selenium元素定位

1.1 元素定位

在本章开始之前，我们先来看一张页面。



图1.1 web 页面

这就是360搜索的首页，在这张页面上有输入框、按钮和文字链接，当然还有图片，页面的底部还有一行文字，左侧还一个下拉框。自动化要做的就是模式鼠标和键盘来操作这些元素，或点击，或输入，或右击，甚至是鼠标拖动等操作。

那么我们要想操作这些元素的前提是需要找到它们。那么如何找到他们呢？自动化工具可不像我们一样可以通过肉眼来分辨页面上的元素，并且知道是它们是做什么用的。那么我们来看看这些元素的真实面目。

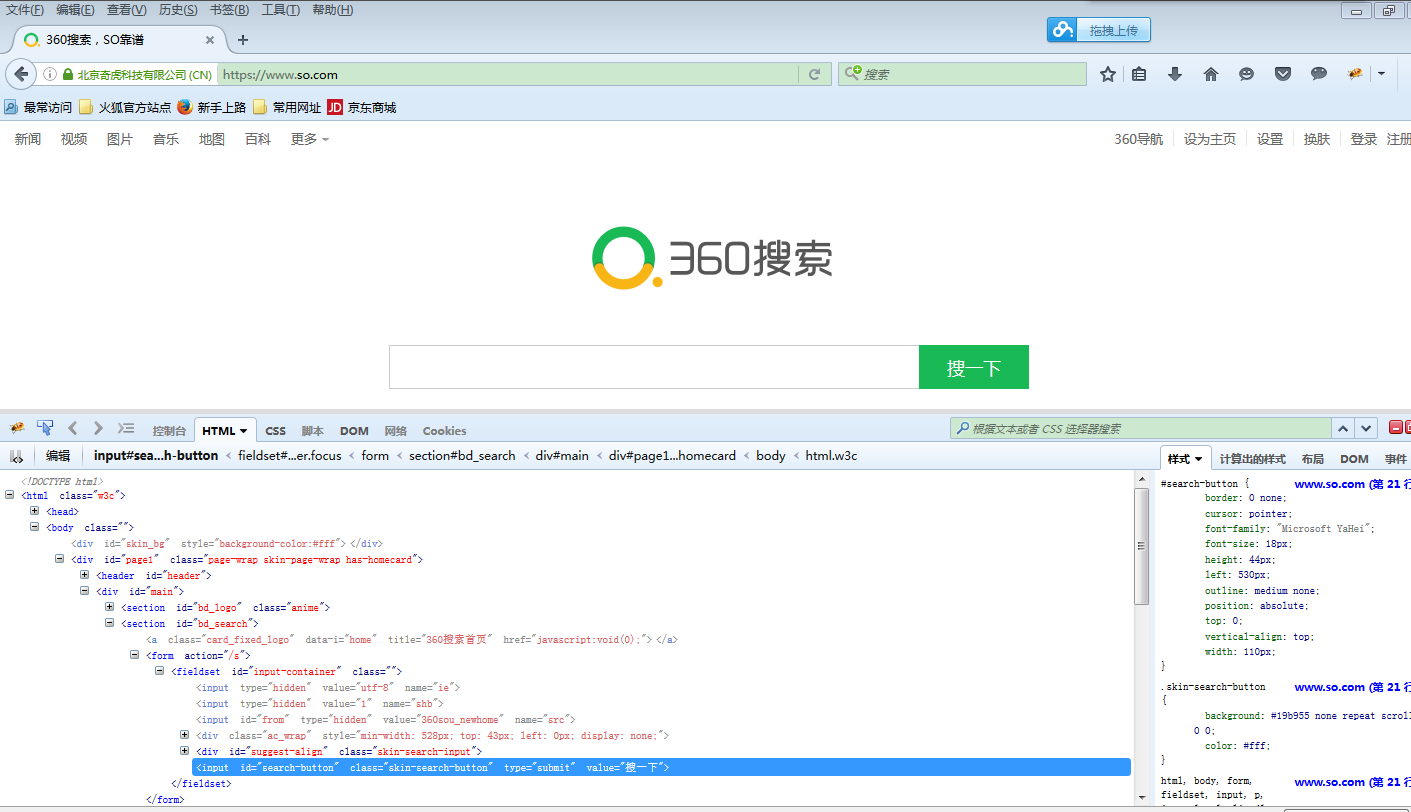


图1.2 通过FireBug 查看页面元素

通过前端工具，我们看可以看到页面上的元素都是由一行一行的代码组成。它们之间有层级的组织起来，每个元素有不同的标签名和属性值。

**那么在Selenium 当中就是通过这信息来找到不同的元素的。**

webdriver 提供了八种元素定位方法：

**id**

**name**

**class name**

**tag name**

**link text**

**partial link text**

**xpath**

**css selector**

在Python 语言中对应的定位方法如下：

**find\_element\_by\_id()**

**find\_element\_by\_name()**

**find\_element\_by\_class\_name()**

**find\_element\_by\_tag\_name()**

**find\_element\_by\_link\_text()**

**find\_element\_by\_partial\_link\_text()**

**find\_element\_by\_xpath()**

**find\_element\_by\_css\_selector()**

下面我们就逐一的来看这些定位方法的使用。在此之前，我们拷取360搜索首页的前端代码，以定位页面上的元素为例进行讲解。



注意这段代码并非360搜索首页的页面源代码，而是通过前端工具查看所得到页面代码与结构。那么这样的HTML 结构有如下特征。

（1）它们由标签对组成：

<html></html>

<body></body>

<div></div>

<form></form>

那么html、div 就是标签的标签名。

（2）标签各种属性属性：

<div id="suggest-align"class="skin-search-input">

<ul class="ac\_menu" unselectable="on">

<input id="input" class="placeholder" type="text" autocomplete="off"  name="q">

就像人一样也会有各种属性，身份证号（id）、姓名（name）、职业（class）等。

（3）标签对之间可以有文本数据。

<a>新闻</a>

<a>视频</a>

<a>360导航</a>

（4）标签有由层级关系

<html>

<body>

</body>

</html>

<div>

<from>

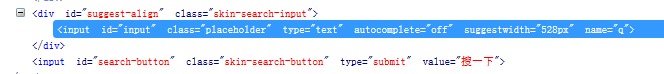
<input />

</from>

<div>

对于上面结构，如果把 input 看作是子标签，那么 form 就是它的父标签。

理解了上面这些特性是学习定位方法的基础。我们以360输入框和360搜索按钮为例来学习不同的定位方法，两个元素的代码如下。

1.1.1 通过 id 定位元素

如果把页面上看元素看作一个人的话，如果我们想找一个人如何去找，那么这个人一定有其别于其它人的“属性”，比如他的身份证号一定和别人不一样，他的名字和别人不一样。那么我们就可以通过身证号和名字来找到一个人。那么id 就可以看做是一个人的身份号，当然这个id 并不像我们现实中的身份证号有那么强的唯一性，如果在一个页面上发现有两个元素的id="input"也是不足为奇的，这个取决前端代码的规范程度。

对360搜索首页上的输入框与360搜索按钮来说，定位方法如下：

find\_element\_by\_id("input")

find\_element\_by\_id("search-button ")

find\_element\_by\_id()方法用于元素中id 属性的定位。

1.1.2 通过 name 定位元素

name 的定位与id 类似，每一个人都会有名字，那么name 就可作是一个元素的名字。通过name 定位输入框：

find\_element\_by\_name("q")

find\_element\_by\_name()方法用于元素中name 属性的定位，如果360搜索按钮并没有提供name 属性，那么我们就不能通过name 去定位360搜索按钮。

1.1.3 通过class 定位元素

class 也是不少元素会有的一个属性，它的定位和name 以及id 类似，下面通过class 去定位360输入框和360搜索按钮：

find\_element\_by\_class\_name("placeholder")

find\_element\_by\_class\_name("skin-search-button")

find\_element\_by\_class\_name()方法用于元素中class 属性的定位。

1.1.4 通过tag 定位元素

tag 定位取的是一个元素的标签名，通过标签名去定位单个元素的唯一性最低，因为在一个页面中有太多的元素标签为类似<div>和<input>，所以很难通过标签名去区分不同的元素。

通过标签名定位360首页上的输入框与360搜索按钮：

find\_element\_by\_tag\_name("input")

find\_element\_by\_tag\_name("input")

find\_element\_by\_tag\_name()方法通过元素的tag name 来定位元素。通过上面的例子，我们并不能区别不同的元素，因为在一个页面上标签名相同很难以避免。所以，我们通常用一下方法来确定一个元素：

----find\_elements\_by\_tag\_name() #返回多个值，使用方法：

inputs = aa.find\_elements\_by\_tag\_name("input")

for i in inputs:

if i.get\_attribute("type") == "submit":

i.click()

# get\_attribute ----- 获取某个对象的指定属性的值

1.1.5 通过 link 定位元素

link 定位与前面介绍的几种定位方法有所不同，它专门用来定位超链接。360输入框上面的几个文本链接的代码如下：



通过查看上面的代码，我们发现不可以通过name属性定位。通过link 定位实现如下：

find\_element\_by\_link\_text("360导航")

find\_element\_by\_link\_text("设为主页")

find\_element\_by\_link\_text("设置")

find\_element\_by\_link\_text()方法通过元素标签对之间的文本信息来定位元素。不过，需要强调的是Python 对于中文的支持并不好，如查Python 在执行中文的地方出现在乱码，可以在中文件字符串的的前面加个小“**u**”可以有效的避免乱码的问题，加 u 的作用是把中文字符串转换中unicode 编码，如：

find\_element\_by\_link\_text(u"新闻")

1.1.6 通过partial link 定位元素

parial link 定位是对link 定们的一个种补充，有些文本连接会比较长，这个时候我们可以取文本链接的有一部分定位，**只要这一部分信息可以唯一的标识这个链接**。

<a class="mnav" name="tj\_lang" href="#">**一个很长很长的文本链接**</a>

通过partial link 定位如下：

find\_element\_by\_partial\_link\_text("一个很长的")

find\_element\_by\_partial\_link\_text("文本连接")

如定位以下文本：



可以用：find\_element\_by\_partial\_link\_text("Selenium\_360")

find\_element\_by\_link\_text()方法通过元素标签对之间的部分文本信息来定位元素。

#############################################################

前面所介绍的几种定位方法相对来说比较简单，我们理想状态下在一个页面当中每一个元素都会有一个唯一id 和name 属性值，我们通过它的属性值来找到他们，但在实际的项目中并非想象的这般美好。有时候一个元素并没有id 或name 属性，或者会有多个元素的id 和name 属性值是一样的，如：



又或者每一次刷新页面，id 的值都会随机变化。那么在这种情况下我们如何来定位元素呢？

下面介绍xpath 与CSS 定位相比上面介绍的方式来说比较难理解，但他们的灵活的定位能力远比上面的几种方式要强大得多。

1.1.7 XPath 定位

XPath 是一种在XML文档中定位元素的语言。因为HTML 可以看做XML 的一种实现，所以selenium用户可是使用这种强大语言在web应用中定位元素。

**1、绝对路径定位**：

XPath 有多种定位策略，最简单和直观的就是写元素的绝对路径。如果仍然把一个元素看做一个人的话，那么现在有一个人，他没有任何属性特征，那么这个人一定会存在于某个地理位置，如：xx 省xx 市xx 区xx 路xx 号。那么对于一个元素在一个页面当中也会有这样的一个绝对地址。

参考360搜索前端工具所展示的代码，我们可以用下面的方式来找到360输入框和360搜索按钮。

find\_element\_by\_xpath("/html/body/div[2]/div[1]/section[2]/form/fieldset/div[2]/input")

find\_element\_by\_xpath("/html/body/div[2]/div[1]/section[2]/form/fieldset/input[4]")

**find\_element\_by\_xpath()**方法用于 XPath 语言定位元素。XPath 的绝对路径主要用标签名的层级关系来定位元素的绝对路径。最外层为html 语言，body 文本内，一级一级往下查找，如果一个层级下有多个相同的标签名，那么就按上下顺序确定是第几个，div[2]表示第二个div 标签。

**2、利用元素属性定位：**

除了使用绝对路径的以外，XPath也可以使用元素的属性值来定位。同样以360输入框和搜索按钮为例子：

find\_element\_by\_xpath("//input[@id='input']")

find\_element\_by\_xpath("//input[@id='search-button']")

//表示当前页面某个目录下，input 表示定位元素的标签名，[@id='input'] 表示这个元素的id 属性值等于input。下面通过name 和class 属性值来定位。

find\_element\_by\_xpath("//input[@id='input']")

find\_element\_by\_xpath("//input[@class='placeholder']")

find\_element\_by\_xpath("//\*[@class='skin-search-button']")

如果不想指定标签名也可以用星号（\*）代替。当然，使用XPath 不仅仅只局限在id、name 和class 这三个属性值，元素的任意属性值都可以使用，只要它能唯一的标识一个元素。

find\_element\_by\_xpath("//input[@suggestwidth ='528px']")

find\_element\_by\_xpath("//input[@autocomplete='off']")

find\_element\_by\_xpath("//input[@type='submit']")

**3、层级与属性结合**：

如果一个元素本身并没有可以唯一标识这个元素的属性值，我们可以找其上一级元素，如果它的上级有可以唯一标识属性的值，也可以拿来使用。参考以下文本。

假如360输入框本身没有可利用的属性值，我们可以查找它的上一级属性。比如，“小明”刚出生的时候没有名字，没上户口（没身份证号），那么亲朋好友来找“小明”可以先到小明的爸爸，因为他爸爸是有很多属性特征的，找到了小明的爸爸，抱在怀里的一定就是小明了。通过XPath 描述如下：

find\_element\_by\_xpath("//div[@class='skin-search-input']/input") -- 输入框

find\_element\_by\_xpath("//fieldset [@id='input-container']/input") -- 搜索框

div[@class='skin-search-input'] 通过class 属性定位到是父元素，后面/input 也就表示父元素下面标签名为input 的子元素。如果父元素没有可利用的属性值，那么可以继续向上查找“爷爷”元素。

find\_element\_by\_xpath("//fieldset [@id='input-container']/div/input")

find\_element\_by\_xpath("//form[@action='/s']/fieldset /input")

我们可以通过这种方法一级一级的向上打找，直到找到最外层的<html>标签，那么就是一个绝对路径的写法了。

**4、运行逻辑运算符**

如果一个属性不能唯一的区分一个元素，我们还可以使用逻辑运算符连接多个属性来区别于其它属性。

……

<input id="input" class="placeholder" type="text" autocomplete="off" suggestwidth="528px" name="q">

<input id="search-button" class="skin-search-button" type="submit" value="搜一下">

……

如上面的三行元素，同时使用id 和class 来唯一的标识这个元素。那么这个时候就可以通过逻辑运算符号连接。

find\_element\_by\_xpath("//input[@id=' input'and @class='placeholder']")

当然，我们也可以用and 连接更多的属性来唯一的标识一个元素。

用 Firebug 前端调试工具和FirePath 插件可以方便的辅助XPath 语法。

打开FireFox 浏览器的FireBug 插件，点击插件左上角的鼠标箭头，再点击页面上需要定位的元素，在元素行上右键弹出快捷菜单，选择“复制XPath”，将会获得当前元素的XPath 语法，如下图：



图1.3 通过FireBug 复制XPath 语法

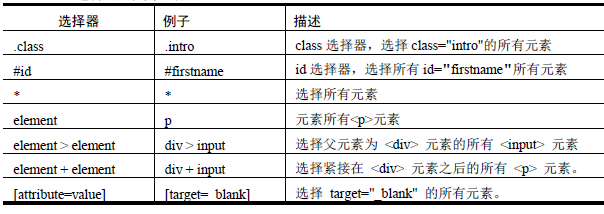
FirePath 插件的使用就更加方便和快捷了，选中元素后，直接在XPath 的输入框中生成当前元素的XPath 语法。

1.1.8 CSS 定位

CSS(Cascading Style Sheets)是一种语言，它被用来描述HTML和XML文档的表现。CSS 使用选择器来为页面元素绑定属性。这些选择器可以被selenium 用作另外的定位策略。

CSS 可以比较灵活选择控件的任意属性，一般情况下定位速度要比XPath 快，但对于初学者来说比较难以学习使用，下面我们就详细的介绍CSS 的语法与使用。

CSS 选择器的常见语法：



下面同样以360输入框和搜索按钮为例介绍CSS 定位的用法。

……

<input id="input" class="placeholder" type="text" autocomplete="off" suggestwidth="528px" name="q">

<input id="search-button" class="skin-search-button" type="submit" value="搜一下">

……

**1、通过class 属性定位：**

find\_element\_by\_css\_selector(".placeholder")

find\_element\_by\_css\_selector(".skin-search-button")

find\_element\_by\_css\_selector()方法用于CSS语言定位元素，点号（.）表示通过class 属性来定位元素。

通过id 属性定位：

find\_element\_by\_css\_selector("#input")

find\_element\_by\_css\_selector("#search-button")

井号（#）表示通过id 属性来定位元素。

**2、通过标签名定位：**

find\_element\_by\_css\_selector("input")

在CSS 语言中用标签名定位元素不需要任何符号标识，直接使用标签名即可，但我们前面已经了解到标签名重复的概率非常大，所以通过这种方式很难唯一的标识一个元素。

**3、通过父子关系定位：**

find\_element\_by\_css\_selector("div >input")

上面的写法表示有父亲元素，它的标签名叫div，查找它的所有标签名叫input的子元素。

**4、通过属性定位：**

find\_element\_by\_css\_selector("input[autocomplete=off]")

find\_element\_by\_css\_selector("input[suggestwidth=528px]")

find\_element\_by\_css\_selector("input[type=submit]")

在CSS 当中也可以使用元素的任意属性，只要这些属性可以唯一的标识这个元素。

**5、组合定位：**

我们当然可以把上面的定位策略组合起来使用，这样就大大加强了元素的唯一性。

find\_element\_by\_css\_selector("div.skin-search-input >input.placeholder")

find\_element\_by\_css\_selector("fieldset #input-container >input#search-button")

有一个父元素，它的标签名叫div，它有一个class 属性值叫skin-search-input，它有一个子元素，标签名叫input，并且这个子元素的class 属性值叫placeholder。我们要找的就是具有这么多特征的一个子元素。

我们通过可以使用Firebug 工具帮助我们生成CSS 语法。通过Firebug 定位元素，在元素上右键点击选择“复制CSS”

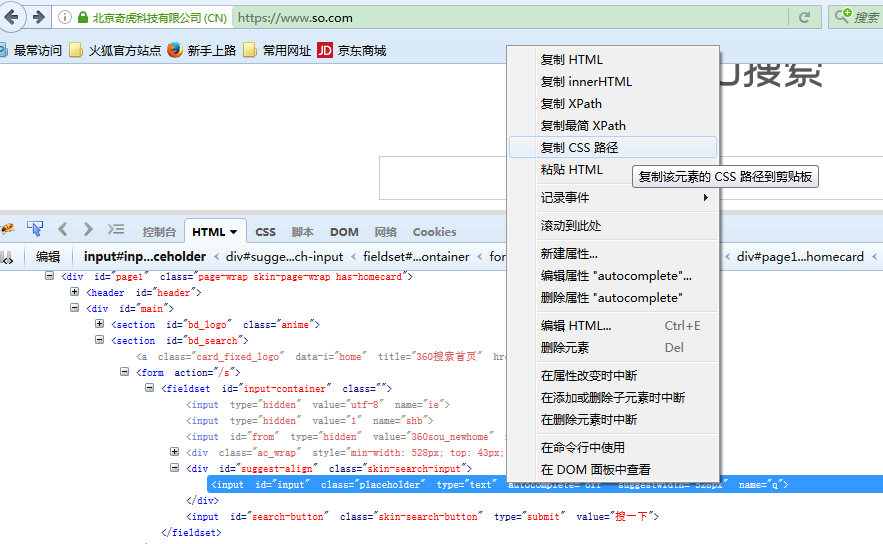
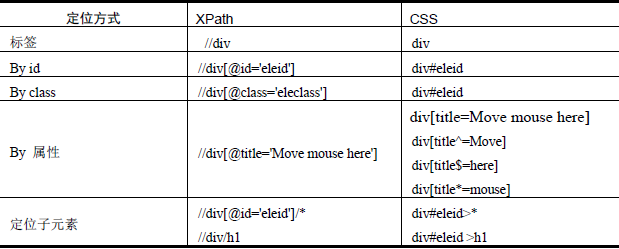


图1.5 通过FireBug 复制CSS 路径

XPath 与CSS 的类似功能的简单对比



通过前面的学习XPath 和CSS 都提供了非常强大和灵活的定位方法。相比较CSS 语法更加简洁，但真正的理解和使用学难度要更大一点。

对于web 自动化来说，学会元素的定位，那么，自动化测试已经入门了，剩下的就selenim的WebDriver中所提供的各种方法的使用，如鼠标模拟，键盘模拟等。