通过id

Selenium 自带 id 定位，可以通过元素的 id 属性进行定位，比如下面的代码：

- Python 版本

```
driver.find_element_by_id('kw')
```

- Java 版本

```
driver.findElement(By.id("kw"));
```

通过name

Selenium 自带 name 定位，可以通过元素的 name 属性进行定位，比如下面的代码：

- Python 版本

```
driver.find_element_by_name('wd')
```

- Java 版本

```
driver.findElement(By.name("wd"));
```

注意：通常来说 name 属性与 id 属性在页面中唯一，推荐使用这两个属性进行定位。

通过XPath

XPath 是一个定位语言，英文全称为：XML Path Language，用来对 XML 上的元素进行定位，但也适用于 HTML。下面来看一个例子：

要定位的元素是百度首页的搜索输入框

首先寻找 id 为 form 的 form 元素，然后再寻找它的子元素 span，span 的 class 属性为 `bg s_ipt_wr quickdelete-wrap`，最后找 span 的子元素 input：

- Python 版本

```
driver.find_element_by_xpath\    ("//form[@id='form']/span[@class='bg s_ipt_wr quickdelete-wrap']/input")
```

- Java 版本

```
driver.findElement(By.xpath("//form[@id='form']/span[@class='bg s_ipt_wr quickdelete-wrap']/input"));
```

下面的定位也可以找到这个 input，请注意，这里使用了双斜杠`//`，它可以找到子孙节点，而但斜杠`/`只能找到子节点：

- Python 版本

```
driver.find_element_by_xpath("//form[@id='form']//input[@id='kw']")
```

- Java 版本

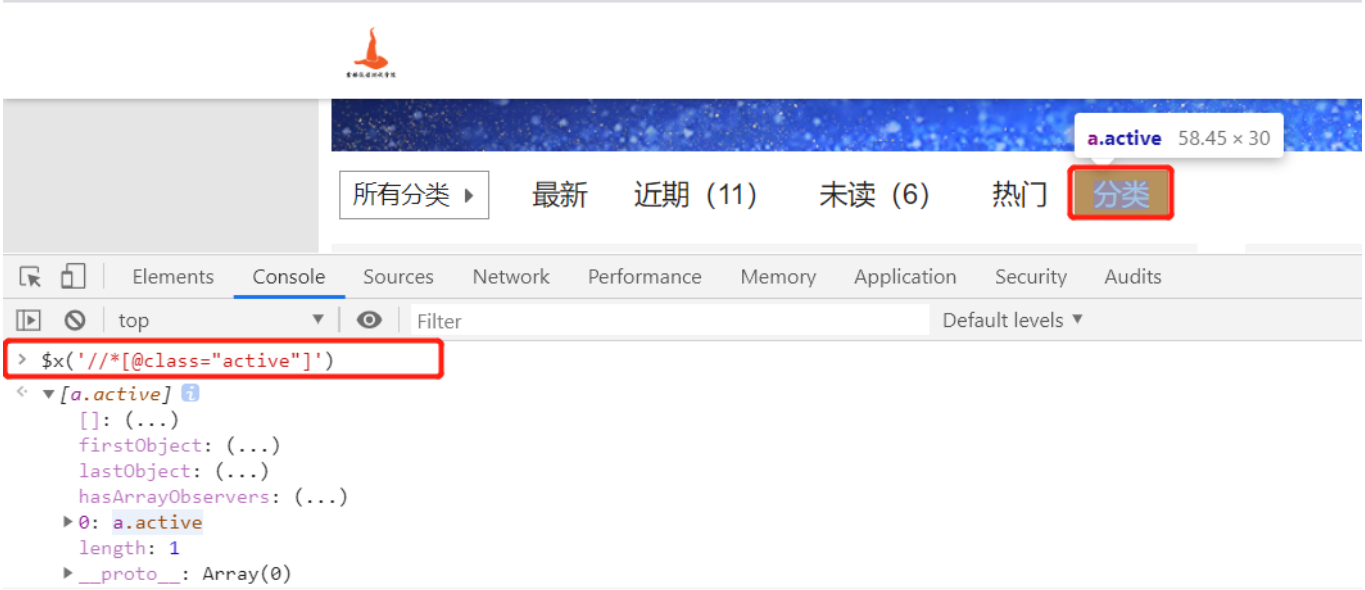
```
driver.findElement(By.xpath("//form[@id='form']//input[@id='kw']"));
```

XPath 表达式更多内容可参考下面表格：

表达式	描述
nodename	选取此节点的所有子节点。

/	从根节点选取。
//	从匹配选择的当前节点选择文档中的节点，而不考虑它们的位置。
.	选取当前节点。
..	选取当前节点的父节点。
@	选取属性。

如何检验 XPath 定位是否正确？可以使用 chrome 的检查模式 -> Console，输入`$x('XPath 表达式')`即可，例如：



通过css_selector

XPath 可以定位绝大多数元素，但是XPath采用从上到下的遍历模式，速度并不快，而 css_selector 采用样式定位，速度要优于 XPath，而且语法更简洁：

下面是 Selenium 使用 css_selector 的例子：

css_selector 找到 class 属性为 active 的元素，然后 `>` 表示找 class 属性为 active 的元素的子节点

- Python 版本

```
driver.find_element_by_css_selector('.logo-big')
```

- Java 版本

```
driver.findElement(By.cssSelector(".logo-big"));
```

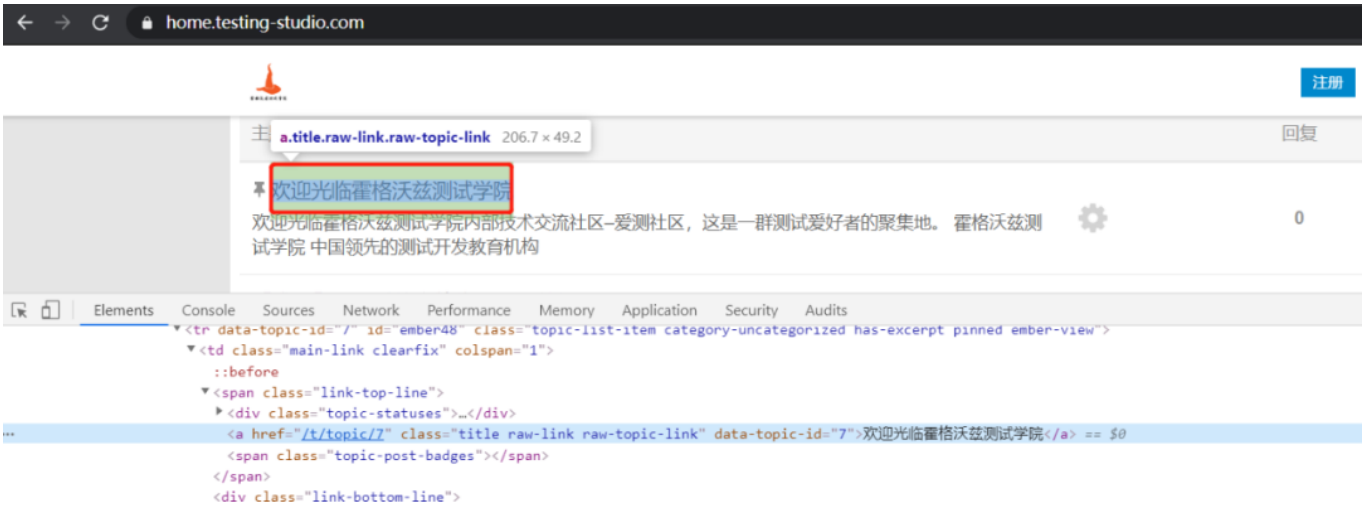
下表列出了常用的 css_selector 表达式的用法：

表达式	描述
.intro	class="intro" 的所有元素
#firstname	id="firstname" 的所有元素
a[target=_blank]	具有属性 target="_blank" 的所有 a 元素
p:nth-child(2)	属于其父元素的第二个 p 元素

使用 Chrome 的检查模式 -> Console 也可以在当前页面检测 css_selector 是否正确，输入`$('css selector 表达式')`即可：

通过link

元素中会出现文字，比如下面的分类，可以利用这段文字进行定位：



- Python 版本

```
1  driver.find_element_by_link_text('欢迎光临霍格沃兹测试学院')
```

- Java 版本

```
1  driver.findElement(By.linkText("欢迎光临霍格沃兹测试学院"));
```

也可以采用部分匹配方式，不必写全：“欢迎光临”、“欢迎光临霍格沃兹测试学院”、“霍格沃兹”

- Python 版本

```
1  driver.find_element_by_partial_link_text('霍格沃兹测试学院')
```

- Java 版本

```
1  driver.findElement(By.partialLinkText("霍格沃兹测试学院"));
```

注意：partial_link_text 与 link_text 的区别：
partial_link_text 不用写全，只需写部分即可，比如上面使用“霍格沃兹”即可匹配到“欢迎光临霍格沃兹测试学院”。

通过tag_name
DOM 结构中，元素都有自己的 tag，比如 input tag, button tag, anchor tag 等等，每一个 tag 拥有多个属性，比如 id, name, value class,等等。下面的高亮部分就是 tag：

Password
Field:
< input type="password" name="password" placeholder="Password" class="form-control mt-3 form-control-lg" >

Login
Button:
< button type="submit" class="btn btn-primary btn-lg btn-block mt-3">LOGIN< /button >

Forgot Password
Link:
< button type="submit" class="btn btn-primary btn-lg btn-block mt-3">LOGIN< /button >

可以使用 tag 进行定位:

- Python 版本

```
driver.find_element_by_tag_name('input')
```

- Java 版本

```
driver.findElement(By.tagName("input"));
```

注意：尽量避免使用 tag_name 定位元素，因为有大量重复的元素！

通过class_name

可以通过元素的 class 属性值进行定位：

这里的 active 用的就是上图 class 的值

- Python 版本

```
driver.find_element_by_class_name('active')
```

- Java 版本

```
driver.findElement(By.className("active"));
```

推荐使用

- ID/Name 是最安全的定位选项。根据 W3C 标准，它在页面中是唯一的，ID 在树结构中也是唯一的。
- CSS Selector 语法简洁，搜索速度快于 XPath。
- XPath 定位功能强大，采用遍历搜索，速度略慢。
- link，class name， tag name：不推荐使用，无法精准定位。

常见操作

Selenium 常见操作有：

- 输入、点击、清除
- 关闭窗口、浏览器
- 获取元素属性
- 获取网页源代码、刷新页面
- 设置窗口大小

输入、点击、清除

输入、点击、清除在 Selenium 中对应的方法分别是 send_keys、click、clear

- Python 版本

```
from selenium import webdriverdriver = webdriver.Chrome()driver.get('http"//www.baidu.com')driver.find_element_by_name('wd').send_keys('霍格沃兹测试学院')driver.find_element_by_id('su').click()driver.find_element_by_name('wd').clear()
```

- Java 版本

```
import org.openqa.selenium.By;import org.openqa.selenium.WebDriver;import org.openqa.selenium.chrome.ChromeDriver;public class AiceTest {    public static void main(String[] args) {        WebDriver driver = new ChromeDriver();        driver.get("http://www.baidu.com");         driver.findElement(By.id("kw")).sendKeys("霍格沃兹测试学院");        driver.findElement(By.id("su")).click();        driver.findElement(By.name("wd")).clear();         try {            Thread.sleep(2000);         } catch (InterruptedException e) {            e.printStackTrace();         }         String title =
```

```
    driver.getTitle();          System.out.println(title);
    driver.close();     }}
```

关闭窗口、浏览器

关闭当前句柄窗口（不关闭进程）close()，关闭整个浏览器进程 quit()

- Python 版本

```
1  #导入对应的依赖from selenium import webdriver#初始化webdriverdriver = webdriver.Chrome()#
   访问网站driver.get('http"//www.baidu.com')#关闭当前窗口driver.close()#关闭浏览器
   driver.quit()
```

- Java 版本

```
1  //导入对应的依赖import org.openqa.selenium.WebDriver;//初始化webdriverWebDriver driver =
   new ChromeDriver();//访问网站driver.get("
   http://www.baidu.com
   ");//关闭当前窗口driver.close();//关闭浏览器driver.quit();
```

获取元素属性

获取元素标签上的属性 get_attribute('value')，元素的坐标 location，元素的大小 size

- Python 版本

```
1  import loggingfrom selenium import webdriverdef test_baidu():    driver =
   webdriver.Chrome()    driver.get('
   https://www.baidu.com
   ')     search = driver.find_element_by_id('su')
   logging.basicConfig(level=logging.INFO)
   logging.info(search.get_attribute('value'))     #获取search的value属性值
   并打印     logging.info(search.get_attribute('value'))     #打印search的位
   置坐标     logging.info(search.location)     #打印search的元素大小
   logging.info(search.size)
```

输出结果为：

```
1  INFO:root:百度一下INFO:root:百度一下INFO:root:{'x': 844, 'y': 188}INFO:root:{'height':
   44, 'width': 108}
```

- Java 版本

```
1  @Test    void baiduTest(){        webDriver = new ChromeDriver();        webDriver.get("
   https://www.baidu.com/
```

```
");         WebElement search = webDriver.findElement(By.id("su"));
    //获取search的value属性值并打印
System.out.println(search.getAttribute("value"));         //打印search的位置坐标         System.out.println(search.getLocation());         //打印search的元素大小         System.out.println(search.getSize());     }
```

输出结果为：

```
百度一下(902, 188)(108, 44)
```

获取网页源代码、刷新页面

- Python 版本

网页源代码 page_source，刷新页面 refresh()

```
import loggingfrom selenium import webdriverdriver = webdriver.Chrome()driver.get('http"//www.baidu.com')#刷新页面
driver.refresh()logging.basicConfig(level=logging.INFO)#打印当前页面的源代码
logging.info(driver.page_source)
```

- Java 版本

```
WebDriver webDriver = new ChromeDriver();webDriver.get("
https://www.baidu.com/
");//刷新页面
webDriver.navigate().refresh();System.out.println(webDriver.getPageSource());
```

设置窗口大小

设置窗口大小主要有最小化、最大化和自定义设置窗口具体的大小。

- Python版本

```
from selenium import webdriverdriver = webdriver.Chrome()driver.get('http"//www.baidu.com')#最小化窗口driver.minimize_window()#最大化窗口driver.maximize_window()#将浏览器设置为1000*1000的大小
driver.set_window_size(1000, 1000)
```

- Java版本

```
import org.openqa.selenium.Dimension;import org.openqa.selenium.WebDriver;import org.openqa.selenium.chrome.ChromeDriver;import static java.lang.Thread.sleep;public
```

```
class AiceTest {    public static void main(String[] args) throws InterruptedException {
        WebDriver driver = new ChromeDriver();        driver.get("
http://www.baidu.com
");          //设置窗口最大化          driver.manage().window().maximize();
        //浏览器的设定大小          sleep(2000);          Dimension dimension
= new Dimension(800, 600);
driver.manage().window().setSize(dimension);          sleep(2000);
  //浏览器全屏          driver.manage().window().fullscreen();
sleep(2000);          driver.close();     }}
```