Python selenium —— 一定要会用selenium的等待，三种等待方式解读

Sep 14, 2016

发现太多人不会用等待了，博主今天实在是忍不住要给大家讲讲等待的必要性。

很多人在群里问，这个下拉框定位不到、那个弹出框定位不到…各种定位不到，其实大多数情况下就是两种问题：1 有frame，2 没有加等待。殊不知，你的代码运行速度是什么量级的，而浏览器加载渲染速度又是什么量级的，就好比闪电侠和凹凸曼约好去打怪兽，然后闪电侠打完回来之后问凹凸曼你为啥还在穿鞋没出门？凹凸曼分分中内心一万只羊驼飞过，欺负哥速度慢，哥不跟你玩了，抛个异常撂挑子了。

那么怎么才能照顾到凹凸曼缓慢的加载速度呢？只有一个办法，那就是等喽。说到等，又有三种等法，且听博主一一道来：

1. 强制等待

第一种也是最简单粗暴的一种办法就是强制等待sleep(xx)，强制让闪电侠等xx时间，不管凹凸曼能不能跟上速度，还是已经提前到了，都必须等xx时间。

看代码：

```
# -*- coding: utf-8 -*-
from selenium import webdriver
from time import sleep

driver = webdriver.Firefox()
driver.get('https://huilansame.github.io')

sleep(3)  # 强制等待3秒再执行下一步

print driver.current_url
driver.quit()
```

这种叫强制等待，不管你浏览器是否加载完了，程序都得等待3秒，3秒一到，继续执行下面的代码，作为调试很有用，有时候也可以在代码里这样等待，不过不建议总用这种等待方式，太死板，严重影响程序执行速度。

2. 隐性等待

第二种办法叫隐性等待，implicitly_wait(xx)，隐性等待的意义是：闪电侠和凹凸曼约定好，不论闪电侠去哪儿，都要等凹凸曼xx秒，如果凹凸曼在这段时间内来了，则俩人立即出发去打怪兽，如果凹凸曼在规定时间内没到，则闪电侠自己去，那自然就等着凹凸曼给你抛异常吧。

看代码：

```
# -*- coding: utf-8 -*-
from selenium import webdriver

driver = webdriver.Firefox()
driver.implicitly_wait(30)  # 隐性等待，最长等30秒
driver.get('https://huilansame.github.io')
```

```
driver.get('https://huilansame.github.io')

print driver.current_url
driver.quit()
```

隐形等待是设置了一个最长等待时间，如果在规定时间内网页加载完成，则执行下一步，否则一直等到时间截止，然后执行下一步。注意这里有一个弊端，那就是程序会一直等待整个页面加载完成，也就是一般情况下你看到浏览器标签栏那个小圈不再转，才会执行下一步，但有时候页面想要的元素早就在加载完成了，但是因为个别js之类的东西特别慢，我仍得等到页面全部完成才能执行下一步，我想等我要的元素出来之后就下一步怎么办？有办法，这就要看selenium提供的另一种等待方式——显性等待wait了。

需要特别说明的是：隐性等待对整个driver的周期都起作用，所以只要设置一次即可，我曾看到有人把隐性等待当成了sleep在用，走哪儿都来一下...

3. 显性等待

第三种办法就是显性等待，WebDriverWait，配合该类的until()和until_not()方法，就能够根据判断条件而进行灵活地等待了。它主要的意思就是：程序每隔xx秒看一眼，如果条件成立了，则执行下一步，否则继续等待，直到超过设置的最长时间，然后抛出TimeoutException。

先看个代码示例：

```
# -*- coding: utf-8 -*-
from selenium import webdriver
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By

driver = webdriver.Firefox()
driver.implicitly_wait(10)  # 隐性等待和显性等待可以同时用，但要注意：等待的最长时间取两者之中的大
driver.get('https://huilansame.github.io')
locator = (By.LINK_TEXT, 'CSDN')

try:
    WebDriverWait(driver, 20, 0.5).until(EC.presence_of_element_located(locator))
    print driver.find_element_by_link_text('CSDN').get_attribute('href')
finally:
    driver.close()
```

上例中，我们设置了隐性等待和显性等待，在其他操作中，隐性等待起决定性作用，在WebDriverWait..中显性等待起主要作用，但要注意的是：**最长的等待时间取决于两者之间的大者，此例中为20，如果隐性等待时间 > 显性等待时间，则该句代码的最长等待时间等于隐性等待时间。**

我们主要用到了WebDriverWait类与expected_conditions模块，下面博主带大家细看一下这两个模块：

WebDriverWait

wait模块的WebDriverWait类是显性等待类，先看下它有哪些参数与方法：

```
selenium.webdriver.support.wait.WebDriverWait（类）

__init__
    driver: 传入WebDriver实例，即我们上例中的driver
    timeout: 超时时间，等待的最长时间（同时要考虑隐性等待时间）
    poll_frequency: 调用until或until_not中的方法的间隔时间，默认是0.5秒
    ignored_exceptions: 忽略的异常，如果在调用until或until_not的过程中抛出这个元组中的异常，
            则不中断代码，继续等待，如果抛出的是这个元组外的异常，则中断代码，抛出异常。默认只有No

until
    method: 在等待期间，每隔一段时间调用这个传入的方法，直到返回值不是False
```

```
    message: 如果超时，抛出TimeoutException，将message传入异常

until_not 与until相反，until是当某元素出现或什么条件成立则继续执行，
        until_not是当某元素消失或什么条件不成立则继续执行，参数也相同，不再赘述。
    method
    message
```

看了以上内容基本上很清楚了，调用方法如下：

```
WebDriverWait(driver, 超时时长, 调用频率, 忽略异常).until(可执行方法, 超时时返回的信息)
```

这里需要特别注意的是until或until_not中的可执行方法method参数，很多人传入了WebElement对象，如下：

```
WebDriverWait(driver, 10).until(driver.find_element_by_id('kw'))  # 错误
```

这是错误的用法，这里的参数一定要是可以调用的，即这个对象一定有 `__call__()` 方法，否则会抛出异常：

```
TypeError: 'xxx' object is not callable
```

在这里，你可以用selenium提供的 `expected_conditions` 模块中的各种条件，也可以用WebElement的 `is_displayed()` 、 `is_enabled()` 、 `is_selected()` 方法，或者用自己封装的方法都可以，那么接下来我们看一下selenium提供的条件有哪些：

expected_conditions

expected_conditions是selenium的一个模块，其中包含一系列可用于判断的条件：

```
selenium.webdriver.support.expected_conditions（模块）

这两个条件类验证title，验证传入的参数title是否等于或包含于driver.title
title_is
title_contains

这两个人条件验证元素是否出现，传入的参数都是元组类型的locator，如(By.ID, 'kw')
顾名思义，一个只要一个符合条件的元素加载出来就通过；另一个必须所有符合条件的元素都加载出来才行
presence_of_element_located
presence_of_all_elements_located

这三个条件验证元素是否可见，前两个传入参数是元组类型的locator，第三个传入WebElement
第一个和第三个其实质是一样的
visibility_of_element_located
invisibility_of_element_located
visibility_of

这两个人条件判断某段文本是否出现在某元素中，一个判断元素的text，一个判断元素的value
text_to_be_present_in_element
text_to_be_present_in_element_value

这个条件判断frame是否可切入，可传入locator元组或者直接传入定位方式：id、name、index或WebElement
frame_to_be_available_and_switch_to_it

这个条件判断是否有alert出现
alert_is_present

这个条件判断元素是否可点击，传入locator
element_to_be_clickable
```

这四个条件判断元素是否被选中，第一个条件传入WebElement对象，第二个传入locator元组
第三个传入WebElement对象以及状态，相等返回True，否则返回False
第四个传入locator以及状态，相等返回True，否则返回False

```
element_to_be_selected
element_located_to_be_selected
element_selection_state_to_be
element_located_selection_state_to_be
```

最后一个条件判断一个元素是否仍在DOM中，传入WebElement对象，可以判断页面是否刷新了

```
staleness_of
```

上面是所有17个condition，与until、until_not组合能够实现很多判断，如果能自己灵活封装，将会大大提高脚本的稳定性。

今天就分享这些内容，有什么问题可以留言给我交流，希望能帮助到有需要的同学。

> 更多关于python selenium的文章，请关注我的CSDN专栏：**Python Selenium自动化测试详解**

灰蓝

自动化测试工程师/兼职给人写点网页自动化

GitHub CSDN LinkedIn

最新文章

- Python selenium —— 将你的自动化脚本打包成一个exe
- selenium 学习网站
- Python必会的单元测试框架 —— unittest
- [译]Selenium —— 怎样使用FireBug和FirePath
- [译]Selenium Webdriver - 下载、安装稳定版本