链滴

细说 selenium 的等待条件

作者：zyjImmortal
原文链接：https://ld246.com/article/1596295726746
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

selenium的显示等待

在进行UI自动化测试的时候，我们为了保持用例的稳定性，往往要设置显示等待，显示等待就是说明的要等到某个元素的出现或者元素的某些条件出现，比如可点击、可见等条件，如果在规定的时间之都没有找到，那么就会抛出 Exception.

```
class TestCase:
    def __init__(self):
        self.driver = webdriver.Chrome(executable_path="./driver/chromedriver")
        self.driver.get('http://www.baidu.com')
        # sleep(2)

    def __call__(self):
        print(self.driver.title)

    def test_wait(self):
        wait = WebDriverWait(self.driver, 2)
        wait.until(EC.title_is("百度一下，我就知道"))

        self.driver.find_element_by_id('kw').send_keys('selenium')
        self.driver.find_element_by_id('su').click()
```

上面是我用 selenium写的一个测试用例，展示了 selenium中显示等待的使用方式，其中会使用到 ex ected_conditions模块和 WebDriverWait类，注意这里 expected_conditions是一个py文件的文件，也就是一个模块名，这个模块下面有很多的条件类，而我们用例中使用的 title_is就是一个条件类。

WebDriverWait是一个类，这个类的作用就是根据一定的条件，不断的检查这个条件是否被满足了。ebDriverWait类只有两个方法，一个是 until直到满足某个条件，另一个是 until_not直到不满足某个件。

class WebDriverWait(object):

原文链接：细说 selenium 的等待条件

```
    def __init__(self, driver, timeout, poll_frequency=POLL_FREQUENCY, ignored_exceptions=None):
```

WebDriverWait有四个参数分别是， driver驱动, timeout超时时间, poll_frequency=POLL_FREQUNCY轮训时间，也就是去判断条件是否满足的时间间隔，默认是0.5秒, ignored_exceptions=None等待的过程中需要忽略的异常，是一个可迭代的异常类集合，比如我们可以设置一个list，里面是 [NouchElementException,NoSuchAttributeException,InvalidElementStateException....]，默认情况，是一个元组，只包含一个 NoSuchElementException,因为只有元素出现，才能去判断条件是否满，在不断轮训的过程中，肯定会发生 NoSuchElementException，这个时候必须忽略掉这个异常，然程序就会中断。

其中 driver和 timeout是毕传的位置参数，另外两个是选择传递的关键字参数，如果不传都有指定的认值。

下面就进入我们今天的主题，selenium中的等待条件的讨论

等待条件

条件类的实现原理

在 selenium.webdriver.support.expected_conditions这个模块里，存放着所有的等待条件，每个件类的结构都是一样的一个 __init__构造方法和一个 __call__方法。

在python中，如果想把类型的对象当做函数来使用，那么就可以给这个类实现 __call__方法，如下：

```
class TestCase:
    def __init__(self):
        self.driver = webdriver.Chrome(executable_path="./driver/chromedriver")
        self.driver.get('http://www.baidu.com')
        # sleep(2)

    def __call__(self):
        print(self.driver.title)

if __name__ == '__main__':
    case = TestCase()
    case()
```

case()对象的调用，就会执行 __call__方法里面的逻辑打印当前页面的标题，我们取一个selenium的现类：

```
class presence_of_element_located(object):

    def __init__(self, locator):
        self.locator = locator

    def __call__(self, driver):
        return _find_element(driver, self.locator)
```

这个条件类的意思是判断一个元素是否已经渲染到页面当中，在使用这个条件的时候需要先实例化，入元素的定位，然后要进行判断的时候需要对实例对象进行调用并传入 driver,对实例对象进行调用的候就会执行 __call__方法里的条件判断逻辑。

WebDriverWait是如何进行条件判断的

再回到文章开头看一下我们使用显示等待的代码：

```
wait = WebDriverWait(self.driver, 2)
wait.until(EC.title_is('百度一下，你就知道'))
```

先是实例化一个 WebDriverWait对象，然后再调用 until方法并且传递一个条件的实例对象，until方里就会不断的去轮训条件是否满足。

```
def until(self, method, message=''):
    screen = None
    stacktrace = None

    end_time = time.time() + self._timeout
    while True:
        try:
            value = method(self._driver)
            if value:
                return value
        except self._ignored_exceptions as exc:
            screen = getattr(exc, 'screen', None)
            stacktrace = getattr(exc, 'stacktrace', None)
        time.sleep(self._poll)
        if time.time() > end_time:
            break
    raise TimeoutException(message, screen, stacktrace)
```

method这个参数就是我们传递进来的条件的实例对象，value = method(self._driver)这里就是进行象的调用，也就是执行了 __call__方法里的逻辑。

selenium里都有哪些条件

- title_is 判断title是否出现
- title_contains 判断title页面标题是否包含某些字符
- presence_of_element_located 判断某个元素是否被加载到了dom树里，但是并不代表这个元素见
- url_contains 判断当前url是否包含某个url
- url_matches 判断当前url是否符合某种格式
- url_to_be 判断当前url是否出现
- url_changes 判断当前url是否已经发生了变化
- visibility_of_element_located 判断某个元素是否被添加到了dom树里，且宽高都大于0
- visibility_of 判断看某个元素是否可见
- presence_of_all_elements_located 判断至少有一个元素存在于dom树中，返回所有定位到的元素
- visibility_of_any_elements_located 判断至少有一个元素在页面中可见
- visibility_of_all_elements_located 判断是否所有元素都在页面中可见
- text_to_be_present_in_element 判断指定的元素中是否包含了预期的字符串

原文链接：细说 selenium 的等待条件

- text_to_be_present_in_element_value 判断指定的元素属性值中是否包含了预期的字符串
- frame_to_be_available_and_switch_to_it 判断iframe是否可以switch进去
- invisibility_of_element_located 判断某个元素是否在dom中不可见
- element_to_be_clickable 判断某个元素是否可见并且是enable的，也就是说是是否可以点击
- staleness_of 等待某个元素从dom中删除
- element_to_be_selected 判断某个元素是否被选中了，一般用于下拉列表中
- element_located_to_be_selected 与上面的意思一样，只不过上面实例化的时候传入的是元素对，这个传入的是定位
- 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 与上面一样，只不过传值不同而已
- number_of_windows_to_be 判断当前窗口数是否等于预期
- new_window_is_opened 判断是否有窗口增加
- alert_is_present 判断页面是否有弹窗

以上就是selenium支持的所有条件。

然后就是自定义了

说了那么多条件，其实我们也可以自己实现一个条件类，

```
class page_is_load:

    def __init__(self, expected_title, expected_url):
        self.expected_title = expected_title
        self.expected_url = expected_url

    def __call__(self, driver):
        is_title_correct = driver.title == self.expected_title
        is_url_correct = driver.current_url == self.expected_url
        return is_title_correct and is_url_correct
```

上面是自己实现的一个条件类，根据页面的url和标题来判断页面是否被正确加载，

```
class TestCase:
    def __init__(self):
        self.driver = webdriver.Chrome(executable_path="./driver/chromedriver")
        self.driver.get('http://www.baidu.com/')
        # sleep(2)

    def __call__(self):
        print(self.driver.title)

    def test_wait(self):
        wait = WebDriverWait(self.driver, 2)
        wait.until(page_is_load("百度一下，你就知道", "http://www.baidu.com/"))
```