to.frame(0)
```

如果框架没有id，name属性，也不想用索引，也可以直接定位frame标签元素本身，传进去也可以进行切换。

```
#方法3：直接使用frame标签对象本身
e=driver.find_element_by_tag_name('iframe')
driver.switch_to.frame(e)
```

2）切换回主页面

在切换进框架后，只能定位框架内的html中的元素，如果想要操作框架外主页面的元素，需要切换回主页面。

```
driver.switch_to.default_content()
```

示例：

```
# 3.宝贝计划，进入管理员后台，新添加一个用户，使用新用户登录，判断是否登录成功，给出测试用例结果
# 思考：添加新用户时，如何让用户名不重复
driver.get('http://172.30.63.23:8080/BabyPlan/login.jsp')
#定位用户名输入框
driver.find_element_by_id('userName').send_keys('admin')
#定位密码输入框
driver.find_element_by_id('password').send_keys('admin11')
#点击登录按钮
driver.find_element_by_id('loginbtn').click()
sleep(1)
#添加一个新用户
#点击添加用户按钮  由于该按钮在框架内，先切换框架
driver.switch_to.frame('manage')
driver.find_element_by_id('addUser').click()
sleep(1)
#输入用户信息  添加用户信息页面元素不在框架内，切换出框架
driver.switch_to.default_content()
#使用python中内置模块random，产生随机数，在用户名中，保证用户名的不重复
username='user_'+str(random.randrange(10000,99999))
driver.find_element_by_id('addName').send_keys(username)
driver.find_element_by_name('password').send_keys('123456')
driver.find_element_by_name('email').send_keys('123@qq.com')
driver.find_element_by_name('phone').send_keys('13312341234')
driver.find_element_by_xpath('//*
[@id="addUser"]/div/div/div[2]/button[1]').click()
sleep(1) #等待跳转回列表页
#点击注销，回到登录页
driver.find_element_by_link_text('注销').click()
sleep(1)#等待跳转回登录页
#使用新用户登录
#定位用户名输入框
driver.find_element_by_id('userName').send_keys(username)
#定位密码输入框
driver.find_element_by_id('password').send_keys('123456')
#点击登录按钮
driver.find_element_by_id('loginbtn').click()
sleep(1)
if driver.title=='宝贝计划在线教育' and
driver.current_url=='http://172.30.63.23:8080/BabyPlan/index.jsp':
    print('通过')
else:
    print('不通过')
```

8.下拉框操作

对于网页上的下拉框(select标签)，需要导入Select类。

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

```
#年龄下拉框
#需要Select类导入
#1)定位到select下拉框标签元素
e2=driver.find_element_by_name('babyAge')
#2)创建Select类的对象,传入下拉框元素
s=Select(e2)
#3)通过对象s调用类中选择下拉选项的方法
#（1）通过选项的索引 从0开始
s.select_by_index(2)
sleep(2)
#（2）通过选项的文本来选择
s.select_by_visible_text('4')
sleep(2)
#（3）通过选项的value属性来选择
s.select_by_value("5")
```

9.执行js

对于有一些浏览器的操作，在selenium中没有封装对应的方法，但是js可以试想相应的操作，那么可以通过执行js代码的方式来间接实现对浏览器操作。

```
#1.定义js代码==拖动滚动条  scrollTo(x,y)  js中的内置方法，拖动滚动条到某个位置
script='window.scrollTo(0,200)'
#2.执行js代码
driver.execute_script(script)
```

10.网页截图

在执行自动化用例时，由于无人值守，那么可以在执行错误的时候对浏览器网页进行截图保存，方便后续查看具体页面报错信息。

```
driver.get_screenshot_as_file('d:\\img1.png')  #传入图片文件保存的路径
```

3.unittest

1）unittest基本用法

2）断言

断言：在自动化测试中，判断实际结果与预期结果，这个过程叫做断言。

在unittest中，封装了断言方法。常用的有三种：

1）相等断言

assertEqual(a，b) 判断a和b是否相等 如果a等于b，断言成功，用例通过，否则断言失败，用例不通过

2）包含断言

assertIn(a，b) 判断b是否包含a 如果b包含a，断言成功，用例通过，否则断言失败，用例不通过

3）为真断言

assertTrue(a) 判断a是否为真，如果为真，断言成功，用例通过，如果为假，断言失败，用例不通过

3）自动化项目架构及生成报告

4.元素等待

在UI自动化测试中，需要等待页面元素加载出来，才可以进行元素定位，否则会定位报错。

三种等待方式：

1）强制等待

特点：

1.简单，等待时间是固定

2.会受到环境影响，可能代码健壮性不高，可能会由于等待时间不足造成元素定位失败

2）隐性等待

特点：

1.等待时间是设置的最大等待时间，当加载快时，加载完就继续执行，不会等待剩余时间。

2.等待整个网页加载，只用写一次可以等待整个页面的所有元素。

3.很多时候页面上会有一些元素加载的很慢，但是这些元素不是我们需要等待的元素，所以会造成时间浪费。

3）显性等待

也叫智能等待。

特点：

1.只等待指定元素。

2.等待时间是设置的最大等待时间。

3.在实际应用比较多，有一些特殊元素必须要用显性等待，另外如果使用显性等待可以增加代码健壮性。

driver.implicitly_wait(最大等待时间)显性等待是通过调用WebDriverWait类来实现的，在实际使用过程中会封装一个智能等待方法，在使用时直接调用，更加方便。

封装智能等待代码：==已经封装好在webDriverWait.py，直接放在python安装目录Lib文件夹下面，就可以直接调用。

调用智能等待findElement()实现元素等待

该函数的参数设计说明：

1）参数1：driver===浏览器对象

2）参数2：locator===指定的需要等待元素的定位信息，以元组形式传入，元组中第一个值，定位方式，第二个是对应方式定位值。其中定位方式，需要导入By类来获取8种定位方式。

3）参数3：time===设置最大等待时间，该参数设置了默认值，默认等待最大时间为10秒。默认值参数可以不传值。

示例1：

```
```

示例2：使用智能等待，等待乐鲜页面元素

```
```

5.excel操作

如果测试数据在代码中写死，当需要修改测试数据进行测试时，就需要修改代码，不方便。

如果能够将代码和数据进行分离，那么就可以集中管理测试数据，方便管理维护代码，减少代码量。

可以将测试数据写在单独的文件中，比如excel，csv，txt，yaml，json等。当在测试用例中需要用到测试数据时，可以从文件中读取数据出来。

在python中通过第三方的xlrd库实现对excel的读取。

安装：在cmd中，输入命令：pip install xlrd

```
```