Selenium 实现元素定位

极客时间

扫码试看/订阅

《Selenium自动化测试实战》视频课程

Selenium 实现元素定位

- 要想操作 Web 界面上的元素，首先要定位到该元素，Selenium 提供了定位元素的 API，这些方法都被定义在 WebDriver 类中，这些方法都以 find 开头 例如：find_element_by_id，find_element_by_name 等。本节我们就来看一下如何实现元素的定位。

- 下面通过一张表格来认识一下这些API

#	方法名称	描述
1	find_element_by_id	通过id定位元素
2	find_element_by_xpath	通过xpath定位元素
3	find_element_by_link_text	通过连接文本定位元素
4	find_element_by_partial_link_text	通过部分链接文本定位
5	find_element_by_name	通过标签名称定位
6	find_element_by_tag_name	通过标签名称定位
7	find_element_by_class_name	通过css class定位

find_element 方法解析

- 另外，还有一些是 elements 是复数的，这些可以返回多个元素，例如：find_elements_by_name、find_elements_by_id 等。

- 这些方法都会调用 def find_element(self, by=By.ID, value=None)：该方法的第二个参数默认是 id，也可以传递其他参数，该参数是 ID 类的一些常量：

- By 类源码解析，下面我们来一起看一下它的源码。

元素定位实例演示

- 下面通过实例，来看一下这些方法的用法。

Selenium WebDriver

Selenium WebDriver 属性

- 除了上一节，我们介绍的元素定位方法外，WebDriver 类还有一些其他的常用属性和方法，这一节我们就来学习一下。

- 下面表格列出了 WebDriver 的常用属性：

#	属性	属性描述
1	driver.name	浏览器名称
2	driver.current_url	当前url
3	driver.title	当前页面标题
4	driver.page_source	当前页面源码
5	driver.current_window_handle	窗口句柄
6	driver.window_handles	当前窗口所有句柄

- 下面通过实例来演示一下这些属性的用法

Selenium WebDriver 方法

- 下面我们再来看一下 WebDriver 其他的一些方法的用法，如下表所示。

#	方法	方法描述
1	driver.back()	浏览器后退
2	driver.forward()	浏览器前进
3	driver.refresh()	浏览器刷新
4	driver.close()	关闭当前窗口
5	driver.quit()	退出浏览器
6	driver.switch_to.frame()	切换到frame
7	driver.switch_to.alert	切换到alert
8	driver.switch_to.active_element	切换到活动元素

- 下面通过实例来演示一下他们的用法，这里主要演示控制浏览器的一些方法，其他的方法在后面章节会详细讲述。

Selenium WebElement

Selenium WebElement 属性

- 当我们使用 WebDriver 的 find 方法定位到元素后，会返回一个 WebElement 对象，该对象用来描述 Web 页面上的一个元素，这一节，我们就来看一下 WebElement 的常用属性和方法。

- WebElement 常用属性如下表所示。

#	属性	属性描述
1	id	标示
2	size	宽高
3	rect	宽高和坐标
4	tag_name	标签名称
5	text	文本内容

- 下面通过实例来演示一下这些属性的用法

Selenium WebElement 方法

- 接下来我们再来看一下 WebElement 的相关方法，如下表所示

#	方法	方法描述
1	send_keys()	输入内容
2	clear()	清空内容
3	click()	单击
4	get_attribute()	获得属性值
5	is_selected()	是否被选中
6	is_enabled()	是否可用
7	is_displayed()	是否显示
8	value_of_css_property()	css属性值

- 下面通过代码来演示一下这些方法的用法。

Selenium 操作 form 表单

Selenium 操作 form 表单

- form 表单是我们经常测试的用例，绝大多数 Web 程序都有这方面功能，例如：用户登录、注册都会用到 form 表单。本节为你讲述如何测试 form 表单。

- form 表单的是流程是这样的：

1. 定位表单元素
2. 输入测试值
3. 判断表单元素属性
4. 获得表单元素属性
5. 提交表单进行验证

Selenium 操作 form 表单-实例

- 下面通过代码来演示一下如何操作 form 表单。

Selenium 操作 checkbox 和 radiobutton

Selenium 操作 checkbox 和 radiobutton

- form 表单中也经常会用到 checkbox 和 radiobutton，checkbox 是多选框，radiobutton 是单选框。

- 例如，一个注册表单要收集用户的爱好可以使用 checkbox，输入性别可以使用 radio。

Selenium 操作 checkbox 和 radiobutton

- 这个一节，我将为你讲述这两个元素的用法：
  - checkbox
    - 如果checkbox有id属性可以直接通过id定位，如果没有可以通过input标签名称定位，然后通过type属性过滤。
    - 选择或者反选checkbox，使用click()方法。
  - radiobutton
    - radiobutton有相同的名称，多个值，可以先通过名称获得，然后通过值判断
    - 选择或者反选checkbox，使用click()方法。

Selenium 操作 checkbox 和 radiobutton-实例

- 下面通过实例来演示一下。

Selenium 操作下拉列表

Selenium 操作下拉列表

- 处理下拉列表，需要用到 Selenium 中的一个工具类 Select，下面来看一下这个来中的常用方法。

#	方法/属性	方法/属性描述
1	select_by_value()	根据值选择
2	select_by_index()	根据索引选择
3	select_by_visible_text	根据文本选择
4	deselect_by_value	根据值反选
5	deselect_by_index	根据索引反选
6	deselect_by_visible_text	根据文本反选
7	deselect_all	反选所有
8	options	所有选项
9	all_selected_options	所有选中选项
10	first_selected_option	第一个选择选项

Selenium 操作下拉列表-实例

- 下面通过实例来演示一下。

Selenium 处理弹框

Selenium 处理弹框

- 页面上的弹框有三种：
  - alert：用来提示
  - confirm：用来确认
  - prompt：输入内容

#	方法/属性	方法/属性描述
1	accept()	接受
2	dismiss()	取消
3	text	显示的文本
4	send_keys	输入内容

Selenium 处理弹框-实例

- 下面通过实例来演示一下。

Selenium 三种等待方式

Selenium 三种等待方式

- 在 UI 自动化测试中，必然会遇到环境不稳定，网络慢的情况，这时如果不做任何处理的话，代码会由于没有找到元素而报错。另外，一种情况就是页面使用 ajax 异步加载机制。这时我们就要用到 wait，而在 Selenium 中，我们可以用到一共三种等待，每一种等待都有自己的优点或缺点，下面我们就来学习一下。

time.sleep（固定等待）

- 在开发自动化框架过程中，最忌讳使用 python 自带模块的 time 的 sleep 方式进行等待，虽然可以自定义等待时间，但当网络条件良好时，依旧按照预设定的时间继续等待，导致整个项目的自动化时间无限延长，不建议使用。

（注：脚本调试过程时，还是可以使用的，方便快捷）

implicitly_wait（隐式等待）

- 隐式等待实际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，否则一直等到时间结束，然后执行下一步。这样的隐式等待会有个坑，我们都知道 JavaScript 一般都是放在我们的 body 的最后进行加载，实际这是页面上的元素都已经加载完毕，我们却还在等待全部页面加载结束。

- 隐式等待对整个 driver 周期都起作用，在最开始设置一次就可以了。不要当作固定等待使用，到哪都来一下隐式等待。

WebDriverWait（显式等待）

- WebDriverWait是selenium提供得到显示等待模块引入路径：
- from selenium.webdriver.support.wait import WebDriverWait
- WebDriverWait参数

#	参数	参数说明
1	driver	传入WebDriver实例
2	timeout	超时时间，等待的最长时间
3	poll_frequency	调用until或until_not中的方法的间隔时间，默认是0.5秒
4	ignored_exceptions	忽略的异常

- 这个模块中，一共只有两种方法 until 与 until_not。

#	参数	参数说明
1	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
2	message	如果超时，抛出TimeoutException，将message传入异常

Selenium 三种等待方式-实例演示

- 下面通过实例来演示一下三种等待方式的用法。

Selenium 等待条件

Selenium 等待条件

- Selenium中的鼠标和键盘事件被封装在ActionChains类中，正确的使用方法是：ActionChains(driver).click(btn).perform()下面列出ActionChains中常用方法：

1	title_is	判断title，是否出现	布尔
2	title_contains	判断title，是否包含某些字符	布尔
3	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	WebElement
4	visibility_of_element_located	判断某个元素是否被添加到了dom里并且可见，宽和高都大于0	WebElement
5	visibility_of	判断元素是否可见，如果可见就返回这个元素	WebElement
6	presence_of_all_elements_located	判断是否至少有1个元素存在于dom树中	列表
7	visibility_of_any_elements_located	判断是否至少有一个元素在页面中可见	列表
8	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	布尔
9	text_to_be_present_in_element_value	判断指定元素的属性值中是否包含了预期的字符串	布尔
10	frame_to_be_available_and_switch_to_it	判断该frame是否可以switch进去	布尔
11	invisibility_of_element_located	判断某个元素在是否存在于dom或不可见	布尔
12	element_to_be_clickable	判断某个元素中是否可见并且是enable的，代表可点击	布尔
13	staleness_of	等待某个元素从dom树中移除	布尔
14	element_to_be_selected	判断某个元素是否被选中了,一般用在下拉列表	布尔
15	element_selection_state_to_be	判断某个元素的选中状态是否符合预期	布尔
16	element_located_selection_state_to_be	判断某个元素的选中状态是否符合预期	布尔
17	alert_is_present	判断页面上是否存在alert	alert

Selenium 等待条件

- 下面通过实例来演示一下等待条件的用法。

Selenium 鼠标和键盘事件

Selenium 鼠标和键盘事件

- Selenium中的鼠标和键盘事件被封装在ActionChains类中，正确的使用方法是：ActionChains(driver).click(btn).perform()下面列出ActionChains中常用方法：

#	方法	方法描述
1	`click(on_element=None)`	单击鼠标左键
2	`click_and_hold(on_element=None)`	点击鼠标左键，不松开
3	`context_click(on_element=None)`	点击鼠标右键
4	`double_click(on_element=None)`	双击鼠标左键
5	`drag_and_drop(source, target)`	拖拽到某个元素然后松开
6	`drag_and_drop_by_offset(source, xoffset, yoffset)`	拖拽到某个坐标然后松开
7	`key_down(value, element=None)`	按下某个键盘上的键
8	`key_up(value, element=None)`	松开某个键
9	`move_by_offset(xoffset, yoffset)`	鼠标从当前位置移动到某个坐标
10	`move_to_element(to_element)`	鼠标移动到某个元素
11	`move_to_element_with_offset(to_element, xoffset, yoffset)`	动到距某个元素（左上角坐标）多少距离的位置
12	`perform()`	执行链中的所有动作
13	`release(on_element=None)`	在某个元素位置松开鼠标左键
14	`send_keys(*keys_to_send)`	发送某个键到当前焦点的元素
15	`send_keys_to_element(element, *keys_to_send)`	发送某个键到指定元素

Selenium 鼠标和键盘事件

- 下面通过实例来演示一下鼠标和键盘事件的用法。

Selenium 执行 JavaScript 脚本

Selenium 执行 JavaScript 脚本

- WebDriver 有两个方法来执行 JavaScript，分别是：
  - execute_script 同步执行
  - execute_async_script 异步执行

- 通过 JavaScript 通常可以实现页面滚动，下面通过实例来演示一下他们的用法：

Selenium 屏幕截图

Selenium 屏幕截图

- WebDriver 内置了一些在测试中捕获屏幕并保存的方法：

#	方法	方法描述
1	save_screenshot(filename)	获取当前屏幕截图并保存为指定文件，filename指指定保存的路径或者图片的文件名
2	get_screenshot_as_base64()	获取当前屏幕截图base65编码字符串
3	get_screenshot_as_file(fimename)	获取当前的屏幕截图，使用完整的路径
4	get_screenshot_as_png()	获取当前屏幕截图的二进制文件数据

- 下面通过实例来学习一下：

Selenium 定位 frame iframe

Selenium 定位 frame iframe

- frame 标签有 frameset、frame、iframe 三种，frameset 跟其他普通标签没有区别，不会影响到正常的定位，而 frame 与 iframe 对 Selenium 定位而言是一样的，Selenium 有一组方法对 frame 进行操作。

#	方法	方法描述
1	switch_to.frame(reference)	切换frame，reference是传入的参数，用来定位frame，可以传入id、name、index以及selenium的WebElement对象
2	switch_to.default_content()	返回主文档
3	switch_to.parent_frame()	返回父文档

- 下面通过实例来学习一下

极客时间

扫码试看/订阅

《Selenium自动化测试实战》视频课程