# Python

## Selenium

### 理论

Selenium：

本身是Firefox下的一个插件，主要用于火狐浏览器的UI操作录制。可以转换为Java、Ruby、js语言的内容。然后被Google关注，推出Chrome浏览器所支持的浏览器驱动：ChromeDriver

核心是基于js core来实现的

Selenium grid：传统的自动化测试框架都是基于代码的线性运行来实现的。对于更为复杂、耗费时间成本更高的自动化测试，就会应用更为高级的方式来实现。例如：多用例的并发，测试框架分布式部署。Grid就是用于实现测试框架的分布式部署，基于M/S形态的架构部署，主从节点。主要通过主节点下发任务，从节点接收并执行任务，将结果返回至主节点进行记录。

Webdriver：

1. 不同的浏览器对应不同的webdriver
2. 不同的版本对应不同版本的webdriver，除了Chrome浏览器以为，其余浏览器对应的webdriver都可以在selenium官网了解。
3. 保存在python安装根路径即可
4. 浏览器安装在默认路径即可
5. Chromedriver本身是一个服务，在selenium自动化测试中，所有的代码都是不与浏览器交互的，其实都是只与webdriver服务进行交互，包含各类的指令数据和返回信息

### **八大元素定位法则**

**Close与quit区别**

Close关闭当前标签页

Quit关闭浏览器，释放进程

#### 八大元素定位

所有的ui层的自动化都是基于元素定位来实现的。所有被操作的元素都是WebElement对象

元素 = HTML标签

异常：Element not interactable 定位的元素无法进行交互

1. Id：基于元素属性中id的值来进行定位，类似于身份证上的身份证号码，不出意外都不会重复
2. Name：基于元素属性中name的值来进行定位，类似于身份证上的身份证名称，很可能会出现重名
3. Link text：主要用于超链接进行定位
4. Partial link text：link text的模糊查询版本，类似于数据库中的like%，当模糊查询匹配到多个符合条件的元素，选取第一个；当需要获取多个元素时用elements，使用for循环遍历结果集
5. Classname：基于元素样式进行定位，非常容易用到重复的
6. Tagname：基于标签名称来进行定位，在重复度最高只有在定位后需要二次筛选时才会使用
7. Cssselector：应用相对较多的一种行为，最初IE浏览器不支持Xpath，完全基于class属性进行定位
8. Xpath：目前应用最多的一种行为，基于页面结构进行定位。绝对路径：从HTML根路径下一层一层往下数，找到相对应的层级，从而找到元素；相对路径：基于匹配制度查找元素，依照xpath语法结构来定

例：//\*[@id=”kw”]

// 表示从根目录下开始查找元素

\* 任意元素

[] 表示筛选条件（查找条件）

@ 表示基于属性来筛选，例：@id=”kw”表示基于id属性值为kw的条件进行筛选

确认xpath路径是否正确：

1. 在开发者工具elements页面使用Ctrl+f查找，进行判断
2. 在console中输入$x()进行校验

基于text来定位元素：

在[]中添加text()=”文本内容”进行查找，例：//a[text()=’登录’]

基于value定位元素：

//input[@value=’百度一下’]

定位元素无法直接定位时，可以定位子级元素返回父级来获取元素

//input[contains(@id,’kw’)]

Contains:表示进一步查找，匹配项模糊查找

//input[contains(text(),’搜索’)]

### **Selenium等待的作用**

### **鼠标操作**

### **窗口切换**

### **Iframe操作**

## Python操作Excel