目录

[一、启动浏览器 4](#_Toc383097679)

[1.1、IE浏览器 4](#_Toc383097680)

[1.2、火狐浏览器 6](#_Toc383097681)

[1.3、Chrome浏览器 7](#_Toc383097682)

[二、 关闭浏览器 7](#_Toc383097683)

[三、 浏览器简单操作 7](#_Toc383097684)

[3.1、打开页面 7](#_Toc383097685)

[3.2、浏览器最大化 8](#_Toc383097686)

[3.3、指定浏览器大小 8](#_Toc383097687)

[3.4、输出driver相关信息 8](#_Toc383097688)

[四、 对象定位 8](#_Toc383097689)

[4.1、单个对象定位 9](#_Toc383097690)

[4.1.1、ClassName定位 9](#_Toc383097691)

[4.1.2、CssSelector定位 9](#_Toc383097692)

[4.1.3、Id定位 10](#_Toc383097693)

[4.1.4、LinkText定位 10](#_Toc383097694)

[4.1.5、Name定位 10](#_Toc383097695)

[4.1.6、PartialLinkText定位 10](#_Toc383097696)

[4.1.7、TagName定位 10](#_Toc383097697)

[4.1.8、XPath定位 11](#_Toc383097698)

[4.2、多个对象定位 11](#_Toc383097699)

[4.3、层级定位 11](#_Toc383097700)

[操作测试对象 11](#_Toc383097701)

[场景 11](#_Toc383097702)

[代码 12](#_Toc383097703)

[*operate\_element.html* 12](#_Toc383097704)

[*operate\_element.java* 14](#_Toc383097705)

[send keys模拟按键输入 15](#_Toc383097706)

[场景 15](#_Toc383097707)

[代码 15](#_Toc383097708)

[*send\_keys.html* 15](#_Toc383097709)

[*send\_keys.java* 16](#_Toc383097710)

[处理button group 18](#_Toc383097711)

[场景 18](#_Toc383097712)

[代码 18](#_Toc383097713)

[*button\_group.html* 18](#_Toc383097714)

[*button\_group.java* 19](#_Toc383097715)

[讨论 21](#_Toc383097716)

[处理button dropdown 21](#_Toc383097717)

[场景 21](#_Toc383097718)

[代码 21](#_Toc383097719)

[*button\_dropdown.html* 21](#_Toc383097720)

[*button\_dropdown.java* 22](#_Toc383097721)

[处理navs 24](#_Toc383097722)

[场景 24](#_Toc383097723)

[代码 24](#_Toc383097724)

[*navs.html* 24](#_Toc383097725)

[*navs.java* 25](#_Toc383097726)

[处理面包屑 27](#_Toc383097727)

[场景 27](#_Toc383097728)

[代码 27](#_Toc383097729)

[*breadcrumb.html* 27](#_Toc383097730)

[*breadcrumb.java* 28](#_Toc383097731)

[处理对话框 29](#_Toc383097732)

[场景 30](#_Toc383097733)

[代码 30](#_Toc383097734)

[*modal.html* 30](#_Toc383097735)

[*modal.java* 32](#_Toc383097736)

[获取测试对象的属性及内容 33](#_Toc383097737)

[场景 33](#_Toc383097738)

[代码 34](#_Toc383097739)

[*attribute.html* 34](#_Toc383097740)

[*attribute.java* 35](#_Toc383097741)

[获取测试对象的css属性 36](#_Toc383097742)

[场景 36](#_Toc383097743)

[代码 36](#_Toc383097744)

[*css.html* 36](#_Toc383097745)

[*css.java* 38](#_Toc383097746)

[获取测试对象的状态 38](#_Toc383097747)

[场景 39](#_Toc383097748)

[*代码* 39](#_Toc383097749)

[*status.html* 39](#_Toc383097750)

[*status.java* 40](#_Toc383097751)

[讨论 42](#_Toc383097752)

[form的操作 42](#_Toc383097753)

[场景 42](#_Toc383097754)

[代码 42](#_Toc383097755)

[*form.html* 42](#_Toc383097756)

[*form.java* 44](#_Toc383097757)

[执行js 46](#_Toc383097758)

[场景 46](#_Toc383097759)

[代码 46](#_Toc383097760)

[*js.html* 46](#_Toc383097761)

[*js.java* 47](#_Toc383097762)

[处理alert/confirm/prompt 48](#_Toc383097763)

[场景 49](#_Toc383097764)

[代码 49](#_Toc383097765)

[*alert.html* 49](#_Toc383097766)

[*alert.java* 50](#_Toc383097767)

[wait 51](#_Toc383097768)

[场景 51](#_Toc383097769)

[代码 52](#_Toc383097770)

[*wait.html* 52](#_Toc383097771)

[*wait.java* 53](#_Toc383097772)

[定位frame中的元素 55](#_Toc383097773)

[场景 55](#_Toc383097774)

[代码 55](#_Toc383097775)

[*frame.html* 55](#_Toc383097776)

[*inner.html* 56](#_Toc383097777)

[*frame.java* 57](#_Toc383097778)

[讨论 58](#_Toc383097779)

[action 59](#_Toc383097780)

[场景 59](#_Toc383097781)

[代码 59](#_Toc383097782)

[讨论 60](#_Toc383097783)

[上传文件 60](#_Toc383097784)

[场景 60](#_Toc383097785)

[代码 60](#_Toc383097786)

[*upload\_file.html* 60](#_Toc383097787)

[*upload\_file.java* 61](#_Toc383097788)

[下载 62](#_Toc383097789)

[场景 62](#_Toc383097790)

[代码 62](#_Toc383097791)

[超时设置 63](#_Toc383097792)

[场景 63](#_Toc383097793)

[代码 63](#_Toc383097794)

[Remote Webdriver 64](#_Toc383097795)

[场景 64](#_Toc383097796)

[安装 64](#_Toc383097797)

[启动driver 64](#_Toc383097798)

[使用watir-webdriver启动driver 65](#_Toc383097799)

[java版本 65](#_Toc383097800)

[python版本 66](#_Toc383097801)

[cookie 66](#_Toc383097802)

[场景 66](#_Toc383097803)

[代码 66](#_Toc383097804)

[*cookie.java* 66](#_Toc383097805)

# 一、启动浏览器

### 1.1、IE浏览器

在启动IE浏览器前，需要对浏览器进行简单的设置。设置如下：

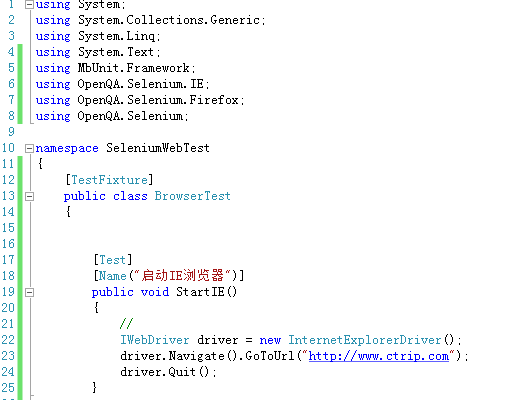
1、缩放设置为100%；

2、Internet选项->安全->四个区域（Internet、本地Internet、受信任的站点、受限制的站点）的“启用保护模式”设置一致（要么全部启动，要么全部关闭）



3、将IEDriverServer.exe放置在程序运行的当前目录，或者加到环境变量里，下载地址：http://docs.seleniumhq.org/download/

启动的代码如下：



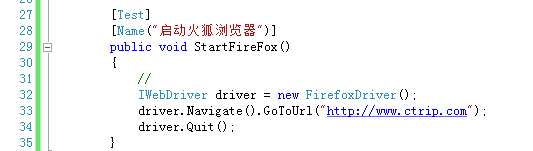
不进行上述两步设置，通过InternetExplorerDriver 的其它重载也可以启动，这将在后续进阶里面讲

### 1.2、火狐浏览器

火狐浏览器版本升级较快，而selenium一般是不能支持最新版本的，所以推荐安装稳定版或者把自动升级关掉，免得因自动升级到最新导致用例运行失败。

稳定版下载：<http://www.mozilla.org/en-US/firefox/organizations/all.html>

启动火狐浏览器的示例代码如下：



### 1.3、Chrome浏览器

启动chrome浏览器前需要把chromedriver.exe放置在程序运行的当前目录，或者加到环境变量里。

下载地址：<http://chromedriver.storage.googleapis.com/index.html>

启动示例代码如下：



# 关闭浏览器

用例执行结束需要关闭浏览器，selenium webdriver关闭浏览器有两种方式：

1、close方法关闭当前的浏览器窗口；

2、quit方法不仅关闭窗口，还会彻底的退出webdriver，释放与driver server之间的连接。所以简单来说quit是更加彻底的close，quit会更好的释放资源。如上述例子，我用的就是Quit。

Selenium Webdriver操作页面时，是不区分浏览器类型的（只有极少部分是需要区分的），所以下面的例子我就不分具体的浏览器了。

# 浏览器简单操作

### 3.1、打开页面

通过driver.Navigate().GoToUrl()打开指定url的页面。

INavigation对象的其他方法：

1、Back():回退；

2、Forward():前进；

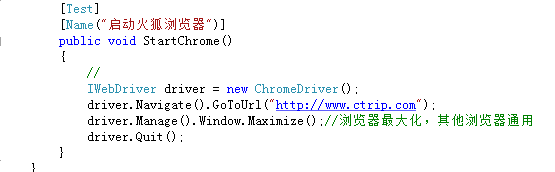
3、GoToUrl(string url)：打开链接；

4、GoToUrl(Uri url) ：打开链接；

5、Refresh():刷新页面。

### 3.2、浏览器最大化

用上面打开浏览器的例子打开的浏览器可能不是最大化，影响测试，如果要是浏览器最大化，用下面代码即可：



### 3.3、指定浏览器大小

某些特殊情况下，可能需要指定浏览器大小，可以使用如下代码：



### 3.4、输出driver相关信息



# 对象定位

Selenium Webdriver提供了强大的元素定位功能，支持以下三种方式：

1、单个对象定位；

2、多个对象定位；

3、层级定位；

在定位单个元素时Selenium Webdriver 提供了如下方法对元素进行定位：

1、By.ClassName

2、By.CssSelector()

3、By.Id()

4、By.LinkText()

5、By.Name()

6、By.PartialLinkText()

7、By.TagName()

8、By.XPath()

注意：Selenium-Webdriver通过FindElement()和FindElements()等方法调用“By”对象来定位和查询元素。By类只是提供查询元素的方式。FindElement()返回一个元素，FindElements()返回符合条件的元素List，如果不存在就返回一个空的list。

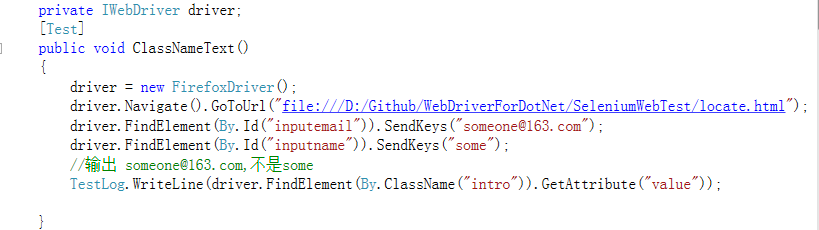
Css教程: <http://www.w3school.com.cn/css/>

### 4.1、单个对象定位

单个对象定位采用FindElement()方法定位，如果有多个元素符合定位要求，则默认取第一个。

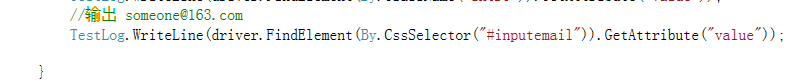
#### 4.1.1、ClassName定位

使用元素的ClassName定位要注意ClassName中间不可以带有空格（其实带有空格的那种是两个不同的CSS的组合），由于Class在页面中可以被使用多次，所以定位时如果定位到多个，则默认第一个！



#### 4.1.2、CssSelector定位

CSS选择器使用规则：<http://www.w3school.com.cn/css/css_syntax_id_selector.asp>



#### 4.1.3、Id定位



#### 4.1.4、LinkText定位

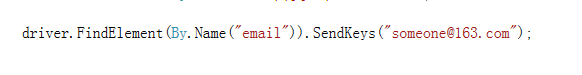
LinkText可以通过连接的名称进行定位，如果定位到多个，默认操作

第一个。



#### 4.1.5、Name定位

Name可以通过元素的name属性的值进行定位。



#### 4.1.6、PartialLinkText定位

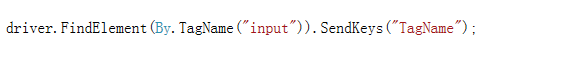
PartialLinkText可以用链接的部分名称进行定位，如果定位到多个，默认操作

第一个。



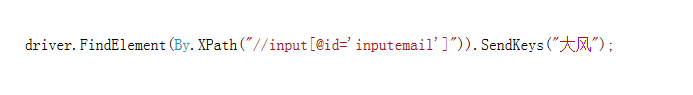
#### 4.1.7、TagName定位

TagName可以通过元素的标签名定位标签，如果是单个元素定位，同时有多个元素满足条件，则默认第一个。



#### 4.1.8、XPath定位

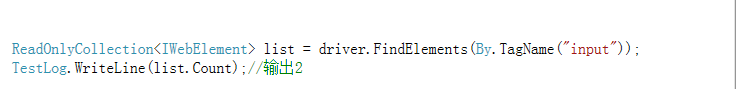
XPath使用规则：<http://www.w3school.com.cn/xpath/>



### 4.2、多个对象定位

多个元素定位采用FindElements()方法，返回一个ReadOnlyCollection<IWebElement>

对象（只读列表），定位方式与单个对象定位相同



### 4.3、层级定位

当页面比较复杂的时候，可以通过层级的方式定位,或者某些情况下，比如单独用Class定位的时候，可能定位到的不是需要的（Class在页面中可以存在多个相同的）



# 五、元素操作

定位到具体的对象后，我们就可以对这个对象进行具体的操作，比如先前已经看到过的点击操作(click)。一般来说，webdriver中比较常用的操作对象的方法有下面几个

### 5.1、常见操作

click 点击对象

sendKeys 在对象上模拟按键输入

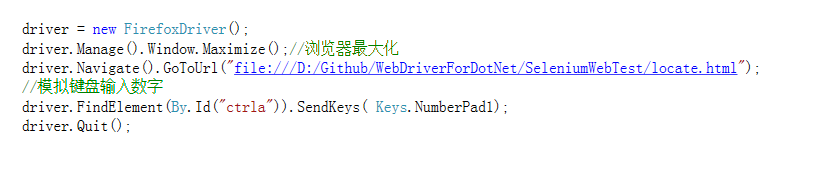
clear 清除对象的内容(如果输入框有默认值，在输入前最好先clear，以免输入内容会接在默认值后面)

下面代码演示了如何点击元素，如何清空文本框内容，如何给文本框输入内容



### 5.2、模拟键盘输入

Keys类配合SendKeys可以模拟多中键盘输入。



5.3、

# 处理button group

## 场景

button group就是按钮组，将一组按钮排列在一起。处理这种对象的思路一般是先找到button group的包裹(wrapper)div，然后通过层级定位，用index或属性去定位更具体的按钮。

## 代码

下面的代码演示了如何找到second这个按钮。其处理方法是先找到button group的父div，class为btn-group的div，然后再找到下面所有的div(也就是button)，返回text是second的div。

### button\_group.html

**<**html**>**

**<**head**>**

**<**metahttp**-**equiv**=**"content-type"content**=**"text/html;charset=utf-8"**/>**

**<**title**>**buttongroup**</**title**>**

**<**scripttype**=**"text/javascript"async**=**""src**=**"http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"**></**script**>**

**<**linkhref**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css"rel**=**"stylesheet"**/>**

**<**scripttype**=**"text/javascript"**>**

$**(**document**).**ready**(**function**(){**

$**(**'**.**btn'**).**click**(**function**(){**

alert**(**$**(this).**text**());**

**});**

**});**

**</**script**>**

**</**head**>**

**<**body**>**

**<**h3**>**buttongroup**</**h3**>**

**<**divclass**=**"row-fluid"**>**

**<**divclass**=**"span3"**>**

**<**divclass**=**"well"**>**

**<**divclass**=**"btn-toolbar"**>**

**<**divclass**=**"btn-group"**>**

**<**divclass**=**"btn"**>**first**</**div**>**

**<**divclass**=**"btn"**>**second**</**div**>**

**<**divclass**=**"btn"**>**third**</**div**>**

**</**div**>**

**</**div**>**

**</**div**>**

**</**div**>**

**</**div**>**

**</**body**>**

**<**scriptsrc**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/js/bootstrap.min.js"**></**script**>**

**</**html**>**

### button\_group.java

**import**java.io.File**;**

**import**java.util.List**;**

**import**org.openqa.selenium.By**;**

**import**org.openqa.selenium.Keys**;**

**import**org.openqa.selenium.WebDriver**;**

**import**org.openqa.selenium.WebElement**;**

**import**org.openqa.selenium.chrome.ChromeDriver**;**

**publicclassButtonGroup{**

**publicstaticvoidmain(**String**[]**args**)throws**InterruptedException**{**

WebDriverdr**=new**ChromeDriver**();**

Filefile**=new**File**(**"src/button\_group.html"**);**

StringfilePath**=**"file:///"**+**file**.**getAbsolutePath**();**

System**.**out**.**printf**(**"now accesss %s \n"**,**filePath**);**

dr**.**get**(**filePath**);**

Thread**.**sleep**(**1000**);**

*// 定位text是second的按钮*

List**<**WebElement**>**btns**=**dr**.**findElement**(**By**.**className**(**"btn-group"**)).**findElements**(**By**.**className**(**"btn"**));**

**for(**WebElementbtn**:**btns**){**

**if(**btn**.**getText**().**equals**(**"second"**)){**

btn**.**click**();**

**break;**

**}**

**}**

Thread**.**sleep**(**1000**);**

System**.**out**.**println**(**"browser will be close"**);**

dr**.**quit**();**

**}**

**}**

## 讨论

自己查资料搞清楚detect方法的作用。

# 处理button dropdown

## 场景

button dropdown就是把按钮和下拉菜单弄到了一起。处理这种对象的思路一般是先点击这个按钮，等待下拉菜单显示出来，然后使用层级定位方法来获取下拉菜单中的具体项。

## 代码

下面的代码演示了如何找到watir-webdriver这个菜单项。其处理方法是先点击info按钮，然后等到下拉菜单出现后定位下拉菜单的ul元素，再定位ul元素中link text为watir-webdriver的link，并点击之。

### button\_dropdown.html

**<**html**>**

**<**head**>**

**<**metahttp**-**equiv**=**"content-type"content**=**"text/html;charset=utf-8"**/>**

**<**title**>**buttongroup**</**title**>**

**<**scripttype**=**"text/javascript"async**=**""src**=**"http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"**></**script**>**

**<**linkhref**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css"rel**=**"stylesheet"**/>**

**<**scripttype**=**"text/javascript"**>**

$**(**document**).**ready**(**function**(){**

$**(**'**.**btn'**).**click**(**function**(){**

alert**(**$**(this).**text**());**

**});**

**});**

**</**script**>**

**</**head**>**

**<**body**>**

**<**h3**>**buttongroup**</**h3**>**

**<**divclass**=**"row-fluid"**>**

**<**divclass**=**"span3"**>**

**<**divclass**=**"well"**>**

**<**divclass**=**"btn-toolbar"**>**

**<**divclass**=**"btn-group"**>**

**<**divclass**=**"btn"**>**first**</**div**>**

**<**divclass**=**"btn"**>**second**</**div**>**

**<**divclass**=**"btn"**>**third**</**div**>**

**</**div**>**

**</**div**>**

**</**div**>**

**</**div**>**

**</**div**>**

**</**body**>**

**<**scriptsrc**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/js/bootstrap.min.js"**></**script**>**

**</**html**>**

### button\_dropdown.java

**import**java.io.File**;**

**import**org.openqa.selenium.By**;**

**import**org.openqa.selenium.WebDriver**;**

**import**org.openqa.selenium.chrome.ChromeDriver**;**

**import**org.openqa.selenium.support.ui.ExpectedCondition**;**

**import**org.openqa.selenium.support.ui.WebDriverWait**;**

**publicclassButtonDropdown{**

**publicstaticvoidmain(**String**[]**args**)throws**InterruptedException**{**

WebDriverdr**=new**ChromeDriver**();**

Filefile**=new**File**(**"src/button\_dropdown.html"**);**

StringfilePath**=**"file:///"**+**file**.**getAbsolutePath**();**

System**.**out**.**printf**(**"now accesss %s \n"**,**filePath**);**

dr**.**get**(**filePath**);**

Thread**.**sleep**(**1000**);**

*// 定位text是watir-webdriver的下拉菜单*

*// 首先显示下拉菜单*

dr**.**findElement**(**By**.**linkText**(**"Info"**)).**click**();**

**(new**WebDriverWait**(**dr**,**10**)).**until**(new**ExpectedCondition**<**Boolean**>(){**

**public**Boolean**apply(**WebDriverd**){**

**return**d**.**findElement**(**By**.**className**(**"dropdown-menu"**)).**isDisplayed**();**

**}**

**});**

*// 通过ul再层级定位*

dr**.**findElement**(**By**.**className**(**"dropdown-menu"**)).**findElement**(**By**.**linkText**(**"watir-webdriver"**)).**click**();**

Thread**.**sleep**(**1000**);**

System**.**out**.**println**(**"browser will be close"**);**

dr**.**quit**();**

**}**

**}**

# 处理navs

## 场景

navs可以看作是简单的类似于tab的导航栏。一般来说导航栏都是ul+li。先定位ul再去层级定位li中的link基本就能解决问题。

## 代码

### navs.html

**<**html**>**

**<**head**>**

**<**metahttp**-**equiv**=**"content-type"content**=**"text/html;charset=utf-8"**/>**

**<**title**>**Navs**</**title**>**

**<**scripttype**=**"text/javascript"async**=**""src**=**"http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"**></**script**>**

**<**linkhref**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css"rel**=**"stylesheet"**/>**

**<**scripttype**=**"text/javascript"**>**

$**(**document**).**ready**(**

function**(){**

$**(**'**.**nav'**).**find**(**'li'**).**click**(**function**(){**

$**(this).**parent**().**find**(**'li'**).**removeClass**(**'active'**);**

$**(this).**addClass**(**'active'**);**

**});**

**}**

**);**

**</**script**>**

**</**head**>**

**<**body**>**

**<**h3**>**Navs**</**h3**>**

**<**divclass**=**"row-fluid"**>**

**<**divclass**=**"span3"**>**

**<**ulclass**=**"nav nav-pills"**>**

**<**liclass**=**"active"**>**

**<**ahref**=**"#"**>**Home**</**a**>**

**</**li**>**

**<**li**><**ahref**=**"#"**>**Content**</**a**></**li**>**

**<**li**><**ahref**=**"#"**>**About**</**a**></**li**>**

**</**ul**>**

**</**div**>**

**</**div**>**

**</**body**>**

**<**scriptsrc**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/js/bootstrap.min.js"**></**script**>**

**</**html**>**

### navs.java

**import**java.io.File**;**

**import**java.util.List**;**

**import**org.openqa.selenium.By**;**

**import**org.openqa.selenium.Keys**;**

**import**org.openqa.selenium.WebDriver**;**

**import**org.openqa.selenium.WebElement**;**

**import**org.openqa.selenium.chrome.ChromeDriver**;**

**publicclassNavs{**

**publicstaticvoidmain(**String**[]**args**)throws**InterruptedException**{**

WebDriverdr**=new**ChromeDriver**();**

Filefile**=new**File**(**"src/navs.html"**);**

StringfilePath**=**"file:///"**+**file**.**getAbsolutePath**();**

System**.**out**.**printf**(**"now accesss %s \n"**,**filePath**);**

dr**.**get**(**filePath**);**

Thread**.**sleep**(**1000**);**

*// 方法1：层级定位，先定位ul再定位li*

dr**.**findElement**(**By**.**className**(**"nav"**)).**findElement**(**By**.**linkText**(**"About"**)).**click**();**

Thread**.**sleep**(**1000**);**

*// 方法2: 直接定位link*

dr**.**findElement**(**By**.**linkText**(**"Home"**)).**click**();**

Thread**.**sleep**(**1000**);**

System**.**out**.**println**(**"browser will be close"**);**

dr**.**quit**();**

**}**

**}**

# 处理面包屑

## 场景

在实际的测试脚本中，有可能需要处理面包屑。处理面包屑主要是获取其层级关系，以及获得当前的层级。一般来说当前层级都不会是链接，而父层级则基本是以链接，所以处理面包屑的思路就很明显了。找到面包屑所在的div或ul，然后再通过该div或ul找到下面的所有链接，这些链接就是父层级。最后不是链接的部分就应该是当前层级了。

## 代码

### breadcrumb.html

**<**html**>**

**<**head**>**

**<**metahttp**-**equiv**=**"content-type"content**=**"text/html;charset=utf-8"**/>**

**<**title**>**breadcrumb**</**title**>**

**<**scripttype**=**"text/javascript"async**=**""src**=**"http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"**></**script**>**

**<**linkhref**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css"rel**=**"stylesheet"**/>**

**<**scripttype**=**"text/javascript"**>**

$**(**document**).**ready**(**

function**(){**

**}**

**);**

**</**script**>**

**</**head**>**

**<**body**>**

**<**h3**>**breadcrumb**</**h3**>**

**<**divclass**=**"row-fluid"**>**

**<**divclass**=**"span3"**>**

**<**ulclass**=**"breadcrumb"**>**

**<**li**><**ahref**=**"#"**>**Home**</**a**><**spanclass**=**"divider"**>/</**span**></**li**>**

**<**li**><**ahref**=**"#"**>**Library**</**a**><**spanclass**=**"divider"**>/</**span**></**li**>**

**<**liclass**=**"active"**>**Data**</**li**>**

**</**ul**>**

**</**div**>**

**</**div**>**

**</**body**>**

**<**scriptsrc**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/js/bootstrap.min.js"**></**script**>**

**</**html**>**

### breadcrumb.java

**import**java.io.File**;**

**import**java.util.List**;**

**import**org.openqa.selenium.By**;**

**import**org.openqa.selenium.WebDriver**;**

**import**org.openqa.selenium.WebElement**;**

**import**org.openqa.selenium.chrome.ChromeDriver**;**

**publicclassBreadcrumb{**

**publicstaticvoidmain(**String**[]**args**)throws**InterruptedException**{**

WebDriverdr**=new**ChromeDriver**();**

Filefile**=new**File**(**"src/breadcrumb.html"**);**

StringfilePath**=**"file:///"**+**file**.**getAbsolutePath**();**

System**.**out**.**printf**(**"now accesss %s \n"**,**filePath**);**

dr**.**get**(**filePath**);**

Thread**.**sleep**(**1000**);**

*// 获得其父层级*

List**<**WebElement**>**ancestors**=**dr**.**findElement**(**By**.**className**(**"breadcrumb"**)).**findElements**(**By**.**tagName**(**"a"**));**

**for(**WebElementlink**:**ancestors**){**

System**.**out**.**println**(**link**.**getText**());**

**}**

*// 获取当前层级*

*// 由于页面上可能有很多class为active的元素*

*// 所以使用层级定位最为保险*

WebElementcurrent**=**dr**.**findElement**(**By**.**className**(**"breadcrumb"**)).**findElement**(**By**.**className**(**"active"**));**

System**.**out**.**println**(**current**.**getText**());**

Thread**.**sleep**(**1000**);**

System**.**out**.**println**(**"browser will be close"**);**

dr**.**quit**();**

**}**

**}**

# 处理对话框

## 场景

页面上弹出的对话框是自动化测试经常会遇到的一个问题。很多情况下这个弹出的对话框是一个iframe，处理起来有点麻烦，需要进行switch\_to操作。但现在很多前端框架的对话框都是div形式的，这就让我们的处理变得十分简单了。

处理对话框一般会做下面的一些事情

* 打开对话框
* 关闭对话框
* 操作对话框中的元素

## 代码

下面的代码演示了如何打开、关闭以及点击对话框中的链接

### modal.html

**<**html**>**

**<**head**>**

**<**metahttp**-**equiv**=**"content-type"content**=**"text/html;charset=utf-8"**/>**

**<**title**>**modal**</**title**>**

**<**scripttype**=**"text/javascript"async**=**""src**=**"http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"**></**script**>**

**<**linkhref**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css"rel**=**"stylesheet"**/>**

**<**scripttype**=**"text/javascript"**>**

$**(**document**).**ready**(**function**(){**

$**(**'#click'**).**click**(**function**(){**

$**(this).**parent**().**find**(**'p'**).**text**(**'**try**watir**-**webdriverrightnow**!**'**);**

**});**

**});**

**</**script**>**

**</**head**>**

**<**body**>**

**<**h3**>**modal**</**h3**>**

**<**divclass**=**"row-fluid"**>**

**<**divclass**=**"span6"**>**

**<!--**Buttontotriggermodal**-->**

**<**ahref**=**"#myModal"role**=**"button"class**=**"btn btn-primary"data**-**toggle**=**"modal"id**=**"show\_modal"**>**Click**</**a**>**

**<!--**Modal**-->**

**<**divid**=**"myModal"class**=**"modal hide fade"tabindex**=**"-1"role**=**"dialog"aria**-**labelledby**=**"myModalLabel"aria**-**hidden**=**"true"**>**

**<**divclass**=**"modal-header"**>**

**<**buttontype**=**"button"class**=**"close"data**-**dismiss**=**"modal"aria**-**hidden**=**"true"**>**×**</**button**>**

**<**h3id**=**"myModalLabel"**>**Modalheader**</**h3**>**

**</**div**>**

**<**divclass**=**"modal-body"**>**

**<**p**>**watir**-**webdriverisbetterthanslenium**-**webdriver**</**p**>**

**<**ahref**=**"#"id**=**"click"**>**clickme**</**a**>**

**</**div**>**

**<**divclass**=**"modal-footer"**>**

**<**buttonclass**=**"btn"data**-**dismiss**=**"modal"aria**-**hidden**=**"true"**>**Close**</**button**>**

**<**buttonclass**=**"btn btn-primary"**>**Savechanges**</**button**>**

**</**div**>**

**</**div**>**

**</**div**>**

**</**div**>**

**</**body**>**

**<**scriptsrc**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/js/bootstrap.min.js"**></**script**>**

**</**html**>**

### modal.java

**import**java.io.File**;**

**import**java.util.List**;**

**import**org.openqa.selenium.By**;**

**import**org.openqa.selenium.JavascriptExecutor**;**

**import**org.openqa.selenium.Keys**;**

**import**org.openqa.selenium.WebDriver**;**

**import**org.openqa.selenium.WebElement**;**

**import**org.openqa.selenium.chrome.ChromeDriver**;**

**import**org.openqa.selenium.support.ui.ExpectedCondition**;**

**import**org.openqa.selenium.support.ui.WebDriverWait**;**

**publicclassModal{**

**publicstaticvoidmain(**String**[]**args**)throws**InterruptedException**{**

WebDriverdr**=new**ChromeDriver**();**

Filefile**=new**File**(**"src/modal.html"**);**

StringfilePath**=**"file:///"**+**file**.**getAbsolutePath**();**

System**.**out**.**printf**(**"now accesss %s \n"**,**filePath**);**

dr**.**get**(**filePath**);**

Thread**.**sleep**(**1000**);**

*// 打开对话框*

dr**.**findElement**(**By**.**id**(**"show\_modal"**)).**click**();**

**(new**WebDriverWait**(**dr**,**10**)).**until**(**

**new**ExpectedCondition**<**Boolean**>(){**

**public**Boolean**apply(**WebDriverd**){**

**return**d**.**findElement**(**By**.**id**(**"myModal"**)).**isDisplayed**();**

**}**

**}**

**);**

*// 点击对话框中的链接*

*// 由于对话框中的元素被蒙板所遮挡，直接点击会报 Element is not clickable的错误*

*// 所以使用js来模拟click*

*// 在watir-webdriver中只需要fire\_event(:click)就可以了*

WebElementlink**=**dr**.**findElement**(**By**.**id**(**"myModal"**)).**findElement**(**By**.**id**(**"click"**));**

**((**JavascriptExecutor**)**dr**).**executeScript**(**"$(arguments[0]).click()"**,**link**);**

Thread**.**sleep**(**1000**);**

System**.**out**.**println**(**"browser will be close"**);**

dr**.**quit**();**

**}**

**}**

# 获取测试对象的属性及内容

## 场景

获取测试对象的内容是前端自动化测试里一定会使用到的技术。比如我们要判断页面上是否显示了一个提示，那么我们就需要找到这个提示对象，然后获取其中的文字，再跟我们的预期进行比较。在webdriver中使用element.getAttribute()方法可以获取dom元素(测试对象)的属性。

获取测试对象的属性能够帮我们更好的进行对象的定位。比如页面上有很多class都是'btn'的div，而我们需要定位其中1个有具有title属性的div。由于selenium-webdriver是不支持直接使用title来定位对象的，所以我们只能先把所有class是btn的div都找到，然后遍历这些div，获取这些div的title属性，一旦发现具体title属性的div，那么返回这个div既可。在webdriver中，使用element.getText()方法可以返回dom节点的内容(text)。

## 代码

下面的代码演示了如何获取测试对象的title属性和该对象的文字内容

### attribute.html

**<**html**>**

**<**head**>**

**<**metahttp**-**equiv**=**"content-type"content**=**"text/html;charset=utf-8"**/>**

**<**title**>**attribute**</**title**>**

**<**scripttype**=**"text/javascript"async**=**""src**=**"http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"**></**script**>**

**<**linkhref**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css"rel**=**"stylesheet"**/>**

**<**scripttype**=**"text/javascript"**>**

$**(**document**).**ready**(**function**(){**

$**(**'#tooltip'**).**tooltip**({**"placement"**:**"right"**});**

**});**

**</**script**>**

**</**head**>**

**<**body**>**

**<**h3**>**attribute**</**h3**>**

**<**divclass**=**"row-fluid"**>**

**<**divclass**=**"span6"**>**

**<**aid**=**"tooltip"href**=**"#"data**-**toggle**=**"tooltip"title**=**"watir-webdriver better than selenium-webdriver"**>**hovertoseetooltip**</**a**>**

**</**div**>**

**</**div**>**

**</**body**>**

**<**scriptsrc**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/js/bootstrap.min.js"**></**script**>**

**</**html**>**

### attribute.java

**import**java.io.File**;**

**import**java.util.List**;**

**import**org.openqa.selenium.By**;**

**import**org.openqa.selenium.WebDriver**;**

**import**org.openqa.selenium.WebElement**;**

**import**org.openqa.selenium.chrome.ChromeDriver**;**

**publicclassAttribute{**

**publicstaticvoidmain(**String**[]**args**)throws**InterruptedException**{**

WebDriverdr**=new**ChromeDriver**();**

Filefile**=new**File**(**"src/attribute.html"**);**

StringfilePath**=**"file:///"**+**file**.**getAbsolutePath**();**

System**.**out**.**printf**(**"now accesss %s \n"**,**filePath**);**

dr**.**get**(**filePath**);**

Thread**.**sleep**(**1000**);**

WebElementlink**=**dr**.**findElement**(**By**.**id**(**"tooltip"**));**

*// 获得tooltip的内容*

System**.**out**.**println**(**link**.**getAttribute**(**"data-original-title"**));**

*// 获取该链接的text*

System**.**out**.**println**(**link**.**getText**());**

Thread**.**sleep**(**1000**);**

System**.**out**.**println**(**"browser will be close"**);**

dr**.**quit**();**

**}**

**}**

# 获取测试对象的css属性

## 场景

当你的测试用例纠结细枝末节的时候，你就需要通过判断元素的css属性来验证你的操作是否达到了预期的效果。比如你可以通过判断页面上的标题字号以字体来验证页面的显示是否符合预期。当然，这个是强烈不推荐的。因为页面上最不稳定的就是css了，css变动频繁，而且通过属性也不能直观的判断页面的显示效果，还不如让人为的去看一眼，大问题一望即知。

## 代码

下面的代码演示了如何获取测试对象的css属性。

### css.html

**<**html**>**

**<**head**>**

**<**metahttp**-**equiv**=**"content-type"content**=**"text/html;charset=utf-8"**/>**

**<**title**>**attribute**</**title**>**

**<**scripttype**=**"text/javascript"async**=**""src**=**"http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"**></**script**>**

**<**linkhref**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css"rel**=**"stylesheet"**/>**

**<**scripttype**=**"text/javascript"**>**

$**(**document**).**ready**(**function**(){**

$**(**'#tooltip'**).**tooltip**({**"placement"**:**"right"**});**

**});**

**</**script**>**

**</**head**>**

**<**body**>**

**<**h3**>**attribute**</**h3**>**

**<**divclass**=**"row-fluid"**>**

**<**divclass**=**"span6"**>**

**<**aid**=**"tooltip"href**=**"#"data**-**toggle**=**"tooltip"title**=**"watir-webdriver better than selenium-webdriver"**>**hovertoseetooltip**</**a**>**

**</**div**>**

**</**div**>**

**</**body**>**

**<**scriptsrc**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/js/bootstrap.min.js"**></**script**>**

**</**html**>**

### css.java

**import**java.io.File**;**

**import**java.util.List**;**

**import**org.openqa.selenium.By**;**

**import**org.openqa.selenium.WebDriver**;**

**import**org.openqa.selenium.WebElement**;**

**import**org.openqa.selenium.chrome.ChromeDriver**;**

**publicclassCss{**

**publicstaticvoidmain(**String**[]**args**)throws**InterruptedException**{**

WebDriverdr**=new**ChromeDriver**();**

Filefile**=new**File**(**"src/css.html"**);**

StringfilePath**=**"file:///"**+**file**.**getAbsolutePath**();**

System**.**out**.**printf**(**"now accesss %s \n"**,**filePath**);**

dr**.**get**(**filePath**);**

Thread**.**sleep**(**1000**);**

WebElementlink**=**dr**.**findElement**(**By**.**id**(**"tooltip"**));**

System**.**out**.**println**(**link**.**getCssValue**(**"color"**));**

System**.**out**.**println**(**dr**.**findElement**(**By**.**tagName**(**"h3"**)).**getCssValue**(**"font"**));**

Thread**.**sleep**(**1000**);**

System**.**out**.**println**(**"browser will be close"**);**

dr**.**quit**();**

**}**

**}**

# 获取测试对象的状态

## 场景

在web自动化测试中，我们需要获取测试对象的四种状态

* 是否显示。使用element.isDisplayed()方法；
* 是否存在。使用findElement方法，捕获其抛出的异常，如果是NoSuchElementException的话则可以确定该元素不存在；
* 是否被选中。一般是判断表单元素，比如radio或checkbox是否被选中。使用element.isSelected()方法；
* 是否enable，也就是是否是灰化状态。使用element.isEnabled()方法；

## 代码

### status.html

**<**html**>**

**<**head**>**

**<**metahttp**-**equiv**=**"content-type"content**=**"text/html;charset=utf-8"**/>**

**<**title**>**status**</**title**>**

**<**scripttype**=**"text/javascript"async**=**""src**=**"http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"**></**script**>**

**<**linkhref**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css"rel**=**"stylesheet"**/>**

**<**scripttype**=**"text/javascript"**>**

$**(**document**).**ready**(**function**(){**

$**(**'#tooltip'**).**tooltip**({**"placement"**:**"right"**});**

**});**

**</**script**>**

**</**head**>**

**<**body**>**

**<**h3**>**status**</**h3**>**

**<**divclass**=**"row-fluid"**>**

**<**divclass**=**"span3"**>**

**<**inputname**=**"user"placeholder**=**"Disabled TextField"disabled**/>**

**</**div**>**

**<**divclass**=**"span3"**>**

**<**aclass**=**"btn disabled"**>**DisabledButton**</**a**>**

**</**div**>**

**<**divclass**=**"span3"**>**

**<**inputname**=**"radio"type**=**"radio"**/>**

**</**div**>**

**</**div**>**

**</**body**>**

**<**scriptsrc**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/js/bootstrap.min.js"**></**script**>**

**</**html**>**

### status.java

**import**java.io.File**;**

**import**java.util.List**;**

**import**org.openqa.selenium.By**;**

**import**org.openqa.selenium.JavascriptExecutor**;**

**import**org.openqa.selenium.NoSuchElementException**;**

**import**org.openqa.selenium.WebDriver**;**

**import**org.openqa.selenium.WebElement**;**

**import**org.openqa.selenium.chrome.ChromeDriver**;**

**publicclassStatus{**

**publicstaticvoidmain(**String**[]**args**)throws**InterruptedException**{**

WebDriverdr**=new**ChromeDriver**();**

Filefile**=new**File**(**"src/status.html"**);**

StringfilePath**=**"file:///"**+**file**.**getAbsolutePath**();**

System**.**out**.**printf**(**"now accesss %s \n"**,**filePath**);**

dr**.**get**(**filePath**);**

Thread**.**sleep**(**1000**);**

WebElementtextField**=**dr**.**findElement**(**By**.**name**(**"user"**));**

System**.**out**.**println**(**textField**.**isEnabled**());**

*// 直接用isEnabled方法去判断该button的话返回的会是true*

*// 这是因为button是使用css方法去disabled的，并不是真正的disable*

*// 这时候需要判断其class里是否有disabled这值来判断其是否处于disable状态*

System**.**out**.**println**(**dr**.**findElement**(**By**.**className**(**"btn"**)).**isEnabled**());**

*// 隐藏掉textField*

*// 判断其是否显示*

**((**JavascriptExecutor**)**dr**).**executeScript**(**"$(arguments[0]).hide()"**,**textField**);**

System**.**out**.**println**(**textField**.**isDisplayed**());**

*// 使用click方法选择raido*

WebElementradio**=**dr**.**findElement**(**By**.**name**(**"radio"**));**

radio**.**click**();**

System**.**out**.**println**(**radio**.**isSelected**());**

**try{**

dr**.**findElement**(**By**.**id**(**"none"**));**

**}catch(**NoSuchElementExceptione**){**

System**.**out**.**println**(**"element does not exist"**);**

**}**

Thread**.**sleep**(**1000**);**

System**.**out**.**println**(**"browser will be close"**);**

dr**.**quit**();**

**}**

**}**

## 讨论

在这里我们遇到了一种情况，那就是测试对象看上去是disabled，但是使用enabled方法却返回true。这时候一般思路是判断该对象的css属性或class，通过这些值去进一步判断对象是否disable。

# form的操作

## 场景

表单对象的操作比较简单，只需要记住下面几点

* 使用sendKeys方法往多行文本框和单行文本框赋值；
* 使用click方法选择checkbox
* 使用click方法选择radio
* 使用click方法点击button
* 使用click方法选择option，从而达到选中select下拉框中某个具体菜单项的效果

# 代码

## form.html

**<**html**>**

**<**head**>**

**<**metahttp**-**equiv**=**"content-type"content**=**"text/html;charset=utf-8"**/>**

**<**title**>**form**</**title**>**

**<**scripttype**=**"text/javascript"async**=**""src**=**"http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"**></**script**>**

**<**linkhref**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css"rel**=**"stylesheet"**/>**

**<**scripttype**=**"text/javascript"**>**

$**(**document**).**ready**(**function**(){**

$**(**'input**[**type**=**submit**]**'**).**click**(**function**(){**

alert**(**'watir**-**webdriverisbetterthanseleniumwebdriver'**);**

**});**

**});**

**</**script**>**

**</**head**>**

**<**body**>**

**<**h3**>**form**</**h3**>**

**<**divclass**=**"row-fluid"**>**

**<**divclass**=**"span6 well"**>**

**<**form**>**

**<**fieldset**>**

**<**legend**>**Legend**</**legend**>**

**<**labelclass**=**"checkbox"**>**

**<**inputtype**=**"checkbox"**>**Checkmeout

**</**label**>**

**<**labelclass**=**"radio"**>**

**<**inputtype**=**"radio"**>**selectme

**</**label**>**

**<**labelclass**=**"select"**>**

**<**select**>**

**<**option**>**0**</**option**>**

**<**option**>**1**</**option**>**

**<**option**>**2**</**option**>**

**</**select**>**selectoneitem

**</**label**>**

**<**inputtype**=**"submit"class**=**"btn"value**=**"submit"**/>**

**</**fieldset**>**

**</**form**>**

**</**div**>**

**</**div**>**

**</**body**>**

**<**scriptsrc**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/js/bootstrap.min.js"**></**script**>**

**</**html**>**

### form.java

**import**java.io.File**;**

**import**java.util.List**;**

**import**org.openqa.selenium.Alert**;**

**import**org.openqa.selenium.By**;**

**import**org.openqa.selenium.Keys**;**

**import**org.openqa.selenium.WebDriver**;**

**import**org.openqa.selenium.WebElement**;**

**import**org.openqa.selenium.chrome.ChromeDriver**;**

**publicclassForm{**

**publicstaticvoidmain(**String**[]**args**)throws**InterruptedException**{**

WebDriverdr**=new**ChromeDriver**();**

Filefile**=new**File**(**"src/form.html"**);**

StringfilePath**=**"file:///"**+**file**.**getAbsolutePath**();**

System**.**out**.**printf**(**"now accesss %s \n"**,**filePath**);**

dr**.**get**(**filePath**);**

Thread**.**sleep**(**1000**);**

*// 选中checkbox*

dr**.**findElement**(**By**.**cssSelector**(**"input[type=checkbox]"**)).**click**();**

Thread**.**sleep**(**1000**);**

*// 选中radio*

dr**.**findElement**(**By**.**cssSelector**(**"input[type=radio]"**)).**click**();**

Thread**.**sleep**(**1000**);**

*// 选择下拉菜单中的最后一项*

List**<**WebElement**>**options**=**dr**.**findElement**(**By**.**tagName**(**"select"**)).**findElements**(**By**.**tagName**(**"option"**));**

options**.**get**(**options**.**size**()-**1**).**click**();**

Thread**.**sleep**(**1000**);**

*// 点击提交按钮*

dr**.**findElement**(**By**.**cssSelector**(**"input[type=submit]"**)).**click**();**

Alertalert**=**dr**.**switchTo**().**alert**();**

System**.**out**.**println**(**alert**.**getText**());**

alert**.**accept**();**

Thread**.**sleep**(**1000**);**

System**.**out**.**println**(**"browser will be close"**);**

dr**.**quit**();**

**}**

**}**

# 执行js

## 场景

如果你熟悉js的话，那么使用webdriver执行js就是一件很高效的事情了。在webdriver脚本中直接执行js的好处很多，这里就不一一枚举了。

webdriver提供了JavascriptExecutor(dr).executeScript()接口来帮助我们完成这一工作。在实际的测试脚本中，以下两种场景是经常遇到的

* 在页面直接执行一段js
* 在某个已经定位的元素的上执行js

## 代码

下面的代码演示了如何在页面以及在已经定位的元素上执行js

### js.html

**<**html**>**

**<**head**>**

**<**metahttp**-**equiv**=**"content-type"content**=**"text/html;charset=utf-8"**/>**

**<**title**>**js**</**title**>**

**<**scripttype**=**"text/javascript"async**=**""src**=**"http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"**></**script**>**

**<**linkhref**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css"rel**=**"stylesheet"**/>**

**<**scripttype**=**"text/javascript"**>**

$**(**document**).**ready**(**function**(){**

$**(**'#tooltip'**).**tooltip**({**"placement"**:**"right"**});**

**});**

**</**script**>**

**</**head**>**

**<**body**>**

**<**h3**>**js**</**h3**>**

**<**divclass**=**"row-fluid"**>**

**<**divclass**=**"span6 well"**>**

**<**aid**=**"tooltip"href**=**"#"data**-**toggle**=**"tooltip"title**=**"watir-webdriver better than selenium-webdriver"**>**hovertoseetooltip**</**a**>**

**<**aclass**=**"btn"**>**Button**</**a**>**

**</**div**>**

**</**div**>**

**</**body**>**

**<**scriptsrc**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/js/bootstrap.min.js"**></**script**>**

**</**html**>**

### js.java

**import**java.io.File**;**

**import**java.util.List**;**

**import**org.openqa.selenium.Alert**;**

**import**org.openqa.selenium.By**;**

**import**org.openqa.selenium.JavascriptExecutor**;**

**import**org.openqa.selenium.Keys**;**

**import**org.openqa.selenium.WebDriver**;**

**import**org.openqa.selenium.WebElement**;**

**import**org.openqa.selenium.chrome.ChromeDriver**;**

**publicclassJs{**

**publicstaticvoidmain(**String**[]**args**)throws**InterruptedException**{**

WebDriverdr**=new**ChromeDriver**();**

Filefile**=new**File**(**"src/js.html"**);**

StringfilePath**=**"file:///"**+**file**.**getAbsolutePath**();**

System**.**out**.**printf**(**"now accesss %s \n"**,**filePath**);**

dr**.**get**(**filePath**);**

Thread**.**sleep**(**1000**);**

*// 在页面上直接执行js*

**((**JavascriptExecutor**)**dr**).**executeScript**(**"$('#tooltip').fadeOut();"**);**

Thread**.**sleep**(**1000**);**

*// 在已经定位的元素上执行js*

WebElementbutton**=**dr**.**findElement**(**By**.**className**(**"btn"**));((**JavascriptExecutor**)**dr**).**executeScript**(**"$(arguments[0]).fadeOut();"**,**button**);**

Thread**.**sleep**(**1000**);**

System**.**out**.**println**(**"browser will be close"**);**

dr**.**quit**();**

**}**

**}**

# 处理alert/confirm/prompt

## 场景

webdriver中处理原生的js alert confirm 以及prompt是很简单的。具体思路是使用switch\_to.alert()方法定位到alert/confirm/prompt。然后使用text/accept/dismiss/send\_keys按需进行操做

* getText。返回alert/confirm/prompt中的文字信息
* accept。点击确认按钮
* dismiss。点击取消按钮，如果有的话
* sendKeys。向prompt中输入文字

## 代码

下面代码简单的演示了如何去处理原生的alert

### alert.html

**<**html**>**

**<**head**>**

**<**metahttp**-**equiv**=**"content-type"content**=**"text/html;charset=utf-8"**/>**

**<**title**>**alert**</**title**>**

**<**scripttype**=**"text/javascript"async**=**""src**=**"http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"**></**script**>**

**<**linkhref**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css"rel**=**"stylesheet"**/>**

**<**scripttype**=**"text/javascript"**>**

$**(**document**).**ready**(**function**(){**

$**(**'#tooltip'**).**tooltip**({**"placement"**:**"right"**});**

$**(**'#tooltip'**).**click**(**function**(){**

alert**(**'watir**-**webdriverbetterthanselenium**-**webdriver'**)**

**});**

**});**

**</**script**>**

**</**head**>**

**<**body**>**

**<**divclass**=**"row-fluid"**>**

**<**divclass**=**"span6 well"**>**

**<**h3**>**alert**</**h3**>**

**<**aid**=**"tooltip"href**=**"#"data**-**toggle**=**"tooltip"title**=**"watir-webdriver better than selenium-webdriver"**>**hovertoseetooltip**</**a**>**

**</**div**>**

**</**div**>**

**</**body**>**

**<**scriptsrc**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/js/bootstrap.min.js"**></**script**>**

**</**html**>**

### alert.java

**import**java.io.File**;**

**import**java.util.List**;**

**import**org.openqa.selenium.Alert**;**

**import**org.openqa.selenium.By**;**

**import**org.openqa.selenium.JavascriptExecutor**;**

**import**org.openqa.selenium.Keys**;**

**import**org.openqa.selenium.WebDriver**;**

**import**org.openqa.selenium.WebElement**;**

**import**org.openqa.selenium.chrome.ChromeDriver**;**

**publicclassAlertExample{**

**publicstaticvoidmain(**String**[]**args**)throws**InterruptedException**{**

WebDriverdr**=new**ChromeDriver**();**

Filefile**=new**File**(**"src/alert.html"**);**

StringfilePath**=**"file:///"**+**file**.**getAbsolutePath**();**

System**.**out**.**printf**(**"now accesss %s \n"**,**filePath**);**

dr**.**get**(**filePath**);**

Thread**.**sleep**(**1000**);**

*// 点击链接弹出alert*

dr**.**findElement**(**By**.**id**(**"tooltip"**)).**click**();**

Alertalert**=**dr**.**switchTo**().**alert**();**

alert**.**accept**();**

Thread**.**sleep**(**1000**);**

System**.**out**.**println**(**"browser will be close"**);**

dr**.**quit**();**

**}**

**}**

# wait

## 场景

Wait类的使用场景是在页面上进行某些操作，然后页面上就会出现或隐藏一些元素，此时使用WebDriverWait类的until方法来等待这些效果完成以便进行后续的操作。另外页面加载时有可能会执行一些ajax，这时候也需要去WebDriverWait的until的等待ajax的请求执行完毕。

具体一点的例子前面也曾出现过，点击一个链接然后会出现一个下拉菜单，此时需要先等待下拉菜单出现方可进行点击菜单项的操作。

在实例化WebDriverWait类时，有下面2个构造方法

* public WebDriverWait(WebDriver driver, long timeOutInSeconds)
* public WebDriverWait(WebDriver driver, long timeOutInSeconds, long sleepInMillis)

其参数为

* WebDriver driver。不言而喻
* long timeOutInSeconds。总体的超时时间，最多等这么久。
* long sleepInMillis。每隔多久去检查一次until的结果

另外要注意的是，默认情况下，unitl中的NotFoundException会被忽略，但是其他异常还是正常传播的。你可以通过ignoring(exceptions to add)自己定义需要忽略的异常。

## 代码

下面代码演示了点击按钮后如何等待label出现。这个例子其实没有前面的下拉菜单例子实用。

### wait.html

**<**html**>**

**<**head**>**

**<**metahttp**-**equiv**=**"content-type"content**=**"text/html;charset=utf-8"**/>**

**<**title**>**wait**</**title**>**

**<**scripttype**=**"text/javascript"async**=**""src**=**"http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"**></**script**>**

**<**linkhref**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css"rel**=**"stylesheet"**/>**

**<**scripttype**=**"text/javascript"**>**

$**(**document**).**ready**(**function**(){**

$**(**'#btn'**).**click**(**function**(){**

$**(**'**<**p**><**spanclass**=**"label label-info"**>**waitr**-**webdriver**</**span**></**p**>**'**).**css**(**'margin**-**top'**,**'1em'**).**insertAfter**(**$**(this));**

$**(this).**addClass**(**'disabled'**).**unbind**(**'click'**);**

**});**

**});**

**</**script**>**

**</**head**>**

**<**body**>**

**<**divclass**=**"row-fluid"**>**

**<**divclass**=**"span6 well"**>**

**<**h3**>**wait**</**h3**>**

**<**buttonclass**=**"btn btn-primary"id**=**"btn"**>**Click**</**button**>**

**</**div**>**

**</**div**>**

**</**body**>**

**<**scriptsrc**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/js/bootstrap.min.js"**></**script**>**

**</**html**>**

### wait.java

**import**java.io.File**;**

**import**java.util.List**;**

**import**org.openqa.selenium.Alert**;**

**import**org.openqa.selenium.By**;**

**import**org.openqa.selenium.JavascriptExecutor**;**

**import**org.openqa.selenium.Keys**;**

**import**org.openqa.selenium.WebDriver**;**

**import**org.openqa.selenium.WebElement**;**

**import**org.openqa.selenium.chrome.ChromeDriver**;**

**import**org.openqa.selenium.support.ui.ExpectedCondition**;**

**import**org.openqa.selenium.support.ui.WebDriverWait**;**

**publicclassWaitExample{**

**publicstaticvoidmain(**String**[]**args**)throws**InterruptedException**{**

WebDriverdr**=new**ChromeDriver**();**

Filefile**=new**File**(**"src/wait.html"**);**

StringfilePath**=**"file:///"**+**file**.**getAbsolutePath**();**

System**.**out**.**printf**(**"now accesss %s \n"**,**filePath**);**

dr**.**get**(**filePath**);**

Thread**.**sleep**(**1000**);**

*// 点击按钮*

dr**.**findElement**(**By**.**id**(**"btn"**)).**click**();**

**(new**WebDriverWait**(**dr**,**10**)).**until**(new**ExpectedCondition**<**Boolean**>(){**

**public**Boolean**apply(**WebDriverd**){**

**return**d**.**findElement**(**By**.**className**(**"label"**)).**isDisplayed**();**

**}**

**});**

Thread**.**sleep**(**1000**);**

System**.**out**.**println**(**"browser will be close"**);**

dr**.**quit**();**

**}**

**}**

# 定位frame中的元素

## 场景

处理frame需要用到2个方法，分别是switchTo().frame(element|index|id)和switchTo.defaultContent()

switchTo().frame()方法的参数值得一提。其支持

* WebElement，可以传入一个已经定位的frame元素。如 switchTo().frame(dr.findElement(By.id("myFrame")))
* int index，可以传入页面上frame的索引，如0表示第1个frame
* String id, 可以传入frame的id

switchTo().frame()方法把当前定位的主体切换了frame里。怎么理解这句话呢？我们可以从frame的实质去理解。frame中实际上是嵌入了另一个页面，而webdriver每次只能在一个页面识别，因此才需要用switch\_to.frame方法去获取frame中嵌入的页面，对那个页面里的元素进行定位。

switchTo.defaultContent方法的话则是从frame中嵌入的页面里跳出，跳回到最外面的原始页面中。

如果页面上只有1个frame的话那么这一切都是很好理解的，但如果页面上有多个frame，情况有稍微有点复杂了。

## 代码

下面的代码中frame.html里有个id为f1的frame，而f1中又嵌入了id为f2的frame，该frame加载了百度的首页。

### frame.html

**<**html**>**

**<**head**>**

**<**metahttp**-**equiv**=**"content-type"content**=**"text/html;charset=utf-8"**/>**

**<**title**>**frame**</**title**>**

**<**scripttype**=**"text/javascript"async**=**""src**=**"http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"**></**script**>**

**<**linkhref**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css"rel**=**"stylesheet"**/>**

**<**scripttype**=**"text/javascript"**>**

$**(**document**).**ready**(**function**(){**

**});**

**</**script**>**

**</**head**>**

**<**body**>**

**<**divclass**=**"row-fluid"**>**

**<**divclass**=**"span10 well"**>**

**<**h3**>**frame**</**h3**>**

**<**iframeid**=**"f1"src**=**"inner.html"width**=**"800"**,**height**=**"600"**></**iframe**>**

**</**div**>**

**</**div**>**

**</**body**>**

**<**scriptsrc**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/js/bootstrap.min.js"**></**script**>**

**</**html**>**

### inner.html

**<**html**>**

**<**head**>**

**<**metahttp**-**equiv**=**"content-type"content**=**"text/html;charset=utf-8"**/>**

**<**title**>**inner**</**title**>**

**</**head**>**

**<**body**>**

**<**divclass**=**"row-fluid"**>**

**<**divclass**=**"span6 well"**>**

**<**h3**>**inner**</**h3**>**

**<**iframeid**=**"f2"src**=**"http://www.baidu.com"width**=**"700"height**=**"500"**></**iframe**>**

**<**ahref**=**"javascript:alert('watir-webdriver better than selenium webdriver;')"**>**click**</**a**>**

**</**div**>**

**</**div**>**

**</**body**>**

**</**html**>**

### frame.java

**import**java.io.File**;**

**import**java.util.List**;**

**import**org.openqa.selenium.Alert**;**

**import**org.openqa.selenium.By**;**

**import**org.openqa.selenium.JavascriptExecutor**;**

**import**org.openqa.selenium.Keys**;**

**import**org.openqa.selenium.WebDriver**;**

**import**org.openqa.selenium.WebElement**;**

**import**org.openqa.selenium.chrome.ChromeDriver**;**

**import**org.openqa.selenium.support.ui.ExpectedCondition**;**

**import**org.openqa.selenium.support.ui.WebDriverWait**;**

**publicclassFrame{**

**publicstaticvoidmain(**String**[]**args**)throws**InterruptedException**{**

WebDriverdr**=new**ChromeDriver**();**

Filefile**=new**File**(**"src/frame.html"**);**

StringfilePath**=**"file:///"**+**file**.**getAbsolutePath**();**

System**.**out**.**printf**(**"now accesss %s \n"**,**filePath**);**

dr**.**get**(**filePath**);**

Thread**.**sleep**(**1000**);**

*// 先到f1再到f2*

dr**.**switchTo**().**frame**(**"f1"**);**

dr**.**switchTo**().**frame**(**"f2"**);**

*// 往f2中的百度关键字文本框中输入内容*

dr**.**findElement**(**By**.**id**(**"kw"**)).**sendKeys**(**"watir-webdriver"**);**

Thread**.**sleep**(**1000**);**

*// 直接跳出所有frame*

dr**.**switchTo**().**defaultContent**();**

*// 再到f1*

dr**.**switchTo**().**frame**(**"f1"**);**

dr**.**findElement**(**By**.**linkText**(**"click"**)).**click**();**

Thread**.**sleep**(**1000**);**

System**.**out**.**println**(**"browser will be close"**);**

dr**.**quit**();**

**}**

**}**

## 讨论

假设页面上有A、B两个frame，其中B在A内，那么定位B中的内容则需要先到A，然后再到B。如果是定位A中的内容，那么直接switch\_to.frame('A')就可以了;

页面中使用frame会影响页面渲染速度，如果你遇到页面中有多个frame的情况，你完全可以提出1个页面前端性能的缺陷;

如果实在搞不定页面上的frame，送你一句歌词：也许放弃才能靠近你。那么及时放弃跟此frame相关的用例才是明智之举;

# action

## 场景

由于webdriver是要模拟真实的用户操作，因此webdriver的Action类中提供了很多与操作有关的方法。

下面列举一下Action类的一些主要方法

* keyDown。模拟按键按下
* keyUp。模拟按键弹起
* click
* sendKeys
* doubleClick。鼠标左键双击
* clickAndHold。鼠标左键点击住不放
* release。鼠标左键弹起，可以与click\_and\_hold配合使用
* moveToElement。把鼠标移动到元素的中心点
* contextClick。鼠标右键点击
* dragAndDrop。拖拽

## 代码

Actions action **=new**Actions**(**driver**)**

action**.**keyDown**(**Keys**.**SHIFT**).**

click**(**element**).**

click**(**second\_element**).**

keyUp**(**Keys**.**SHIFT**).**

dragAndDrop**(**element**,** third\_element**).**

build**().**

perform**()**

## 讨论

具体使用方法可以参考api文档。action的api文档算是比较全面了。

# 上传文件

## 场景

上传文件的方法是找到上传文件的对象，通常是的对象。然后直接往这个对象sendKeys，传入需要上传文件的正确路径。绝对路径和相对路径都可以，但是上传的文件必须存在，否则会报错。

## 代码

### upload\_file.html

**<**html**>**

**<**head**>**

**<**metahttp**-**equiv**=**"content-type"content**=**"text/html;charset=utf-8"**/>**

**<**title**>**upload\_file**</**title**>**

**<**scripttype**=**"text/javascript"async**=**""src**=**"http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"**></**script**>**

**<**linkhref**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css"rel**=**"stylesheet"**/>**

**<**scripttype**=**"text/javascript"**>**

**</**script**>**

**</**head**>**

**<**body**>**

**<**divclass**=**"row-fluid"**>**

**<**divclass**=**"span6 well"**>**

**<**h3**>**upload\_file**</**h3**>**

**<**inputtype**=**"file"name**=**"file"**/>**

**</**div**>**

**</**div**>**

**</**body**>**

**<**scriptsrc**=**"http://netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/js/bootstrap.min.js"**></**script**>**

**</**html**>**

### upload\_file.java

**import**java.io.File**;**

**import**java.util.List**;**

**import**org.openqa.selenium.Alert**;**

**import**org.openqa.selenium.By**;**

**import**org.openqa.selenium.Keys**;**

**import**org.openqa.selenium.WebDriver**;**

**import**org.openqa.selenium.WebElement**;**

**import**org.openqa.selenium.chrome.ChromeDriver**;**

**publicclassUpload{**

**publicstaticvoidmain(**String**[]**args**)throws**InterruptedException**{**

WebDriverdr**=new**ChromeDriver**();**

Filefile**=new**File**(**"src/upload\_file.html"**);**

StringfilePath**=**"file:///"**+**file**.**getAbsolutePath**();**

System**.**out**.**printf**(**"now accesss %s \n"**,**filePath**);**

dr**.**get**(**filePath**);**

Thread**.**sleep**(**1000**);**

dr**.**findElement**(**By**.**cssSelector**(**"input[type=file]"**)).**sendKeys**(**"src/navs.html"**);**

Thread**.**sleep**(**1000**);**

System**.**out**.**println**(**"browser will be close"**);**

dr**.**quit**();**

**}**

**}**

# 下载

## 场景

webdriver允许我们设置默认的文件下载路径。也就是说文件会自动下载并且存在设置的那个目录中。

下面会给出firefox浏览器的具体设置方法。

## 代码

driver**=**Selenium:**:**WebDriver**.**for**:**chrome**,:**profile**=>**profile

#**for**firefox

FirefoxProfilefirefoxProfile**=new**FirefoxProfile**();**

firefoxProfile**.**setPreference**(**"browser.download.folderList"**,**2**);**

firefoxProfile**.**setPreference**(**"browser.download.manager.showWhenStarting"**,false);**

firefoxProfile**.**setPreference**(**"browser.download.dir"**,**"c:\\downloads"**);**

firefoxProfile**.**setPreference**(**"browser.helperApps.neverAsk.saveToDisk"**,**"text/csv"**);**

WebDriverdriver**=new**FirefoxDriver**(**firefoxProfile**);**

*//new RemoteWebDriver(new URL("http://localhost:4444/wd/hub"), capability);*

driver**.**navigate**().**to**(**"http://www.myfile.com/hey.csv"**);**

# 超时设置

## 场景

webdriver中可以设置很多的超时时间

* implicitlyWait。识别对象时的超时时间。过了这个时间如果对象还没找到的话就会抛出NoSuchElement异常
* setScriptTimeout。异步脚本的超时时间。webdriver可以异步执行脚本，这个是设置异步执行脚本脚本返回结果的超时时间
* pageLoadTimeout。页面加载时的超时时间。因为webdriver会等页面加载完毕在进行后面的操作，所以如果页面在这个超时时间内没有加载完成，那么webdriver就会抛出异常

## 代码

#定位对象时给3s的时间

#如果3s内还定位不到则抛出异常

driver**.**manage**().**timeouts**().**implicitlyWait**(**3**,** TimeUnit**.**SECONDS**);**

#页面加载超时时间设置为5s

dr**.**manage**().**timeouts**().**pageLoadTimeout**(**5**,** TimeUnit**.**SECONDS**);**

#异步脚本的超时时间设置成3s

dr**.**manage**().**timeouts**().**setScriptTimeout**(**3**,** TimeUnit**.**SECONDS**);**

# Remote Webdriver

## 场景

简单来说，我们可以把remote webdriver理解成在远程机器上运行webdriver脚本。

想像一下最简单的一个应用场景：你和你的同事两人一起开发一段webdriver脚本，然后你们需要在一个公共的环境去运行这段脚本。为什么要在公共的环境运行？那是因为每个人的开发机器是有差异的，但是如果用同一台测试机的话，那么环境差异的因素就可以基本排除。我们应该经常听到开发说这样的话:"这个bug在我的环境上是好的啊！"。因为运行环境不同而造成的bug比比皆是，因此我们需要一个统一的运行环境来消除差异。

在这样的应用场景下，我们就需要使用remote webdriver，我们在本地开发脚本，然后调用remote webdriver，在测试机器上执行我们的测试。

# 安装

Remote Webdriver的安装很简单。

首先下载[selenium-server-standalone-LAST-VERSION.jar](http://code.google.com/p/selenium/downloads/list)。

然后运行java -jar selenium-server-standalone.jar命令。如果没有错误出现的话，这台机器已经被配置成远程机器了，以后webdriver就会在这台机器上启动浏览器，执行脚本。

## 启动driver

下面的代码可以启动远程机器上的driver，默认情况下这会打开localhost也就是本机上的firefox浏览器

driver = Selenium::WebDriver.for(:remote)

如果你的remote webdriver不在本地运行，而且你又想指定除firefox以外的浏览器，那么使用下面的代码

driver = Selenium::WebDriver.for(:remote, :url => "http://myserver:4444/wd/hub", :desired\_capabilities => :chrome)

通常情况下myserver可以是192.168.x.x之类的ip地址。

另外还可以通过配置Selenium::WebDriver::Remote::Capabilities来实现更加定制化的浏览器配置，这个超出本文范围，不做描述。

## 使用watir-webdriver启动driver

可以使用下面的代码让watir-webdriver也使用remote webdriver模式

browser = Watir::Browser.new(:remote, {desired\_capabilities: :chrome, url: "http://myserver:4444/wd/hub"})

## java版本

// We could use any driver for our tests...

DesiredCapabilities capabilities = new DesiredCapabilities();

// ... but only if it supports javascript

capabilities.setJavascriptEnabled(true);

// Get a handle to the driver. This will throw an exception

// if a matching driver cannot be located

WebDriver driver = new RemoteWebDriver(capabilities);

// Query the driver to find out more information

Capabilities actualCapabilities = ((RemoteWebDriver) driver).getCapabilities();

// And now use it

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

注意，java版本的代码我没有时间去调试，这里只是把wiki上的代码放出来而已。另外remote server在发生错误时会自动截图，下面是获得截图的代码

public String extractScreenShot(WebDriverException e) {

Throwable cause = e.getCause();

if (cause instanceof ScreenshotException) {

return ((ScreenshotException) cause).getBase64EncodedScreenshot();

}

return null;

}

## python版本

c = webdriver.DesiredCapabilities.CHROME

driver = webdriver.Remote(command\_executor='http://127.0.0.1:4444/wd/hub', desired\_capabilities=c)

注意，python binding的wiki中使用的启动remote webdriver的代码跟我上面给出的不太相同，可能是因为我的selenium版本较低(30)，最新版本的同学可以试试[wiki](http://code.google.com/p/selenium/wiki/PythonBindings)上的代码。

# cookie

## 场景

webdriver可以读取并添加cookie。有时候我们需要验证浏览器中是否存在某个cookie，因为基于真实的cookie的测试是无法通过白盒和集成测试完成的。

另外更加常见的一个场景是自动登陆。有很多系统的登陆信息都是保存在cookie里的，因此只要往cookie中添加正确的值就可以实现自动登陆了。什么图片验证码、登陆的用例就都是浮云了。

webdriver读写cookie的接口有以下一些

* addCookie(Cookie cookie)。添加cookie，参数是Cookie对象
* deleteAllCookies。删除所有cookie
* getCookies。返回所有的cookie
* deleteCookieNamed(String name)。删除name这个cookie
* getCookieNamed(String name)。返回特定name的cookie值

## 代码

下面的代码演示了如何自动登陆百度。其中敏感信息我使用了xxxx来代替。

### cookie.java

**import**org.openqa.selenium.Cookie**;**

**import**org.openqa.selenium.WebDriver**;**

**import**org.openqa.selenium.chrome.ChromeDriver**;**

**publicclassCookieExample{**

**publicstaticvoidmain(**String**[]**args**)throws**InterruptedException**{**

WebDriverdr**=new**ChromeDriver**();**

Stringurl**=**"http://www.baidu.com"**;**

System**.**out**.**printf**(**"now accesss %s \n"**,**url**);**

dr**.**get**(**url**);**

Thread**.**sleep**(**2000**);**

System**.**out**.**println**(**dr**.**manage**().**getCookies**());**

dr**.**manage**().**deleteAllCookies**();**

Cookiec1**=new**Cookie**(**"BAIDUID"**,**"xxxxxxxxxx"**);**

Cookiec2**=new**Cookie**(**"BDUSS"**,**"xxxxxxxxxx"**);**

dr**.**manage**().**addCookie**(**c1**);**

dr**.**manage**().**addCookie**(**c2**);**

System**.**out**.**println**(**"browser will be close"**);**

dr**.**quit**();**quit**}**

**}**