UI自动化测试框架--元素定位

|  |  |  |  |
| --- | --- | --- | --- |
| **文件版本：** | **V1.0** | **文件编号：** | UI自动化测试框架--元素定位 |
| **首版发布日期：** | **2016.11.17** | **编 制：** | **吴丁飞** |
| **审 核：** |  | **批 准：** |  |

**铭师堂教育 版权所有**

**内部资料 注意保密**

修订记录：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **版本** | **修改内容** | **修订日期** | **编写人** | **审核人** | **批准人** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

目 录

[1. 编写目的 1](#_Toc465948663)

[2. 元素定位方法 1](#_Toc465948664)

[**2.1.** **name** 1](#_Toc465948665)

[**2.2.** **id** 1](#_Toc465948666)

[**2.3.** **tagName** 2](#_Toc465948667)

[**2.4.** **className** 2](#_Toc465948668)

[**2.5.** **linkText** 3](#_Toc465948669)

[**2.6.** **partialLinkText** 3](#_Toc465948670)

[**2.7.** **xpath** 3](#_Toc465948671)

[**2.8.** **css** 7](#_Toc465948672)

[**2.9.** **元素定位准则** 8](#_Toc465948673)

[3. 注解介绍 8](#_Toc465948674)

# **编写目的**

为了避免在使用UI自动化测试框架来进行元素定位时出现一些问题，更好的利用框架进行元素定位，提高自动化工作效率，特编写此《UI自动化测试框架—元素定位》。通过对各种元素定位方法、使用规范及注意事项的说明，以便于自动化测试人员更好的了解自动化测试元素定位，并规避相关容易犯错的地方，及出现问题时可以找到相关帮助，使自动化测试工作效率得以提升。

本文的阅读针对自动化测试人员，在工作过程中如有其他相关问题未在本文中详尽说明，也请大家提出宝贵的意见，促进文档的完善，以便大家更好的利用该文档。

# **元素定位方法**

在使用框架进行自动化测试时，首先需要理解元素定位，框架中封装的所有方法都是基于元素或者元素集合进行操作的。框架支持web端、android端(IOS端后续支持),是web端则在声明元素对象的时候声明为WebElement、List<WebElement>;是android端则在声明元素对象的时候声明为AndroidElement、List<WebElement>。框架基于注解进行元素定位，主要为：web端有@FindBy、@FindBys、@FindAll，android端有@AndroidFindBy、@AndroidFindBys、@AndroidFindAll。注解在web端、android端使用类似。首先介绍常用定位方式，主要有如下八种，现分别介绍(以web端为例,android端类似)。

## **name**

页面源码如下：(开课啦首页搜索框)

< input id="searchTextIndex" type="text" value="" laceholder="可按知识点、关键词搜索" name="q"/>

使用name属性来引用这个搜索框

**@FindBy**(name= "q")

**@CacheLookup**

**public** WebElement searchBox

## **id**

页面源码如下：(开课啦首页搜索框)

< input id="searchTextIndex" type="text" value="" laceholder="可按知识点、关键词搜索" name="q"/>

使用id属性来引用这个搜索框

**@FindBy**(id= "searchTextIndex")

**@CacheLookup**

**public** WebElement searchBox;

## **tagName**

该方法可以通过元素的标签名称来查找元素。该方法跟之前两个方法的区别是，这个方法搜索到的元素通常不止一个，所以一般会将其声明为List<WebElement>、List<AndroidElement>，再对list对象进行使用。比如我们现在要查找页面上有多少个input，就可以用input这个tagName来进行查找。

页面源码如下：(开课啦首页搜索框)

< input id="searchTextIndex" type="text" value="" laceholder="可按知识点、关键词搜索" name="q"/>

使用tagName属性来引用input对象

**@FindBy**(name= "input")

**@CacheLookup**

**public** List<WebElement> input;

附：单选框、复选框、文本框和密码框的元素标签都可以是input，此时单靠tagName无法准确地得到我们想要的元素，需要结合type属性才能过滤出我们要的元素。参见框架中的接口实现类中的接口:

public abstract <T> void clickFindObj (T Elements,String attribute,String value)

## **className**

className属性是利用元素的css样式表所引用的伪类名称来进行元素查找的方法。对于任何HTML页面的元素来说，一般程序员或页面设计师会给元素直接赋予一个样式属性或者利用css文件里的伪类来定义元素样式，使元素在页面上显示时能够更加美观。一般css样式表可能会长成下面这个样子

.buttonStyle{

width: 50px;

height: 50px;

border-radius: 50%;

margin: 0% 2%;

}

定义好后，就可以在页面元素中引用上述定义好的样式，如下：

< button name=" sampleBtnName" id=" sampleBtnId" class=" buttonStyle"> I'm Button </button>

使用className属性来引用这个按钮

**@FindBy**(className= " buttonStyle")

**@CacheLookup**

**public** WebElement sampleBtnName;

注意：使用className来进行元素定位时，有时会碰到一个元素指定了若干个class属性值的“复合样式”的情况，如下面这个link：<a class="Bn btnLogin" href="http://auth.kaike.la/auth/login\_page.do?systemflag=1">请登录</a>。这个link元素指定了两个不同的css伪类名作为它的样式属性值，此时则不能使用className来进行元素定位，需使用其他定位方式。

## **linkText**

**linkText**即通过超文本链接上的文字信息来定位元素，这种方式一般专门用于定位页面上的超文本链接。

页面源码如下：(开课啦首页登录按钮)

<a class="Bn btnLogin" href="http://auth.kaike.la/auth/login\_page.do?systemflag=1">请登录</a>

使用**linkText**属性来引用这个登录按钮

**@FindBy**(linkText="请登录")

**@CacheLookup**

**public** WebElement href;

## **partialLinkText**

这个方法是上一个方法的扩展。当不能准确知道超链接上的文本信息或者只想通过一些关键字进行匹配时，可以使用这个方法来通过部分链接文字进行匹配。

页面源码如下：(开课啦首页登录按钮)

<a class="Bn btnLogin" href="http://auth.kaike.la/auth/login\_page.do?systemflag=1">请登录</a>

使用partialLinkText属性来引用这个登录按钮

**@FindBy**(partialLinkText= "请登")

**@CacheLookup**

**public** WebElement href;

注意：使用这种方法进行定位时，可能会引起的问题是，当你的页面中不止一个超链接包含请登时，该方法只会返回第一个查找到的元素，而不会返回所有符合条件的元素。如果要想获得所有符合条件的元素，要使用List<WebElement>、List<AndroidElement>进行声明。

## **xpath**

**xpath**是非常强大的元素查找方式，使用这种方法几乎可以定位到页面上的任意元素。XPath是XML Path的简称，由于HTML文档本身是一个标准的XML页面，所以可以使用XPath的语法来定位页面元素。

定位方法决定了自动化的最终效果，xpath是我们常用的元素定位方法，而Xpath就是对应的唯一动态的核心，一个好的Xpath可以极大的降低我们自动化代码维护的成本。

元素的xpath可通过firefox浏览器下的firepath插件直接查询。一般不推荐直接使用firepath的值进行元素定位，因为firepath往往是使用的绝对路径的方式来定位页面元素，一旦页面结构发生变化，该路径也随之失效，必须重新写。

xpath这种定位方式，会将整个页面的所有元素进行扫描以定位我们所需要的元素，所以这是一个非常费时的操作，如果在脚本中大量使用xpath做元素定位的话，将导致脚本执行速度大大降低。

Xpath详细内容详见：<http://www.w3school.com.cn/xpath/>

页面源码如下：(开课啦登录页)

 使用xpath来引用这个用户名输入框

**@FindBy**(xpath= "html/body/div[3]/div/form/label[1]/input")

**@CacheLookup**

**public** WebElement userName;

**常用的路径表达式：**

|  |  |
| --- | --- |
| **表达式** | **描述** |
| nodename | 选取此节点的所有子节点。 |
| / | 表示让Xpath解析引擎从文档的根节点开始解析；当/出现在xpath路径中时，则表示寻找父节点的直接子节点 |
| // | 表示让xpath引擎从文档的任意符合的元素节点开始进行解析；当//出现在xpath路径中时，表示寻找父节点下任意符合条件的子节点，不管嵌套了多少层级 |
| . | 选取当前节点。 |
| .. | 选取当前节点的父节点。 |
| @ | 选取属性。 |

**XPath轴**

|  |  |
| --- | --- |
| **轴名称** | **结果** |
| ancestor | 选取当前节点的所有先辈（父、祖父等）。 |
| ancestor-or-self | 选取当前节点的所有先辈（父、祖父等）以及当前节点本身。 |
| attribute | 选取当前节点的所有属性。 |
| child | 选取当前节点的所有子元素。 |
| descendant | 选取当前节点的所有后代元素（子、孙等）。 |
| descendant-or-self | 选取当前节点的所有后代元素（子、孙等）以及当前节点本身。 |
| following | 选取文档中当前节点的结束标签之后的所有节点。 |
| namespace | 选取当前节点的所有命名空间节点。 |
| parent | 选取当前节点的父节点。 |
| preceding | 选取文档中当前节点的开始标签之前的所有节点。 |
| preceding-sibling | 选取当前节点之前的所有同级节点。 |
| self | 选取当前节点。 |

**XPATH常用函数**

|  |  |  |
| --- | --- | --- |
| **函数** | **实例** | **说明** |
| contains() | //div[contains(@id,'in')] | 表示选择id中包含有’in’的div节点 |
| text() | //a[text()='baidu'] | 由于一个节点的文本值不属于属性，比如“<a class=”baidu“ href=”http://www.baidu.com“>baidu</a>”,所以用text()函数来匹配节点 |
| last() | //input[@name='identity' or @class='Volvo'][last()] | 从【name为identity，或者class为Volvo的节点】上取最后一个 |
| starts-with() | //div[starts-with(@id,'in')] | 表示选择以’in’开头的id属性的div节点 |
| not() | //input[@name=‘identity’ and not(contains(@class,‘a’))] | 表示匹配出name为identity并且class的值中不包含a的input节点。not()表示否定，not()函数通常与返回值为true or false的函数组合起来用，比如contains(),starts-with()等，但有一种特别情况请注意一下：要匹配出input节点含有id属性的，写法如下：//input[@id]，如果要匹配出input节点不含用id属性的，则为：//input[not(@id)] |

**Xpath用法实例**

|  |  |
| --- | --- |
| **实例** | **含义** |
| //form[1] | 查找页面上第一个form元素 |
| //form[1]//input | 查找页面上第一个form元素内的所有子input元素(只要在form元素内的input都算，不管还嵌套了多少个其他标签，使用相对路径表示，双//号) |
| //form[1]/input | 查找页面上第一个form元素内的直接子input元素(即只包括form元素的下一级input元素，使用绝对路径表示，单/号) |
| //form[@id='loginForm']/input[1] | 查找页面上id为loginForm的form元素下的第一个input元素 |
| //input[@name='continue'][@type='button']  //input[@name='continue' and @type='button'] | 查找页面具有name属性为contiune并且type属性为button的input元素 |
| //input[@name='continue' or @type='button'] | 查找页面具有name属性为contiune或者type属性为button的input元素 |
| //a[contains(text(), ’退出’)] | 这种方式一般用于知道元素上显示的部分或全部文本信息时使用 |
| //div/div/. | 表示选择//div/div节点的当前层的节点，与//div/div的结果相同 |
| //div/div/.. | 表示选择//div/div节点的上一层节点，也就是选择一个div节点，该div节点的子节点有div节点。 |
| //div[@id='radio']//label[text()='Saab']/preceding-sibling::input[1] | 选择label的text为Saab的节点之前的同级节点中为input节点的第一个 |

## **css**

cssSelector这种元素定位方式跟xpath比较类似，但执行速度较快，而且各种浏览器对它的支持都相当到位。

|  |  |  |
| --- | --- | --- |
| **表达式** | **描述** | **实例说明** |
| # | 表示id | #input 选择id为input的节点 |
| . | 表示class | .Volvo 选择class为Volvo的节点 |
| > | 表示子元素，层级 | div#radio>input  选择id为radio的div下的所有的input节点 |
| 空格 | 表示子元素，但是是所有的后代子元素，相当于xpath中的相对路径 | div#radio input  选择id为radio的div下的所有的子孙后代input节点 |

cssSelector还有一个用处是定位使用了复合样式表的元素，之前在className里面提到过。如下示例页面元素通过cssSelector来引用className的那个button。页面代码如下：

< button id="login" class="bnt bnt\_big bnt\_submit" type="submit">登录</button>

使用css来引用这个button

**@FindBy**(css= "button.bnt.bnt\_big.bnt\_submit")

**@CacheLookup**

**public** WebElement loginButton;

此外，cssSelector还有一些高级用法，可以更加方便地帮助我们定位元素，如可以利用^用于匹配一个前缀，$用于匹配一个后缀，\*用于匹配任意字符。例如：

匹配一个有id属性，并且id属性是以”id\_prefix\_”开头的超链接元素：a[id^='id\_prefix\_']

匹配一个有id属性，并且id属性是以”\_id\_sufix”结尾的超链接元素：a[id$='\_id\_sufix']

匹配一个有id属性，并且id属性中包含”id\_pattern”字符的超链接元素：a[id\*='id\_pattern']

## **元素定位准则**

在使用各种方式进行元素定位的时候，应该怎么进行最佳选择呢？

1. 当页面元素有id属性时，最好尽量用id来定位。但由于现实项目中很多程序员其实写的代码并不规范，会缺少很多标准属性，这时就只有选择其他定位方法。

2. xpath很强悍，但定位性能不是很好，所以还是尽量少用。如果确实少数元素不好定位，可以选择xpath或cssSelector。

3. 当要定位一组相同元素时，可以考虑用tagName或name。

4. 当有链接需要定位时，可以考虑linkText或partialLinkText方式。

# **注解介绍**

框架基于注解获取页面元素，以web端为例进行说明。共有3种：@FindBy、@FindBys、@FindAll。下面对3种类型的区别和使用场景进行介绍。

1）@FindBy

**@FindBy**(id= "A")

**@CacheLookup**

**public WebElement** A**;**

@FindBy 获取指定条件的元素

2）@FindBys

**@Findbys**({

**@FindBy**(className = "A"),

**@FindBy**(name= "B")

})

**@CacheLookup**

**public WebElement** AB**;**

@FindBys 相当于是取交集，是先根据第一个注解获取到对应元素，然后根据第二个注解再帅选出对应的页面元素,。如先找到符合classname=A的元素，再在这些元素集中找到name=B的所有元素

3）@FindAll

**@FindAll**({

**@FindBy**(id = "A"),

**@FindBy**(id = "B")

})

**@CacheLookup**

**public List<WebElement>** aAndB**;**

@FindAll相当于是取并集，如找到id=A和id=B的所有元素