**Selenium中的“显示等待”和“隐式等待”**

现在越来越多的 web app 使用 AJAX 技术。当一个页面加载到浏览器后，这个页面的很多元素加载的时间可能不一致。这就使得元素定位比较尴尬了，如果某个要定位的元素还没有加载出来，在定位的时候，就会抛出异常： ElementNotVisibleException。这个时候，合理使用 wait ，我们就能很好的缓解这个尴尬。Selenium Webdriver 提供了两种类型的**等待**方法：隐式等待(implicit) 和 显示等待(explicit)。 显示等待情况下，只有特定条件触发后，WebDriver 才会继续执行后续操作。隐式等待情况下，WebDriver 等待一定时间，该时间段内，如果特定元素没加载成功，则抛出异常。

**1. Explicit Waits（显示等待）**

显示等待的代码定义了等待条件，只有该条件触发，才执行后续代码。最垃圾的显示等待就是使用 time.sleep()，这种情况是指定了固定的等待时长。 存在一些特别方便的方法，可以使你的代码只等待需要的时常，而不是固定的时常。WebDriverWait 和 ExpectedCondition 组合使用，就是一种有效的解决手段。

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

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

driver = webdriver.Firefox()driver.get("http://somedomain/url\_that\_delays\_loading")

try:

element = WebDriverWait(driver, 10).until(

EC.presence\_of\_element\_located((By.ID, "myDynamicElement")))

finally:

driver.quit()

上面的代码最多等待 10 秒，超时后就抛出 TimeoutException，假设在第3秒就找到了这个元素，那么也就不会多等剩下的7秒使时间，而是继续执行后续代码。WebDriverWait 默认每 500 毫秒调用一次 ExpectedCondition 中的方法，直到它返回成功信号，如果成功，则返回相应的内容，否则返回布尔类型的 False。

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

wait\_result = WebDriverWait(driver=self.driver, timeout=300, poll\_frequency=0.5, ignored\_exceptions=None).until(

EC.text\_to\_be\_present\_in\_element((By.XPATH, '//\*[@id="VolumeTable"]/tbody/tr[1]/td[4]/label'), u'可用'))

driver：浏览器驱动  
timeout：最长超时等待时间  
poll\_frequency：检测的时间间隔，默认为500ms  
ignore\_exception：超时后抛出的异常信息，默认情况下抛 NoSuchElementException 异常

配合使用的 until() 或者 until\_not() 方法说明：

* until(method, message='')  
  调用该方法体提供的回调函数作为一个参数，直到返回值为True
* until\_not(method, message='')  
  调用该方法体提供的回调函数作为一个参数，直到返回值为False

模块包含一套预定义的条件集合。大大方便了 WebDriverWait 的使用。

**Expected Conditions 类提供的预期条件判断方法**

在进行浏览器自动化的时候，有一些条件是经常出现的，下面列出的是每个条件的实现。Selenium Python binding provides some convienence 提供了很多实用的方法，因此，你无须再去编写 expected\_condition class 或是创建你自己的工具包来实现这个工作。

* title\_is：判断当前页面的title是否等于预期
* title\_contains：判断当前页面的title是否包含预期字符串
* presence\_of\_element\_located：判断某个元素是否被加到了dom树里，并不代表该元素一定可见
* visibility\_of\_element\_located：判断某个元素是否可见. **可见代表元素非隐藏，并且元素的宽和高都不等于0**
* visibility\_of：跟上面的方法做一样的事情，只是上面的方法要传入locator，这个方法直接传定位到的element就好了
* presence\_of\_all\_elements\_located：判断是否至少有1个元素存在于dom树中。举个例子，如果页面上有n个元素的class都是'column-md-3'，那么只要有1个元素存在，这个方法就返回True
* text\_to\_be\_present\_in\_element：判断某个元素中的text是否 **包含** 了预期的字符串
* text\_to\_be\_present\_in\_element\_value：判断某个元素中的value属性是否**包含**了预期的字符串
* frame\_to\_be\_available\_and\_switch\_to\_it：判断该frame是否可以switch进去，如果可以的话，返回True并且switch进去，否则返回False
* invisibility\_of\_element\_located：判断某个元素中是否不存在于dom树或不可见
* element\_to\_be\_clickable - it is Displayed and Enabled：判断某个元素中是否可见并且是enable的，这样的话才叫clickable
* staleness\_of：等某个元素从dom树中移除，注意，这个方法也是返回True或False
* element\_to\_be\_selected：判断某个元素是否被选中了,一般用在下拉列表
* element\_located\_to\_be\_selected
* element\_selection\_state\_to\_be：判断某个元素的选中状态是否符合预期
* element\_located\_selection\_state\_to\_be：跟上面的方法作用一样，只是上面的方法传入定位到的element，而这个方法传入locator
* alert\_is\_present：判断页面上是否存在alert

**2. Implicit Waits（隐式等待）**

隐式等待是在尝试发现某个元素的时候，如果没能立刻发现，就等待固定长度的时间。默认设置是0秒。一旦设置了隐式等待时间，它的作用范围就是Webdriver对象实例的整个生命周期。

from selenium import webdriver

driver = webdriver.Firefox()

driver.implicitly\_wait(10) *# seconds*

driver.get("http://somedomain/url\_that\_delays\_loading")

myDynamicElement = driver.find\_element\_by\_id("myDynamicElement")