**1 从定位元素开始**

         在Python语言中对应的定位方法如下：

find\_element\_by\_id() → find\_element\_by\_id(“kw”)

find\_element\_by\_name() → find\_element\_by\_name (“wd”)

find\_element\_by\_class\_name() → find\_element\_by\_class\_name(“bg s\_btn”)

find\_element\_by\_tag\_name() → find\_element\_by\_tag\_name(“input”)

<a class=”mnav” name=”tj\_trvideo” href=<http://v.baidu.com> ”>视频</a>

find\_element\_by\_link\_text() 标签之间的文本信息（“新闻”“视频”）见↑

find\_element\_by\_partial\_link\_text() 类似link\_text(),但是只取文本信息的一部分。要唯一。

**find\_element\_by\_xpath()**

1.绝对路径定位 find\_element\_by\_xpath(“/html/body/div/div[2]/div/form/span/input”)

2.利用元素属性定位 最好利用Firepath帮助定位

①find\_element\_by\_xpath(“//input[@id=’kw’]”) id为属性，‘kw’为属性值

②find\_element\_by\_xpath(“//input[@name=’kd’]”) ……

③find\_element\_by\_xpath(“//\*[@class=’bg s\_btn’]”) 不想指定标签名，可以用星号（\*）代替

3.层级与属性结合（本身没可利用的属性值，可以查找它的上（上）一级属性）

find\_element\_by\_xpath(“//span[@class=’bg s\_iptwr’]/input”)

4.使用逻辑运算符

<input id=”kw” class=”su” name=”ie”> ↓查找

find\_element\_by\_xpath(“//input[@id=’kw’ and @class=’su’]/span/input”)

find\_element\_by\_css\_selector()  (都是选择所有元素)

//input[@id=’auto-id-1490869281227’ and class=’j-inputtext dlemail’ and ]

**selenium使用Xpath定位之完整篇**

其中有一片文章提到了xpath元素定位，但是该文章中有些并不能适应一些特殊与个性化的场景。在文本中提供xpath元素的定位终极篇，你一定能在这里找到你需要的解决办法。

**第一种方法：通过绝对路径做定位（相信大家不会使用这种方式）**

By.xpath("html/body/div/form/input")

By.xpath("//input")

**第三种方法：通过元素索引定位**

By.xpath("//input[4]")

**第四种方法：使用xpath属性定位（结合第2、第3中方法可以使用）**

By.xpath("//input[@id='kw1']")

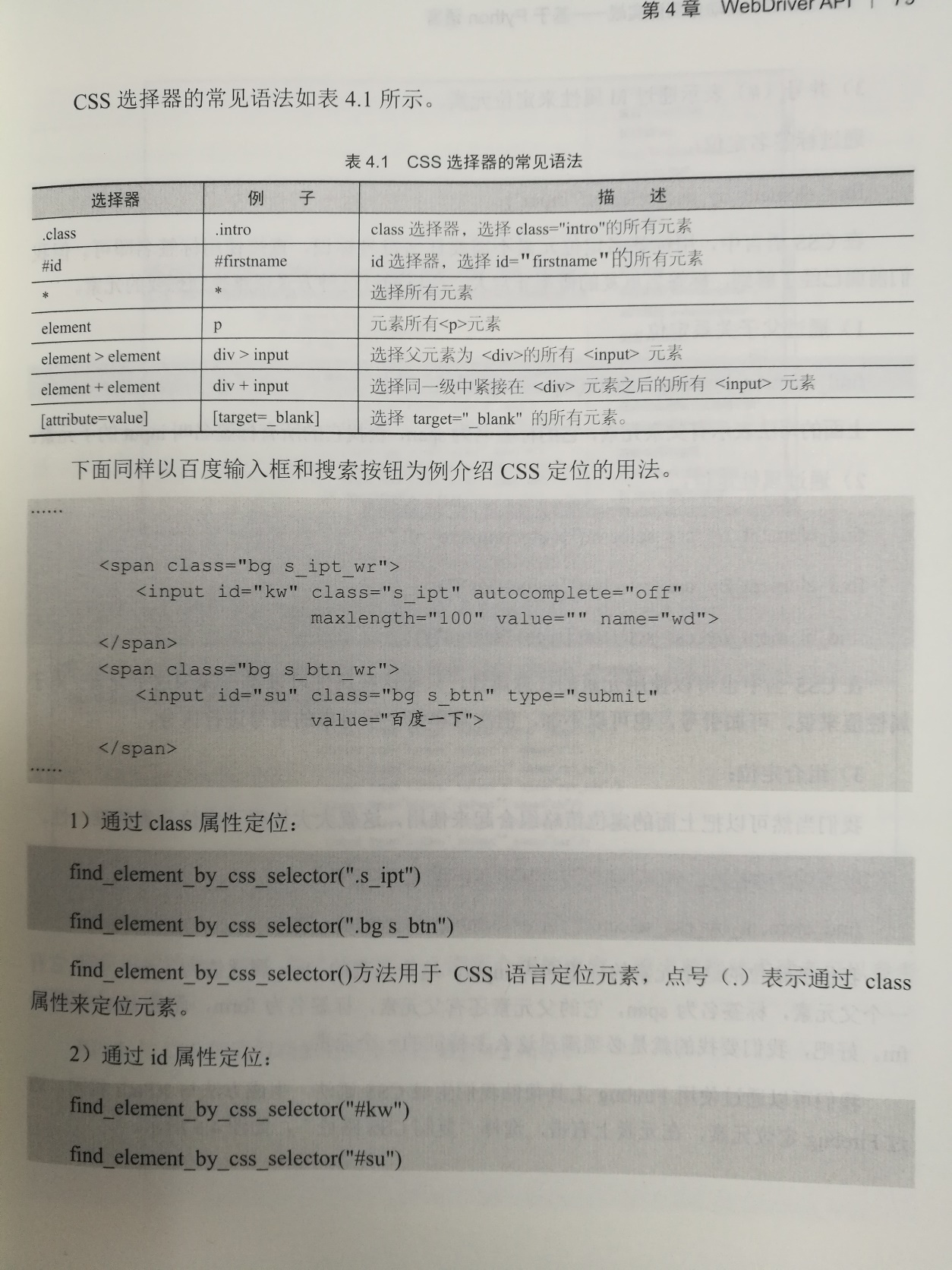
By.xpath("//input[@type='name' and @name='kw1']")

**第五种方法：使用部分属性值匹配（最强大的方法）**

By.xpath("//input[start-with(@id,'nice')

By.xpath("//input[ends-with(@id,'很漂亮')

By.xpath("//input[contains(@id,'那么美')]")

****

find\_element\_by\_css\_selector()  (都是选择所有元素)

用by定位元素，find\_element()方法只用于定位元素。它需要两个参数，第一个参数是定位方式，这个由By提供；第二个参数是定位的值。在使用By时需要将By类导入。

from selenium.webdriver.common.by import By

**2 控制浏览器**

 WebDriver提供了set\_window\_size()方法来设置浏览器的大小。

例：driver.set\_window\_size(480,800)长宽 全屏：driver.maximize\_window()

WebDriver提供back()和forward()方法来模拟后退和前进。

driver.back()   driver.forward()

**3 简单元素操作**

clear()  清除文本，如果是一个文本输入框

refresh() 刷新当前页面

send\_keys(\*value)在元素上模拟按键输入

click()      单击元素

click()方法不仅仅用于点击按钮，还可以单击任何可以点击文字/图片链接、复选框、单选框、甚至是下拉框等。

submit()方法用于提交表单，这里特别用于没提交按钮的情况，例如输入关键字之后的“回车”操作，那么就可以通过submit()来提交搜索框的内容。

检查某个框是否被选中

element.is\_selected()判断是否被选中

element.is\_displayed()判断是否显示/该元素是否用户可见

element.is\_enabled()判断是否被使用

size           没括号 返回元素的尺寸

       text                 没括号     获取元素的文本

       get\_attribute(‘name’)     获取属性值

size=driver.find\_element\_by\_id(“kw”).size

text=driver.find\_element\_by\_id(“cp”).text

attribute=driver.find\_element\_by\_id(“kw”).get\_attribute(“type”)

result=driver.find\_element\_by\_id(“kw”).is\_displayed()

**导航 (Navigationand History)**

**//打开一个新的页面：**

**driver.navigate().to("http://www.example.com");**

**//通过历史导航返回原页面：**

**driver.navigate().forward();**

**driver.navigate().back();**

**4 鼠标事件**

在webdriver中这些关于鼠标操作的方法由ActionChains类提供。

ActionChains类提供的鼠标操作的常用方法：

**click**(*on\_element=None*) ——单击鼠标左键

**click\_and\_hold**(*on\_element=None*) ——点击鼠标左键，不松开

**context\_click**(*on\_element=None*) ——点击鼠标右键

**double\_click**(*on\_element=None*) ——双击鼠标左键

**drag\_and\_drop**(*source, target*) ——拖拽到某个元素然后松开

**drag\_and\_drop\_by\_offset**(*source, xoffset, yoffset*) ——拖拽到某个坐标然后松开

**key\_down**(*value, element=None*) ——按下某个键盘上的键

**key\_up**(*value, element=None*) ——松开某个键

**move\_by\_offset**(*xoffset, yoffset*) ——鼠标从当前位置移动到某个坐标

**move\_to\_element**(*to\_element*) ——鼠标移动到某个元素

**move\_to\_element\_with\_offset**(*to\_element, xoffset, yoffset*) ——移动到距某个元素（左上角坐标）多少距离的位置

**perform**() ——执行链中的所有动作

**release**(*on\_element=None*) ——在某个元素位置松开鼠标左键

**send\_keys**(*\*keys\_to\_send*) ——发送某个键到当前焦点的元素

**send\_keys\_to\_element**(*element, \*keys\_to\_send*) ——发送某个键到指定元素

from selenium import webdriver

1. from selenium.webdriver.common.action\_chains import ActionChains
2. driver=webdrive.Chrome()
3. driver.get(“http://yun.baidu.com/”)
5. driver.find\_element\_by\_id(“TANGRAM\_\_PSP\_4\_\_userName”).clear()
6. driver.find\_element\_by\_id(“TANGRAM\_\_PSP\_4\_\_userName”).send\_keys(“username”)
7. driver.find\_element\_by\_id(“TANGRAM\_\_PSP\_4\_\_password”).clear()
8. driver.find\_element\_by\_id(“TANGRAM\_\_PSP\_4\_\_password”).send\_keys(“password”)
9. driver.find\_element\_by\_id(“TANGRAM\_\_PSP\_4\_\_submit”).click()
11. disk\_NET=driver.find\_element\_by\_class\_name(“pulldown-nav”)
12. ActionChains(driver).move\_to\_element(disk\_NET)
13. DC=driver.find\_element\_by\_class\_name(“li disk”)
14. ActionChains(driver).double\_click(DC)

**5.Select下拉框定位**

from selenium.webdriver.support.select import Select

Select(driver.find\_element\_by\_id("id\_language")).select\_by\_value('en')

通过选项的顺序，第一个为 0

.select\_by\_index(1)

.select\_by\_visible\_text(‘选项内容’)

反选

deselect\_by\_index(index)

deselect\_by\_value(value)

deselect\_by\_visible\_text(text)

deselect\_all()

**5 键盘事件**

Keys类提供键盘上几乎所有的按键的方法，send\_keys()方法可以模拟键盘输入，除此之外还可以模拟键盘上的一些组合键，例如Ctrl+A,Ctrl+C等

1. #coding=utf-8
2. from selenium import webdriver
3. from selenium.webdriver.commom.keys import Keys
4. driver=webdriver.Chrome()
5. driver.get(“http://www.baidu.com”)
6. driver.find\_element\_by\_id(“kw”).send\_keys(“seleniumm”)
7. #删除多输入的一个m
8. driver.find\_element\_by\_id(“kw”).send\_keys(Keys.BACK\_SPACE)
9. 11.#输入空格键+“教程”
10. driver.find\_element\_by\_id(“kw”).send\_keys(Keys.SPACE)
11. driver.find\_element\_by\_id(“kw”).send\_keys(“教程”)
12. #Ctrl+a全选输入框内容
13. driver.find\_element\_by\_id(“kw”).send\_keys(Keys.CONTROL,’a’)
14. #Ctrl+x剪切输入框内容
15. driver.find\_element\_by\_id(“kw”).send\_keys(Keys.CONTROL,’x’)
16. #Ctrl+v粘贴输入框内容
17. driver.find\_element\_by\_id(“kw”).send\_keys(Keys.CONTROL,’v’)
18. #Enter回车键
19. driver.find\_element\_by\_id(“su”).send\_keys(Keys.ENTER)
20. driver.quit()
21. 导入键盘类包：from selenium.webdriver.commom.keys import Keys
22. 经常用到键盘操作：
23. send\_keys(Keys.BACK\_SPACE)删除键（BackSpace）
24. send\_keys(Keys.SPACE)空格键(Space)
25. send\_keys(Keys.TAB)制表键(Tab)
26. send\_keys(Keys.ESCAPE)回退键（Esc）
27. send\_keys(Keys.ENTER)回车键（Enter）
28. send\_keys(Keys.CONTROL,'a')全选（Ctrl+A）
29. send\_keys(Keys.CONTROL,'c')复制（Ctrl+C）
30. send\_keys(Keys.CONTROL,'x')剪切（Ctrl+X）
31. send\_keys(Keys.CONTROL,'v')粘贴（Ctrl+V）
32. send\_keys(Keys.F1)键盘F1
33. send\_keys(Keys.ARROW\_DOWN,Keys.ARROW\_UP,Keys.ARROW\_LEFT,Keys.ARROW\_RIGHT,Keys.ENTER)#上下左右

**#页面下滑**

def type\_rolldown(self):

r="window.scrollBy(0, 700)"

self.script(r)

div内控制下滑



**6.获得验证信息**

1．获取当前页面title

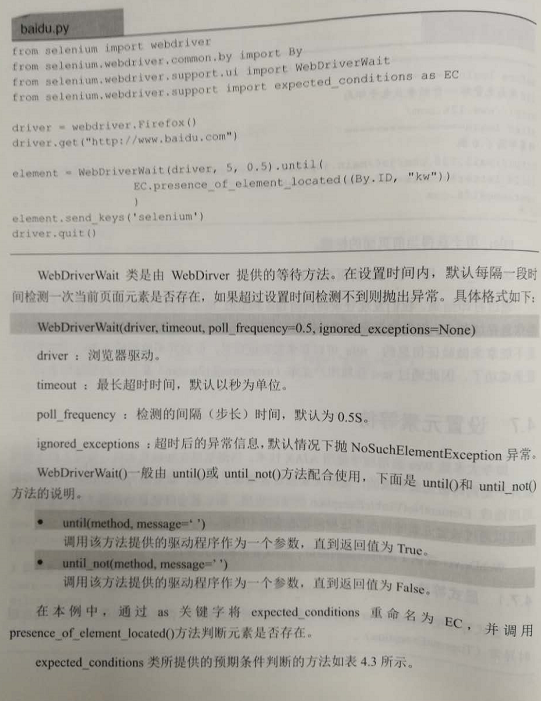
driver.title

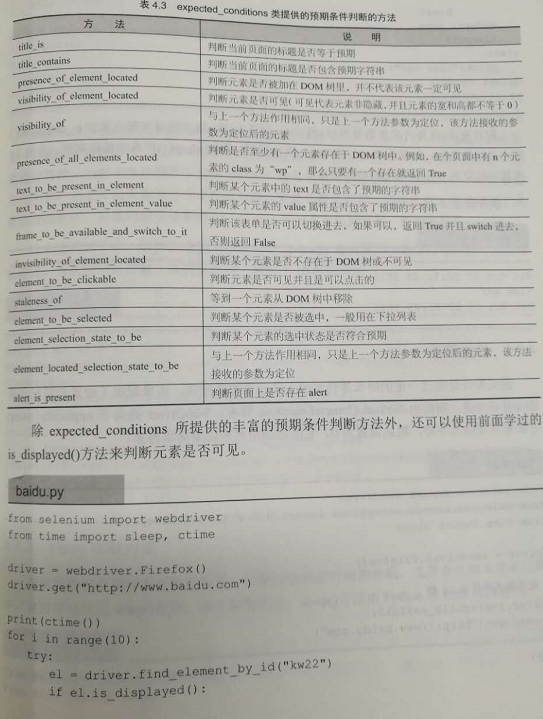
2.打印当前页面url

driver.current\_url

**7.设置元素等待———————————详见“自动化EC方法与实例”**

1.**显示等待** 显示等待是WebDriver等待某个条件成立时继续执行，否则在达到最大时长时抛出超时异常。（TimeoutException）





等待的库——快捷方式

from selenium import webdriver

from selenium.webdriver.common.action\_chains import ActionChains

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

2.**隐式等待**

以轮询的方式不断地判断元素是否被定位到。找到则继续执行下面的代码。

from selenium.common.exceptions import NoSuchElementException

driver.implicitly\_wait(10) //参数为默认时间（秒）

**3.固定时间的休眠**

import time

time.sleep(10)

**8.多表单切换**

**frame/iframe需要切换，frameset不切换**

driver.switch\_to.frame(“属性值”) ——————默认可以直接取表单的id或name属性。如果iframe没有可用的id和name属性，则可以通过下面的方法进行定位。

#先通过xpath定位到iframe

xf=driver.find\_element\_by\_xpath(“//\*[@class=’if’]”)

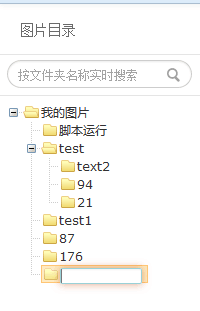
#再将定位对象传给switch\_to.frame()方法

driver.switch\_to.frame(xf)

……

driver.switch\_to.parent\_frame()切回父frame

driver.switch\_to.default\_content()切回主文档

driver.switch\_to.active\_element.send\_keys('filename')

**9.多窗口切换**

·driver.current\_window\_handle:获得当前窗口句柄（后面没有括号）

·driver.window\_handles:返回所有窗口的句柄到当前会话————调用前sleep一会儿

·driver.switch\_to.window( 窗口变量):用于切换到相应的窗口

from selenium import webdriver

browser=webdriver.Firefox()

browser.maximize\_window() # 窗口最大化

browser.get('https://www.baidu.com') # 在当前浏览器中访问百度

# 新开一个窗口，通过执行js来新开一个窗口

js='window.open("https://www.sogou.com");' http://tst195.yylending.com:50004/login

browser.execute\_script(js)

print browser.current\_window\_handle # 输出当前窗口句柄（百度）

handles = browser.window\_handles # 获取当前窗口句柄集合（列表类型）

print handles # 输出句柄集合

for handle in handles:# 切换窗口（切换到搜狗）

if handle!=browser.current\_window\_handle:

print 'switch to ',handle

browser.switch\_to\_window(handle)

print browser.current\_window\_handle # 输出当前窗口句柄（搜狗）

break

sleep(0.5)

browser.close() #关闭当前窗口（搜狗）

browser.switch\_to\_window(handles[0]) #切换回百度窗口

import time

time.sleep(10)

browser.quit()

**10.警告框处理**

·driver.switch\_to.alert.text :返回alert/confirm/prompt中的文字信息。

·driver.switch\_to.alert.accept():接受现有警告框。

·driver.switch\_to.alert.dismiss():解散现有警告框。

·driver.switch\_to.alert.send\_keys(keysToSend):发送文本至警告框。

多用sleep(),让页面加载完。

②selenium.webdriver.common.alert.**Alert**(driver)

Alert(driver).accept() # 等同于点击“确认”或“OK”

Alert(driver).dismiss() # 等同于点击“取消”或“Cancel”

Alert(driver).authenticate(username,password) # 验证，针对需要身份验证的alert，目前还没有找到特别合适的示例页面

Alert(driver).send\_keys(keysToSend) # 发送文本，对有提交需求的prompt框（上图3）

Alert(driver).text # 获取alert文本内容，对有信息显示的alert框

**11.上传文件**

对于通过input标签实现的上传功能，可以将其看作是一个输入框，即通过

send\_keys（）制定本地文件路径的方式实现文件上传。

1. send\_keys实现上传

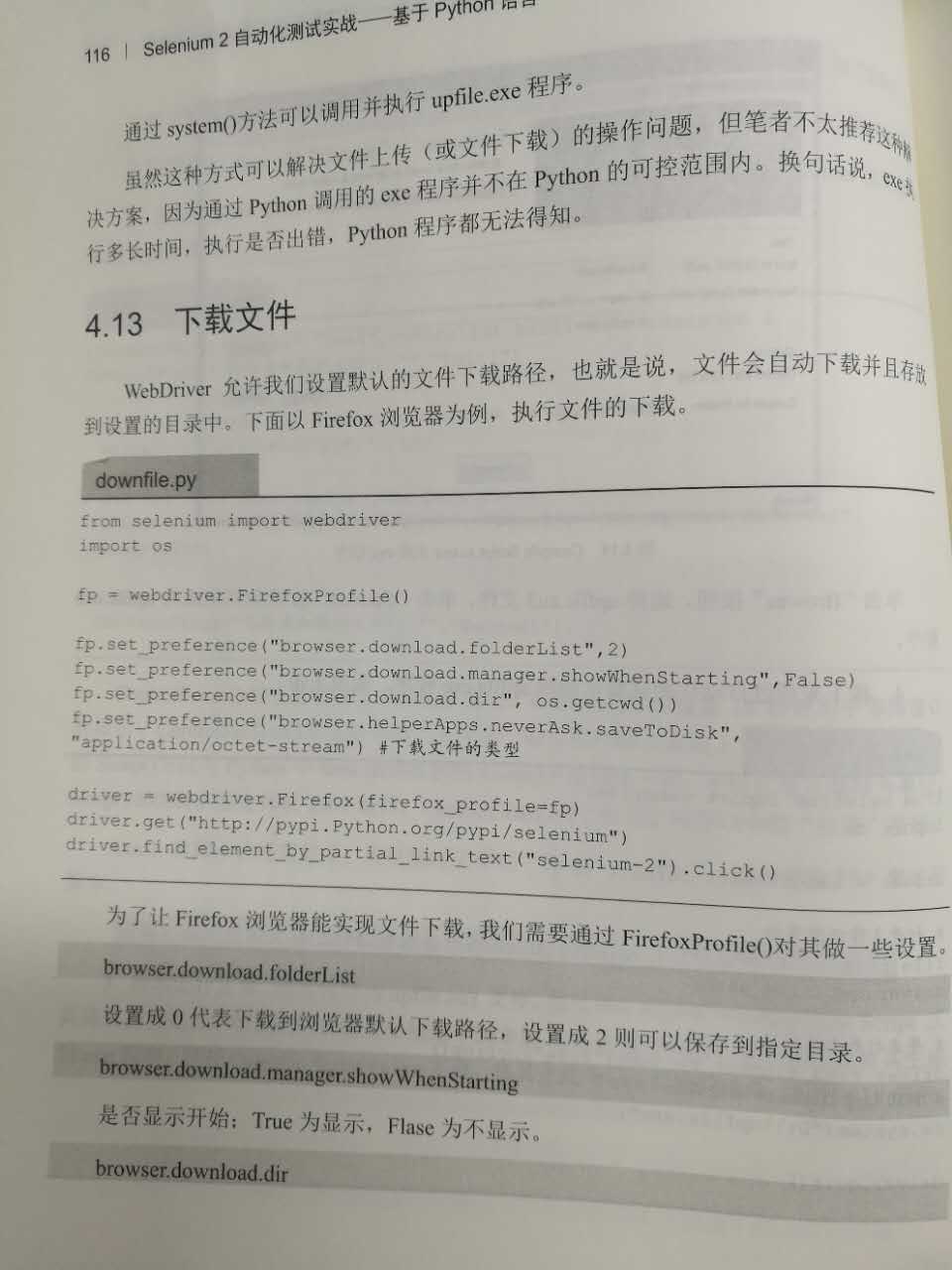
#定位上传按钮，添加本地文件

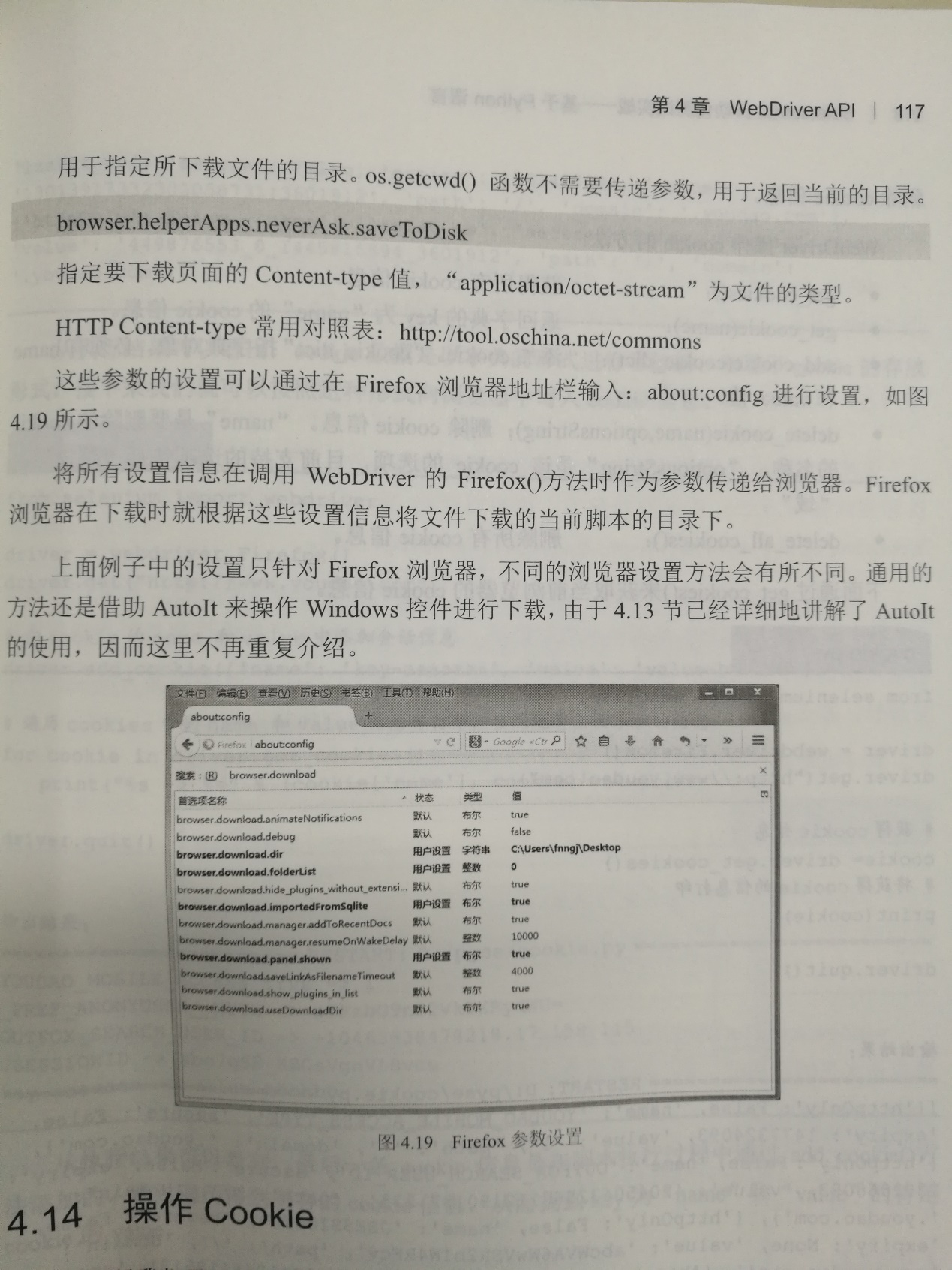
driver.find\_element\_by\_name(“file”).send\_keys(“D:\\upload\_file.txt”)

1. AutoIt实现上传（要下载AutoIt）

编写脚本→转换成exe程序→在python运行（os.system（“D:\\upfile.exe”））

**12.下载文件**





**13.操作Cookie**

·driver.get\_cookies() : 获得所有cookie信息

·driver.get\_cookie(name) : 返回字典的key为“name”的cookie信息

·driver.add\_cookie(cookie\_dict) : 添加cookie。“cookie\_dict”指字典对象，必须有name和value值。

·driver.delete\_cookie(name,optionsString) : 删除cookie的信息。“name”是要删除的cookie的名称，“optionsString”是该cookie的选项，目前支持的选项包括“路径”、“域”。

·driver.delete\_all\_cookies() : 删除所有cookie信息

**14.调用JavaScript\_\_\_\_\_\_\_\_\_\_\_\_\_driver.execute\_script( js )**

#通过javascript设置浏览器窗口的滚动条位置

js=”window.scrollTo(100,450);”

driver.execute\_script(js)

**15.窗口截图**

driver.get\_screenshot\_as\_file(r"E:\PythonWorkspace\test1.jpg")

**16.Selenium IDE常见命令**

1.open 打开url

2.click 点击链接、按钮、复选和单选框

clickAndWait

3.type 模拟键盘的输入，也适合给复选框和单选框赋值，

typeAndWait

4.select根据选择器来选择一个下拉菜单选项

5.goBack模拟单击浏览器的后退按钮

6.selectWindow选择一个弹出窗口，所有的命令将会转移到被选择窗口中执行

7.pause根据制定时间暂停Selenium脚本，常在调试脚本或等待服务器响应时使用

8.fireEvent模拟页面元素时间被激活的处理动作

9.close模拟单击浏览器关闭按钮

**17.断言与验证**

4类命令：assert（断言）、verify（验证）、waitFor（等待）、store（定义变量）

断言失败后测试用例会停止运行；验证失败后不会终止测试。

这四类命令分为5种验证手段：

·Title 获取页面的标题

·Value 获取元素的值

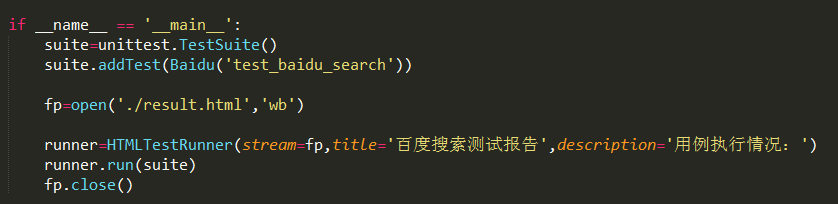
·Text 获取元素的文本信息

·Table 获取元素的标签

·ElementPresent 获取当前元素

**18.等待与变量**

pause设置固定时间的休眠。waitFor用于在一定时间内等待某一元素显示。（waitFor默认60s）

**19.生成HTML测试报告**

优化报告名字(加上当前时间)

now=time.strftime('%Y-%m-%d %H\_%M\_%S')

filename='./'+now+'result.html'

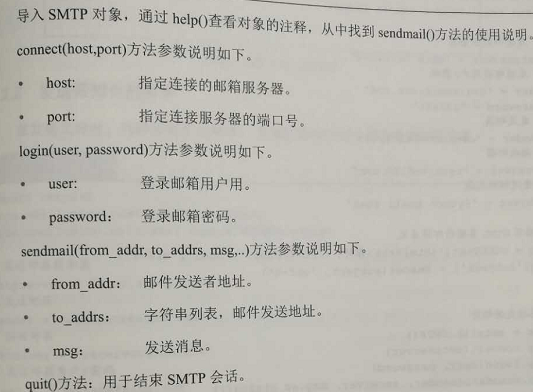
fp=open(filename,'wb')

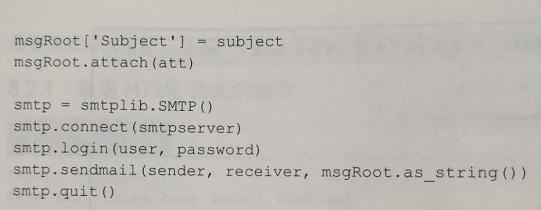
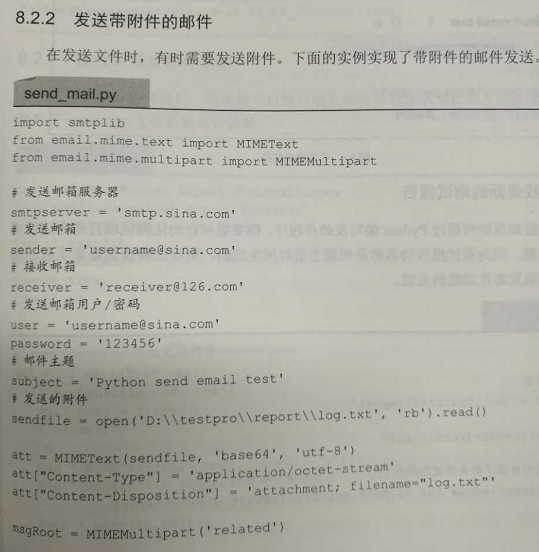
runner=HTMLTestRunner(stream=fp,title='百度搜索测试报告',description='用例执行情况：')

runner.run(suite)

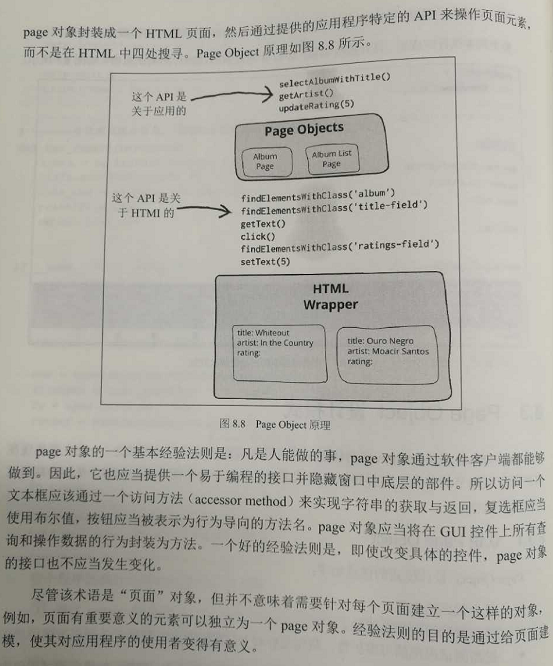
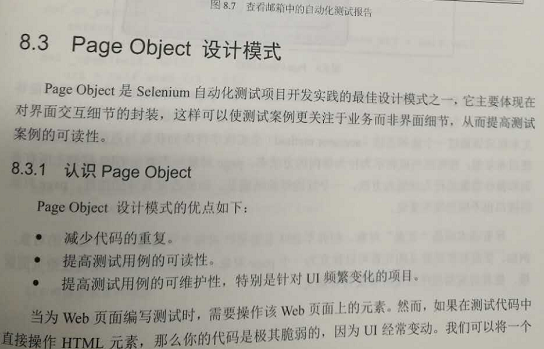
fp.close()

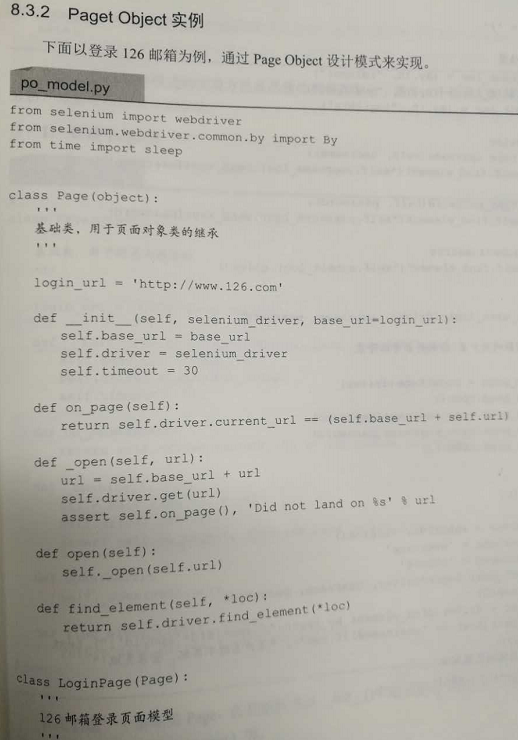
**20.自动发邮件功能**

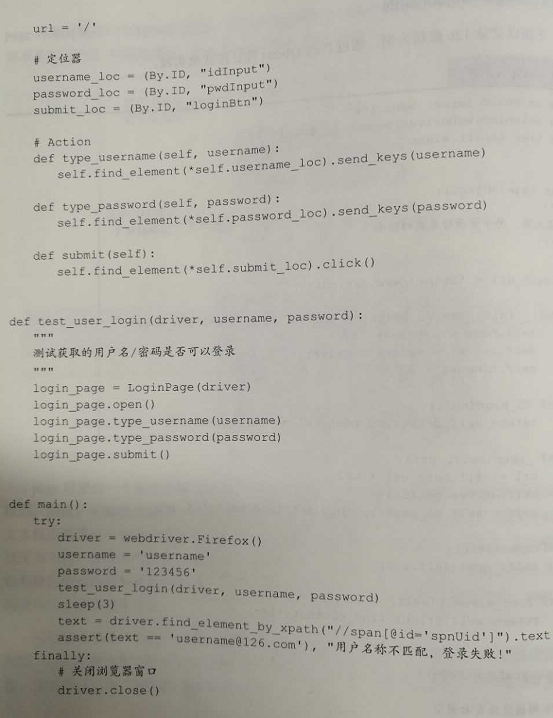
we

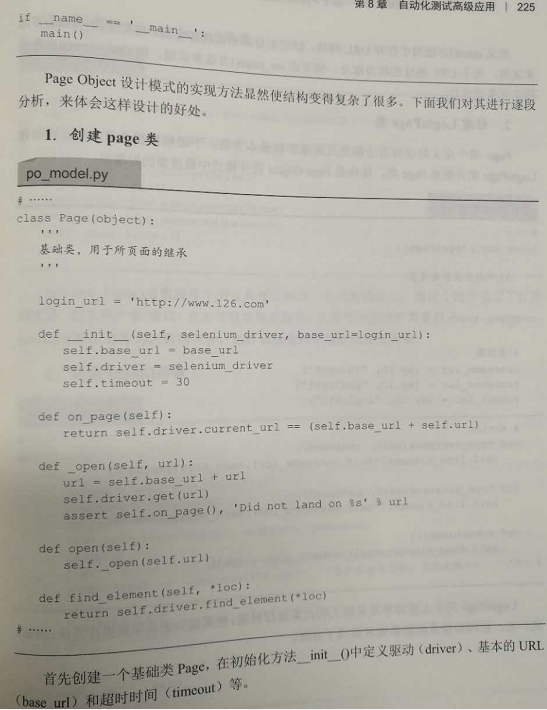


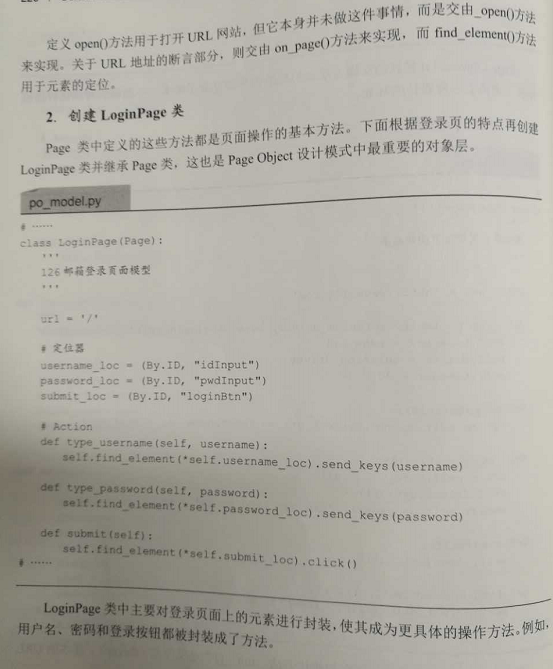
**21.Page Object设计模式**

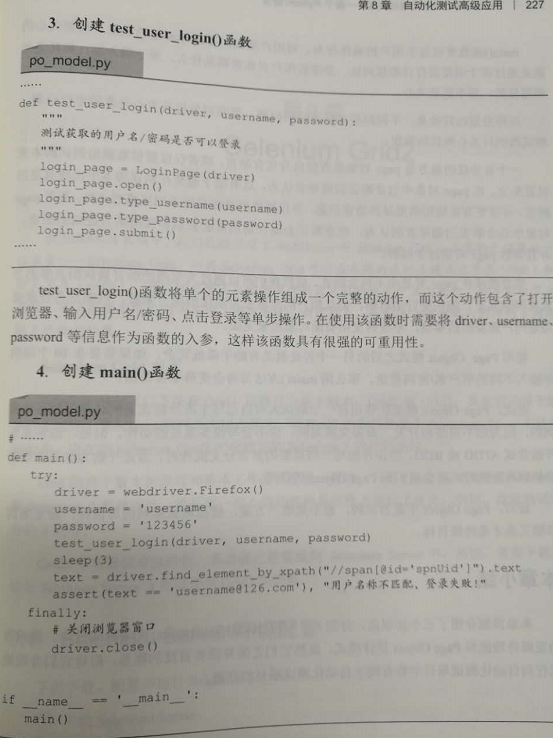


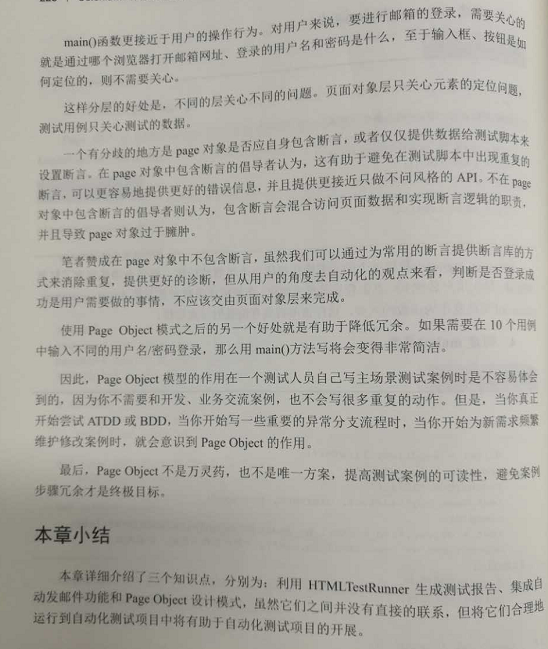








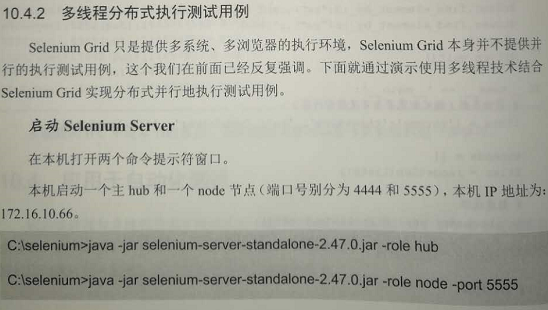


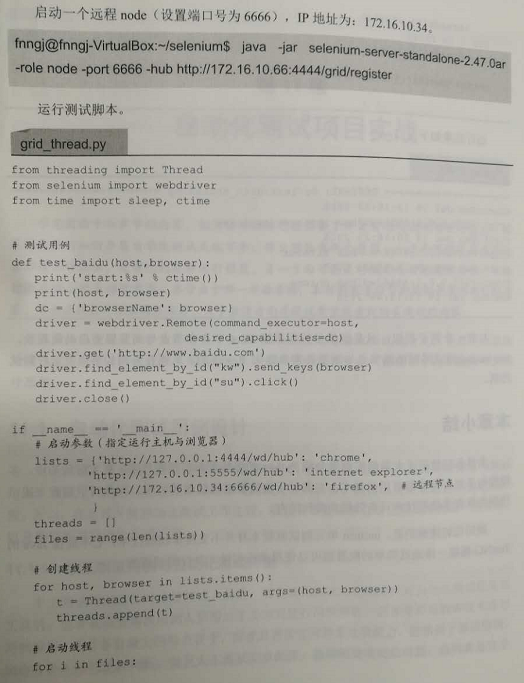


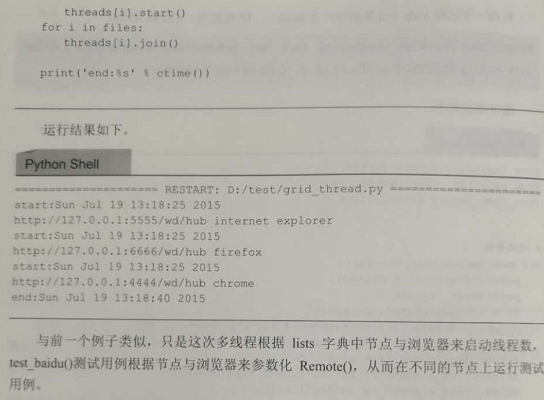
**多线程**



**Selenium Grid多线程分布式执行测试用例**

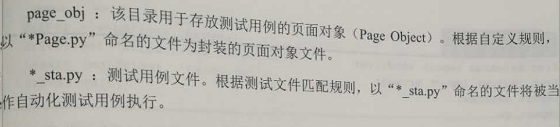
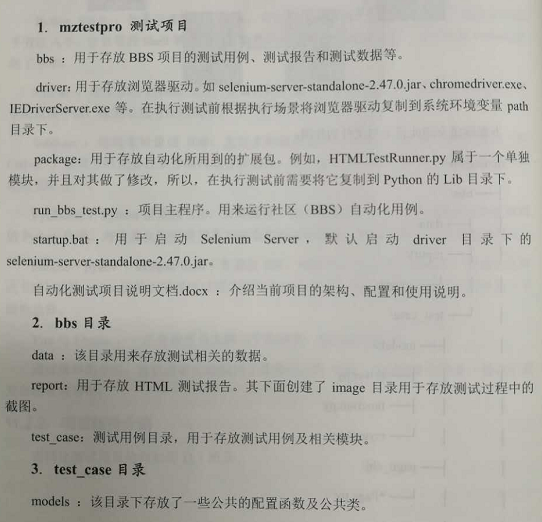






通过浏览器访问Grid的控制台：http://127.0.0.1:4444/grid/console

**自动化目录框架模型：**



**JS方法**

**removeAttribute()**



分享几种兼容以上三种浏览器的js写法：

$(window).scrollTop(300);

$(document).scrollTop(300)

$("html,body").scrollTop(300);