

软件测试开发职位内推Q群：485353510

(三) 命令和操作

这一部分将介绍一下 **WebDriver** 的一些具体操作和命令，实际操作中，我们需要两大工具来帮助我们：**FireBug** 和 **Xpath** 工具，这两者都是 **Firefox** 上的插件。接下来我们所讲解的都是以 **FirefoxDriver** 为基础的，且基于 **WebDriver driver = new FirefoxDriver();** 创建的一个 **driver** 实例：

a) 访问一个页面

第一件你想使用 **WebDriver** 做的事情肯定是访问一个页面，最基础的方法是调用“**get**”方法：

```
driver.get("http://www.google.com");
```

同样我们可以使用：

```
driver.navigate().to("http://www.google.com");
```

WebDriver 会自动等待到该页面完全加载才执行接下来的测试和脚本的执行。但是如果你的页面存在很多的 **AJAX** 加载，此时 **WebDriver** 是无法知道是否完成加载。检查此类页面是否加载完成，那么我们就需要 **Explicit** 和 **Implicit Wait**（这两个将在后面文章解释）。

b) 定位 UI 元素

WebDriver 可以通过 **WebDriver** 实例来定位元素，任何语言库都含有“**Find Element**”和“**Find Elements**”的方法。第一个方法返回一个 **WebElement** 或者抛出异常。后者返回

软件测试开发职位内推Q群：485353510

所有 `WebElement` 的列表，或者空列表。

获取和定位元素我们调用“By”方法。下面具体解释下“By”方法：

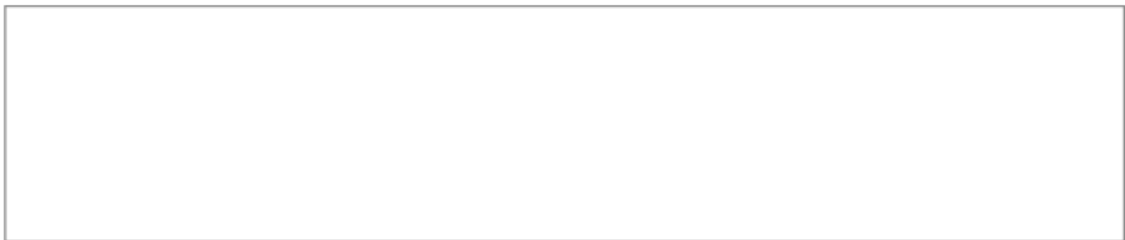
By ID

这是一个极为有效定位元素的方法。普遍的现状是 UI 工程师在实际编写页面时很少写 `id` 或者自动生产一个 ID，这些都是需要避免的。对于一个页面 `Element` 来说，`class` 比自动生产的 `id` 更好。

通过 `id` 定位元素的例子：

```
<div id="coolestWidgetEvah">...</div>
```

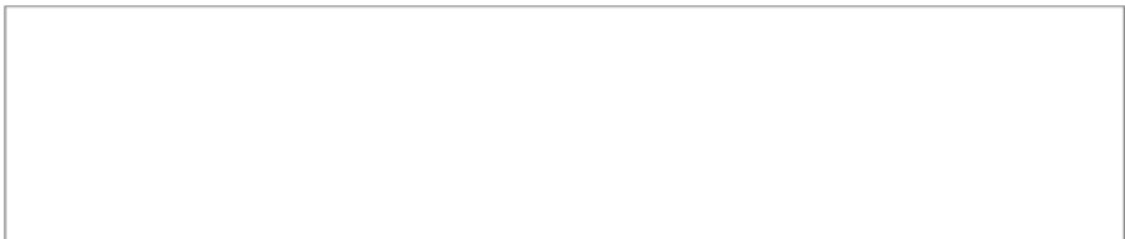
```
WebElement element = driver.findElement(By.id("coolestWidgetEvah"));
```



By Class Name

这里的 `class` 指的是 DOM 中的元素，在实际使用过程中，我们也会发现很多 DOM 元素含有相同的 `class` 名。

通过 `class name` 定位元素例子：



软件测试开发职位内推Q群：485353510

```
<div class="cheese">
```

```
<span>Cheddar</span>
```

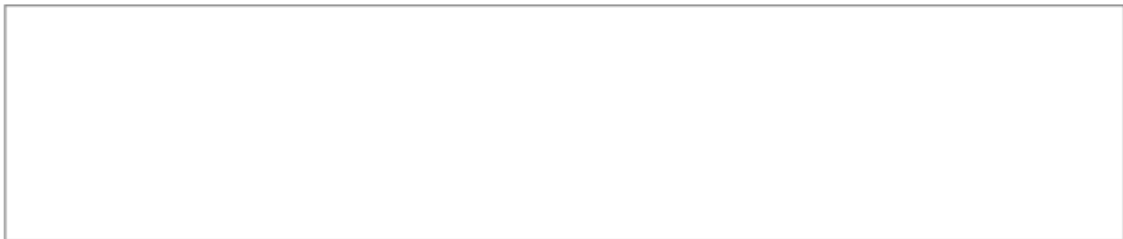
```
</div>
```

```
<div class="cheese">
```

```
<span>Gouda</span>
```

```
</div>
```

```
List<WebElement> cheeses = driver.findElements(By.className("cheese"));
```



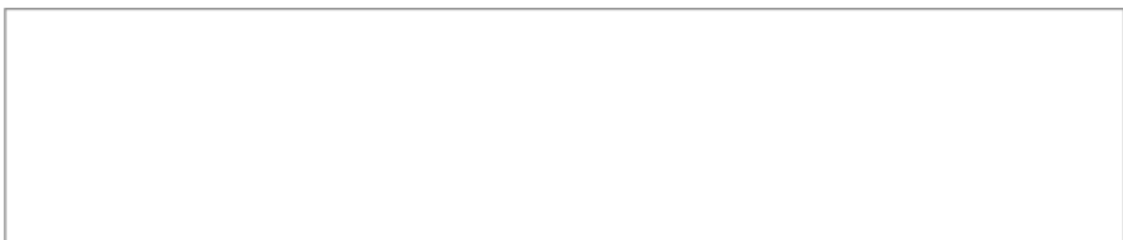
By Tag Name

DOM 的 Tag 元素

用 Tag name 定位元素的例子：

```
<iframe src="..."></iframe>
```

```
WebElement frame = driver.findElement(By.tagName("iframe"));
```



By Name

例子：

```
<input name="cheese" type="text"/>
```

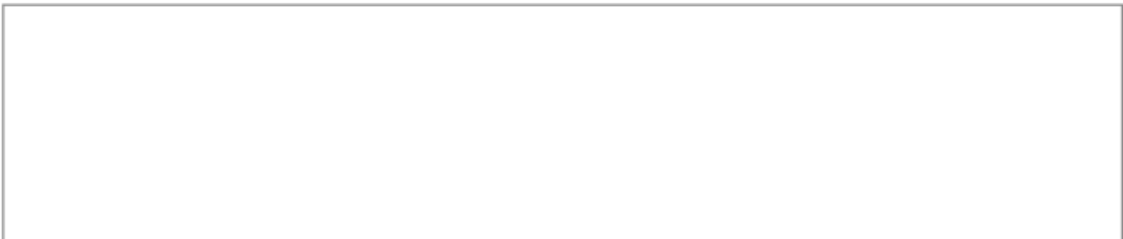
软件测试开发职位内推Q群：485353510

```
WebElement cheese = driver.findElement(By.name("cheese"));
```



By Link Text

例子：



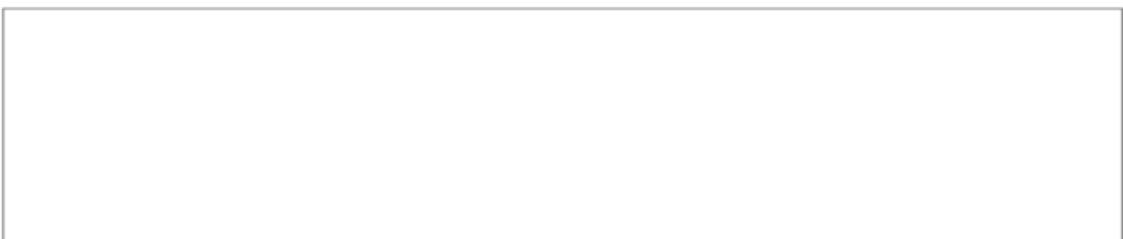
```
<a href="http://www.google.com/search?q=cheese">cheese</a>>
```

```
WebElement cheese = driver.findElement(By.linkText("cheese"));
```

By Partial Link Text

根据链接的部分文字

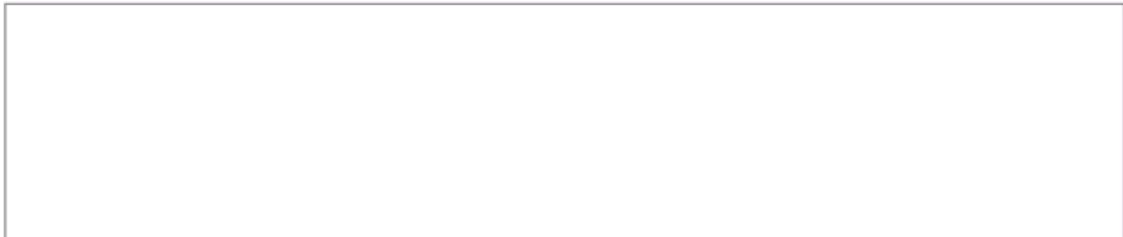
例子：



软件测试开发职位内推Q群：485353510

```
<a href="http://www.google.com/search?q=cheese">search for cheese</a>>
```

```
WebElement cheese = driver.findElement(By.partialLinkText("cheese"));
```



By CSS

从名字上看，这是根据 CSS 来定位元素。

例子：

```
<div id="food">
```

```
    <span class="dairy">milk</span>
```

```
    <span class="dairy aged">cheese</span>
```

```
</div>
```

```
WebElement cheese = driver.findElement(By.cssSelector("#food span.dairy aged"));
```

By XPATH

在高级的水平下，WebDriver 尽可能使用浏览器的原生的 XPath 能力。在那些没有原生的 XPath 支持的浏览器，我们提供自己的实现方式。但是三个 Driver 有一定的区别。

Driver	Tag and Attribute Name	Attribute Values	Native XPath Support
HtmlUnit Driver	Lower-cased	As they appear in the HTML	Yes
Internet Explorer Driver	Lower-cased	As they appear in the HTML	No
Firefox Driver	Case insensitive	As they appear in the HTML	Yes

例子：

软件测试开发职位内推Q群：485353510

```
<input type="text" name="example" />
```

```
<INPUT type="text" name="other" />
```

```
List<WebElement> inputs = driver.findElements(By.xpath("//input"));
```

查找结果：

HTML 元素有时并不需明确声明，因为他们将默认为已知值的属性。例如，input 标签，就不需要设置 type 为 text，默认属性就是 text，经验原则：WebDriver 在使用中的 XPath 时，不应该期望能够对这些隐含属性相匹配。

XPath expression	HtmlUnit Driver	Firefox Driver	Internet Explorer Driver
//input	1 ("example")	2	2
//INPUT	0	2	0

使用 javascript

您可以执行任意 JavaScript 找到一个元素，只要你返回一个 DOM 元素，它会自动转换到一个 WebElement 对象。

例子：

jQuery 的页面加载一个简单的例子：

```
WebElement element= (WebElement)
```

软件测试开发职位内推Q群：485353510

```
((JavascriptExecutor)driver).executeScript("return $('cheese')[0]");
```

寻求所有的页面上的 input 元素：

```
List<WebElement> labels= driver.findElements(By.tagName("label"));
```

```
List<WebElement> inputs= (List<WebElement>)
```

```
((JavascriptExecutor)driver).executeScript(
```

```
"var labels = arguments[0], inputs = []; for (var i=0; i < labels.length; i++){ "
```

```
"inputs.push(document.getElementById(labels[i].getAttribute('for'))); } return inputs; ",
```

```
labels);
```

用户表单填充

例子：

遍历 select 标签

```
WebElement select = driver.findElement(By.tagName("select"));
```

```
List<WebElement> allOptions= select.findElements(By.tagName("option"));
```

```
for (WebElement option: allOptions) {
```

软件测试开发职位内推Q群：485353510

```
System.out.println(String.format("Value is: %s", option.getAttribute("value")));
```

```
option.click();
```

```
}
```

选择某一个选项:

```
Select select = new Select(driver.findElement(By.tagName("select")));
```

```
select.deselectAll();
```

```
select.selectByVisibleText("Edam");
```

上传文件:

```
WebElement FileUpload =driver.findElement(By.id("upload"));
```


软件测试开发职位内推Q群：485353510

```
String filePath= "C:\\test\\uploadfile\\media_ads\\test.jpg";
```

```
FileUpload.sendKeys(filePath);
```

提交:

Submit 在 form 中

```
driver.findElement(By.id("submit")).click();
```

submit 不在 form 中

```
WebElement.submit();
```

拖拽操作:

```
WebElement element= driver.findElement(By.name("source"));
```

```
WebElement target= driver.findElement(By.name("target"));
```

软件测试开发职位内推Q群：485353510

```
(new Actions(driver)).dragAndDrop(element, target).perform();
```

Windows 和 Frames 之间的切换

一些 web 应用程序有许多 Frames 或多个 Windows。WebDriver 支持使用“switchTo”的方法实现的窗口之间切换。

```
driver.switchTo().window("windowName");
```

所有对 **driver** 的调用都会指向特定的窗口，但是我们怎么知道窗口的名字呢？我们可以查看 javascript 代码和打开他的链接：

```
<a href="somewhere.html" target="windowName">Click here to open a new window</a>
```

另外，还可以通过“window handle”去调用“switchTo().window()”，通过这个，我们就遍历来找到所有打开的窗口：

```
for (String handle: driver.getWindowHandles()) {  
  
    driver.switchTo().window(handle);  
  
}
```

软件测试开发职位内推Q群：485353510

Switch 同样支持 frame:

```
driver.switchTo().frame("frameName");
```

同样可以使用他访问 **subframe**，找 **frameName** 的第一个 **subframe** 中叫做 **child** 的 **frame**:

```
driver.switchTo().frame("frameName.0.child");
```

弹出框:

从 **selenium2.0**开始，已经支持对弹出框的获取

```
Alert alert= driver.switchTo().alert();
```

这个方法会返回当前被打开打警告框，你可以进行统一，取消，读取提示内容，后则进入到提示，这个同样使用 **alerts**, **confirms**, **prompts**。

Navigation: History and Location

之前我们就可以通过 **get** 方法来打开一个网页，像我们所看到的，**WebDriver** 同样还有许多小接口，**Navigation** 就是其中一个小接口：

软件测试开发职位内推Q群：485353510

```
driver.navigate().to("http://www.example.com");
```

`navigate().to` 和 `get()` 其实作用是一样的，但是 `navigate` 还可以进行浏览器的前进后退操作：

```
driver.navigate().forward();
```

```
driver.navigate().back();
```