Selenium2 初学者快速入门

# 1 摘要

软件规模会越来越大，重复的测试工作会越来越多，自动化测试是一种趋势，一种必然， 随着企业越来越重视自己研发、重视知识产权，开源自动化测试工具越来越受到重视与使用。

Selenium 是时下很流行的面向 web 的自动化测试工具，它以执行效率高，覆盖的浏览 器广泛等优点得到了很多人的亲睐。Selenium 是一个健壮的工具集合，跨很多平台支持针 对基于 web 的应用程序的功能自动化测试工具。它是一个开源的、轻量级的自动化工具， 很容易集成到各种项目中，支持多种编程语言，比如 .NET、Perl、Python、Ruby、C# 和 Java 编程语言。

本文从初学者的角度对自动化测试工具 Selenium 2（即 WebDriver）进行全面阐述，重点 介绍在非测试框架下使用 Selenium 进行自动化测试，因为只有掌握 Selenium 的自动化测试 原理，以后在各种测试框架下才能灵活运用，达到游刃有余。

# 利用 Selenium 实现 web 自动化测试的优 势

相比 QTP，RFT 昂贵的成本，作为开源工具的 Selenium WebDriver 自然不用多说。而且 selenium 是一款基于浏览器的测试工具，因此在响应 UI 请求时运行速度比较可观，能很好的节省运 行时间，提高执行效率。在与大多数测试平台的整合以及可扩展的脚本语言种类上(Java、 dotNET、Perl、Python、Ruby、C#等)较之其他工具也有很大优势，最后，Selenium 支持多 浏览器操作(IE，Firefox，Safari )，这也是其他测试工具所不具备的，当然，没有什么万能的 测试工具，在全面评估被测系统和测试需求后，合适的就是最好的;而且在做自动化测试过 程中，往往不能只单独使用一种自动化工具，结合不同自动化工具的优势来达到我们的目的 是最佳的实践。

# Selenium 的进化

Selenium 已经从之前的 1.0(RC)进化到了现在的 Selenium2(Selenium1+WebDriver)。

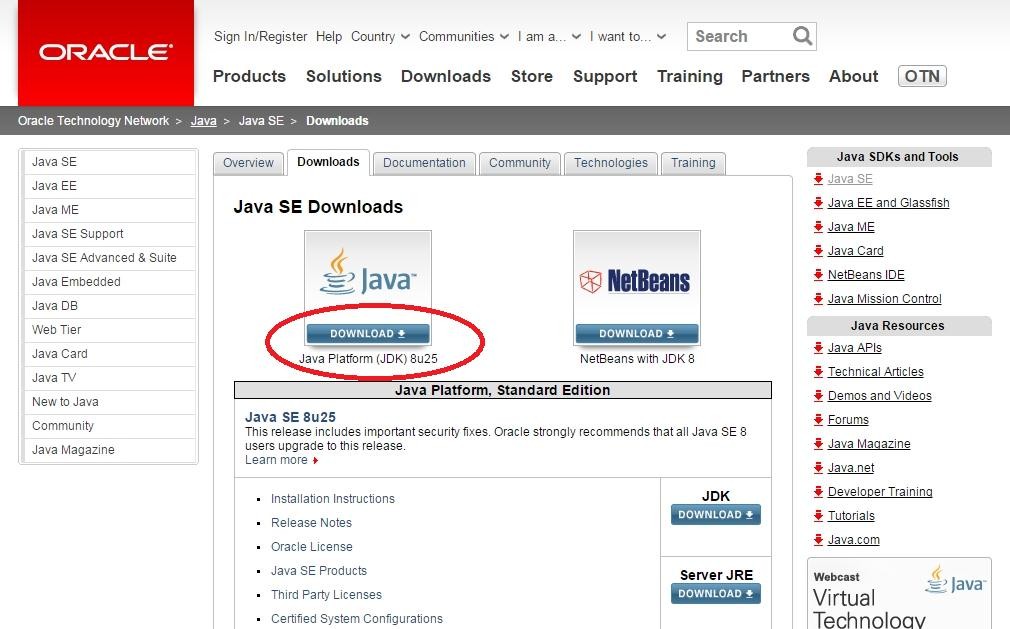
在运行 Selenium1.0 程序之前，我们必须启动 Selenium server 端，也就是 Selenium Remote

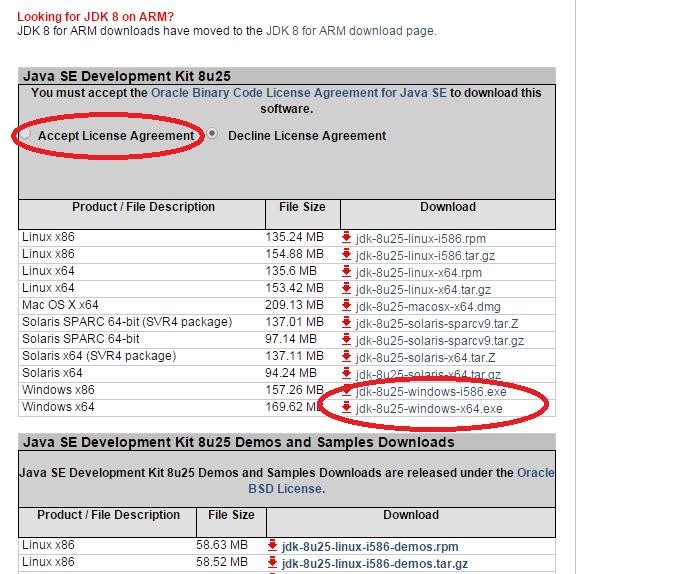
control,我们简称 RC,RC 主要包括三个部分，launcher,http proxy,selenium core, 其中 selenium core 是由一堆 javascript 函数构成,通过调用这些函数，来实现对浏览器的各种操作。既然已 经可以实现对浏览器的操作，那为什么还需要 Selenium2(Selenium1+Webdriver)呢? Selenium1 主要存在以下几个缺点 1.没有原生的鼠标和键盘事件;2.XSS/HTTP 同源数据问题;3.popup dialog 问题。Webdriver 对不同浏览器的处理和 Selenium1.0 有着明显的不同,Selenium1.0 不 管是什么浏览器，都是由 javascript 来处理,而 webdriver 是选择浏览器最容易识别的语言来 处理,比如在 Firefox 中 javascript 最容易,在 IE 中 C++最容易识别,通过灵活选择最容易识别的 语言来处理多浏览器,我们就可以很好的回避某些浏览器对 javascript 的安全限制,Webdriver 不仅可以处理这方面的问题,而且可以调用操作系统 API,尤其是当用户需要模拟鼠标或键盘 操作时,这项能力的作用表现的尤其明显。通过对比, 看来从 Selenium1.0 进化到 Selenium2.0 还是很有必要的。

# 4 环境搭建

## 安装 jdk

下载 jdk：<http://www.oracle.com/technetwork/java/javase/downloads/index.html>



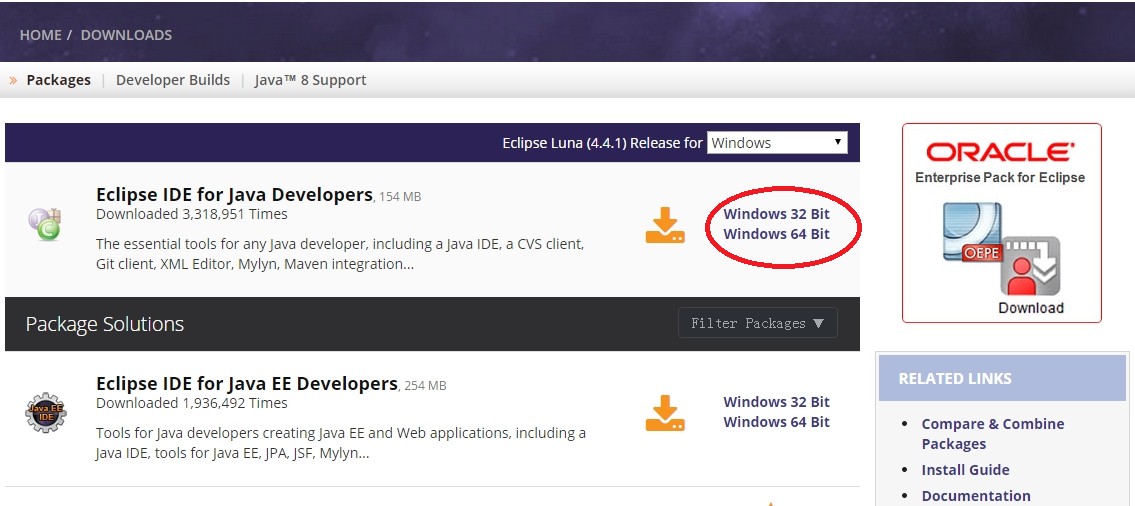


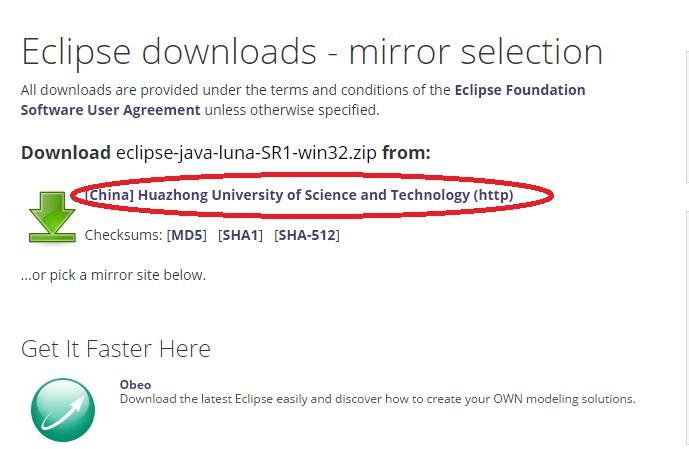
安装完成后配置环境变量： JAVA\_HOME = E:\Java\Java\jdk1.8.0\_25 PATH = %JAVA\_HOME%\bin

CLASSPATH = .;%JAVA\_HOME%\lib\dt.jar;%JAVA\_HOME%\lib\tools.jar

## 安装 eclipse

下载：<http://www.eclipse.org/downloads/>





下载后解压安装。

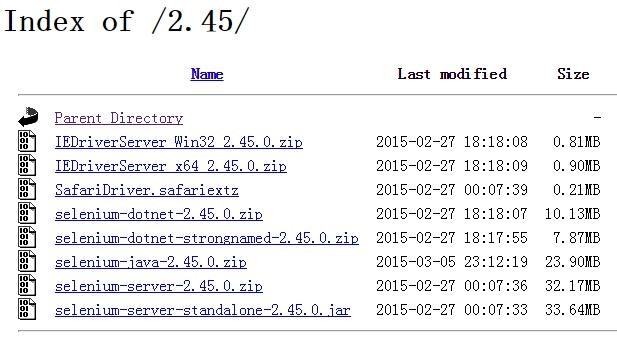
## selenium 下载

Selenium 和 IE 的 driver 的下载：

https://selenium-release.storage.googleapis.com/index.html（需要翻墙才能访问） selenium-server-standalone-2.45.0.jar：模拟服务器端，不可少。

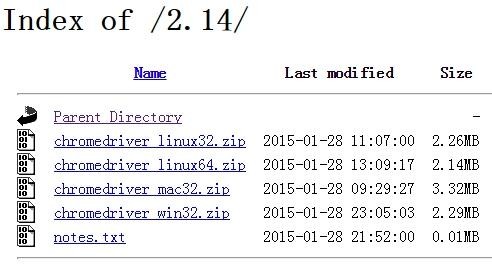
IEDriverServer\_Win32\_2.45.0.zip：IE 驱动，Firfox 不用驱动。

selenium-java-2.45.0.zip ：模拟 Selenium 客户端。



Chrome 浏览器的 driver 下载：

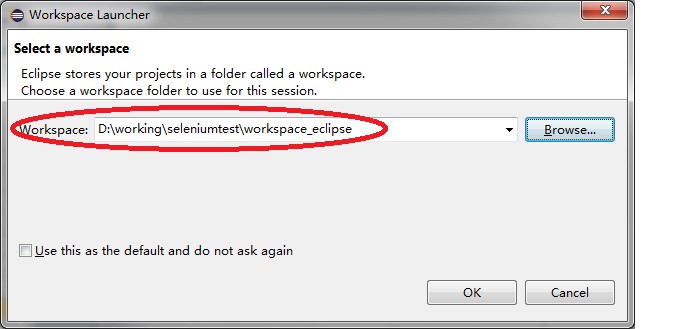
https://chromedriver.storage.googleapis.com/index.html



## 配置 Eclipse IDE 中的 webdriver

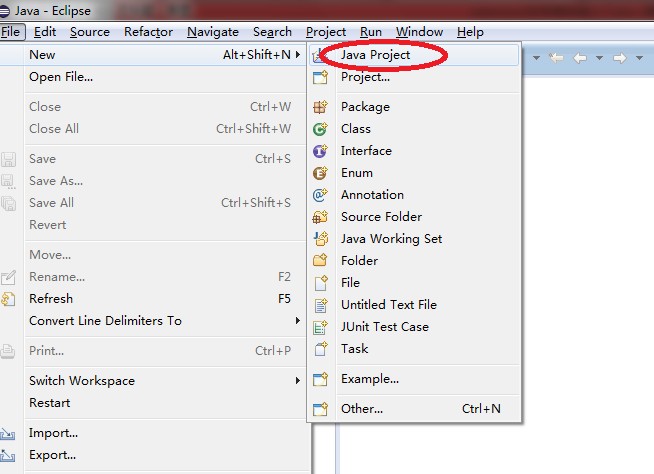
1.启动 eclipse

打开 eclipse，选择新路径创建工作空间。

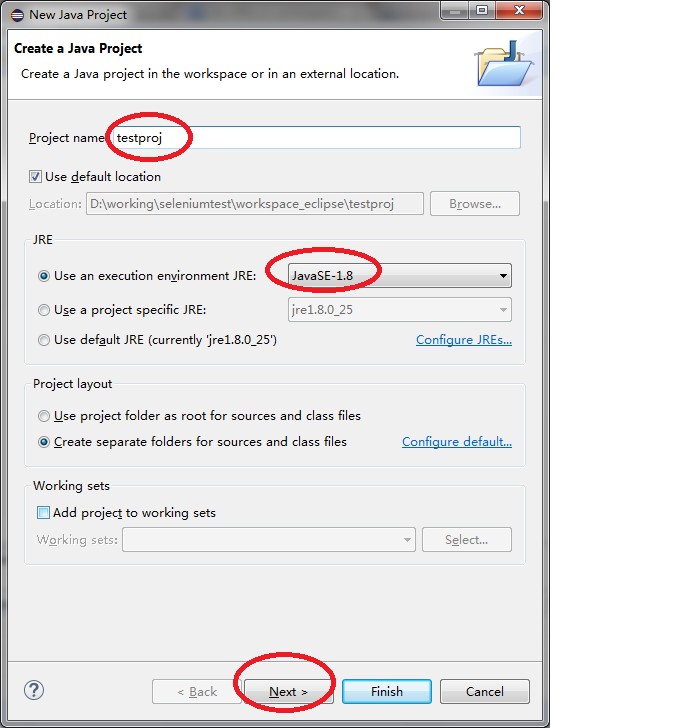


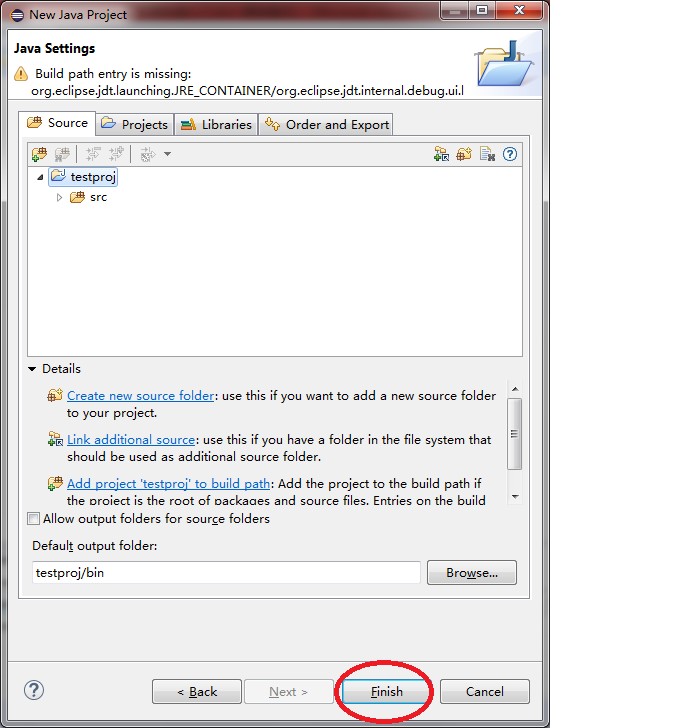
2.创建 java 项目

1）选择 File –> New –> Java Project

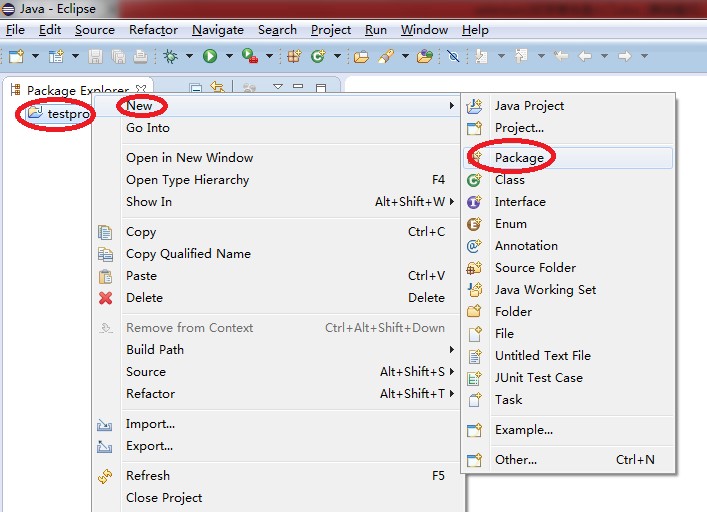


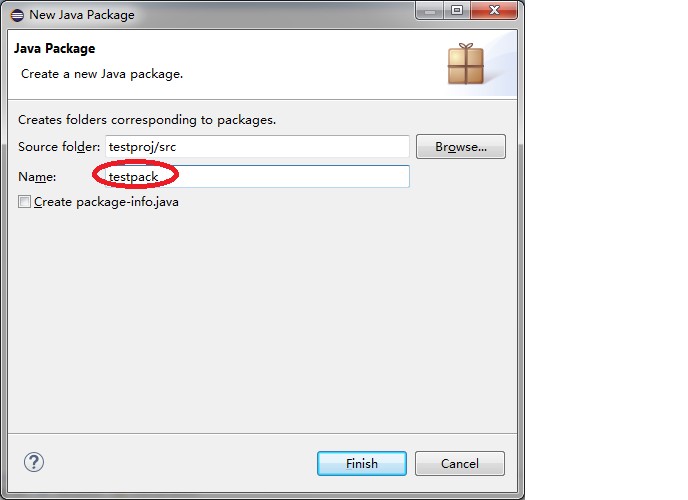
2）输入 Project name，JRE 选择当前安装的 JRE，然后单击 next 或直接点击 Finish



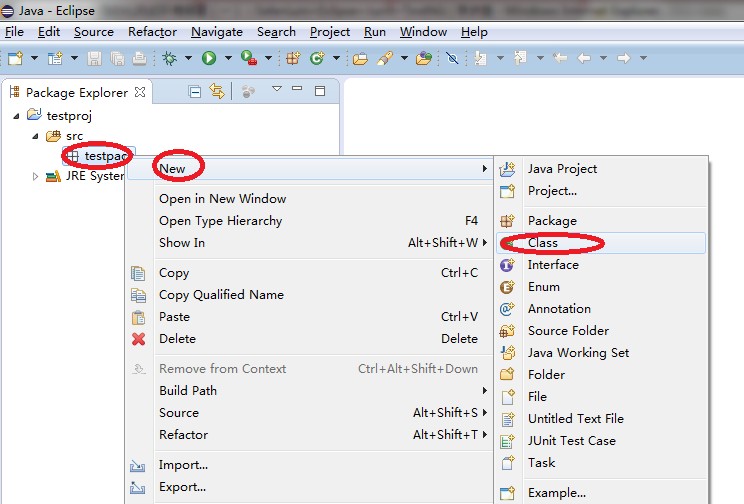


3）进入项目页面，在 Package Explorer 中右键单击该项目名称，选择 New→Package 命令

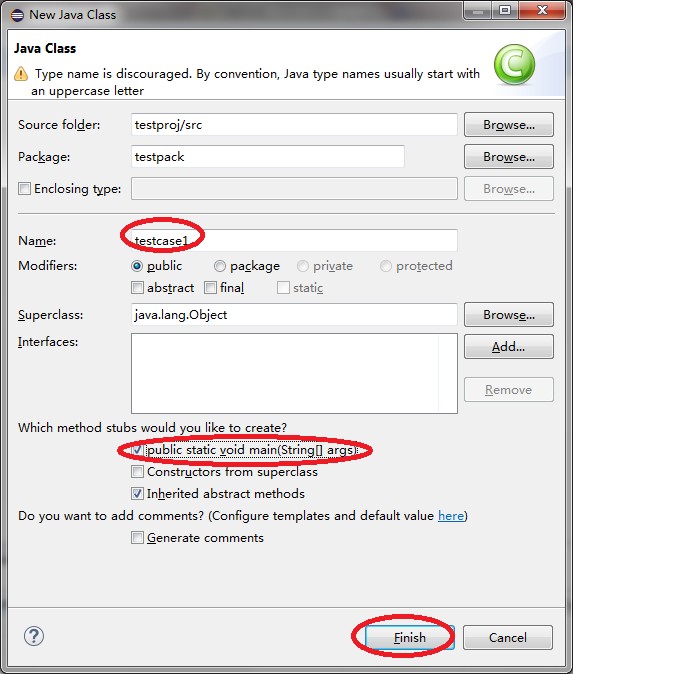


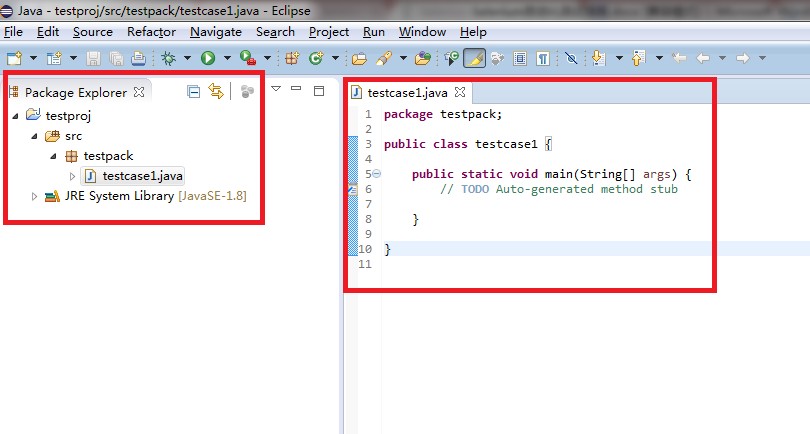
4）输入包名称，点击 finish

5）进入项目页面，在 Package Explorer 中右键单击该包名称，选择 New→Class 命令



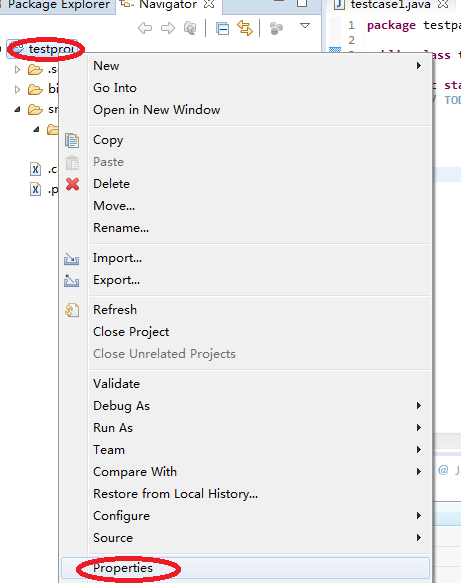
6）输入类名称，并勾选 Public static void main 以生成 main 函数





3.引入 selenium

1）在 Package Explorer 中用鼠标右键单击该项目名称，选择 Properties

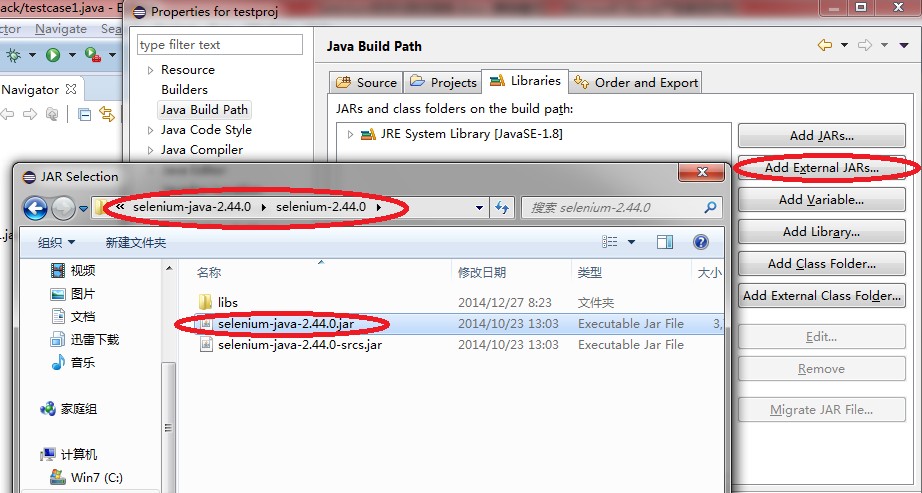


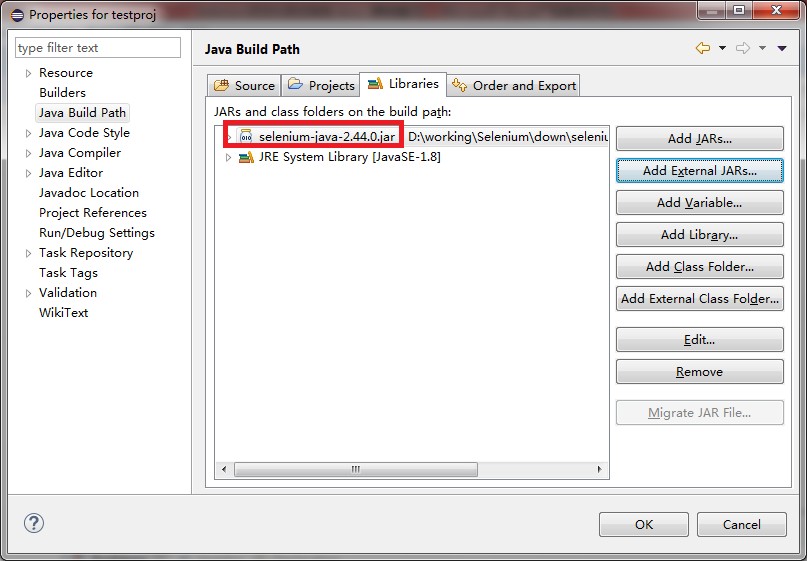
2）选择 Java Build Path，在右边选择 Libraries，单击 Add External JARs

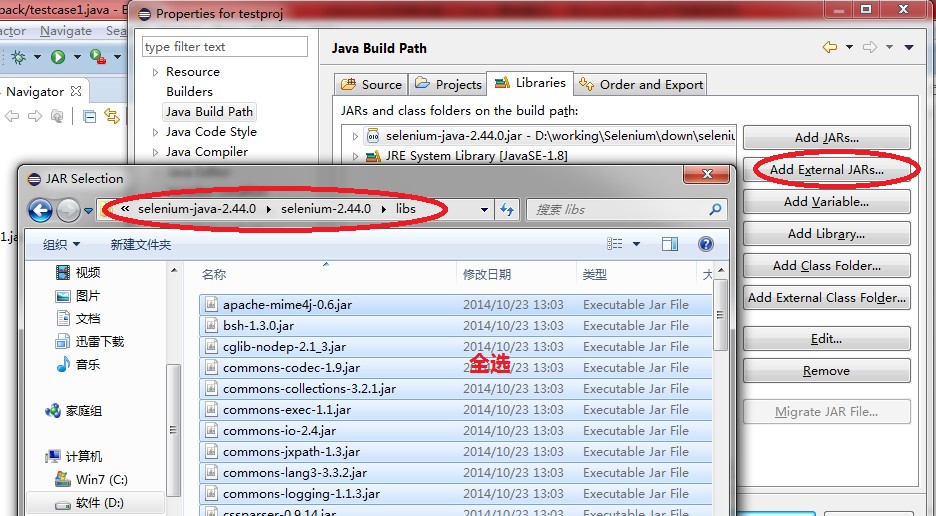


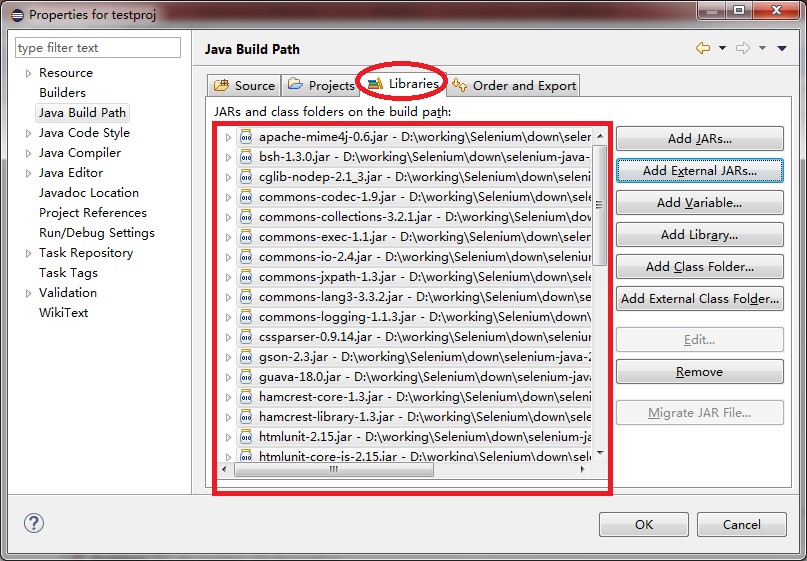
3）选择要添加的 jar 文件

方式 1：使用 selenium-java-2.44.0.zip

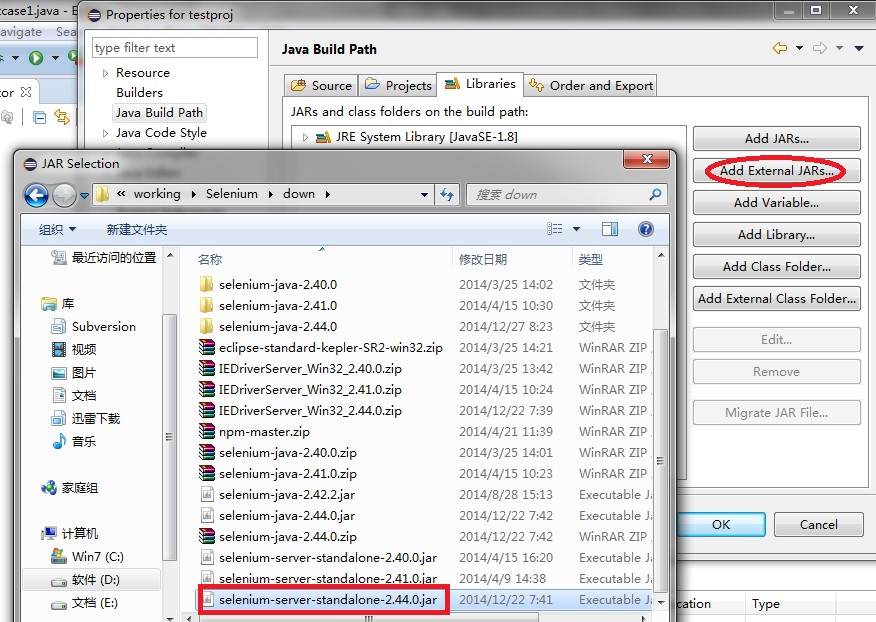


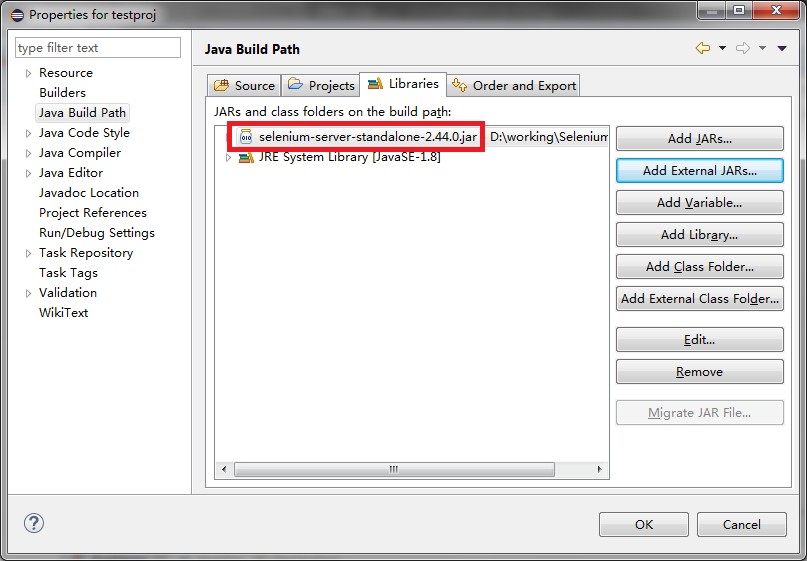






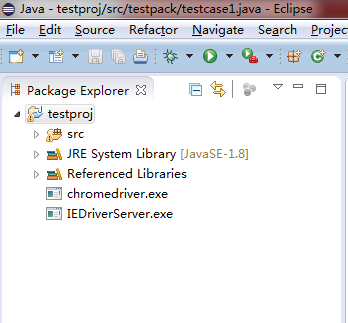
方式 2：使用 selenium-server-standalone-2.44.0.jar





4 将 ie 和 chrome 的浏览器驱动复制到工程

Selenium 启动 ie 和 chrome 浏览器时，需要相应的驱动支持。 将下载的驱动程序复制到 eclipse 工程，方便驱动指定。

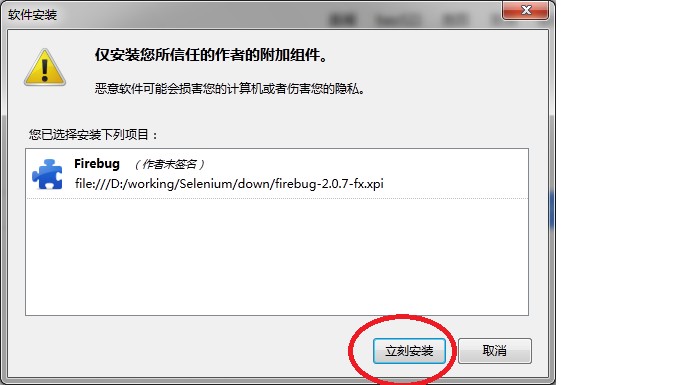


## 4.5 火狐浏览器

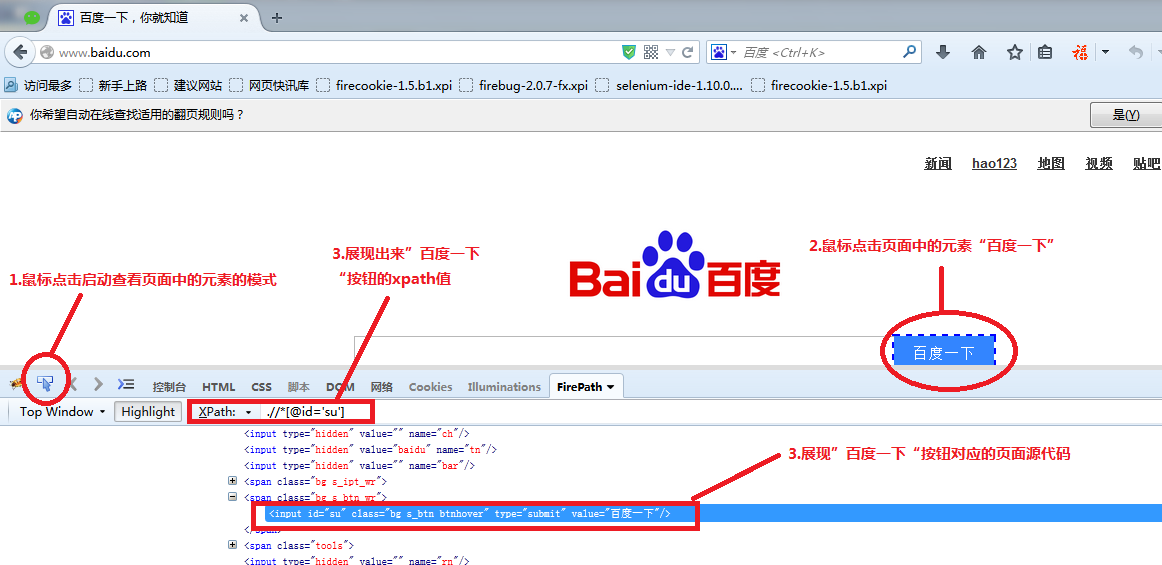
下载最新火狐浏览器，并安装。 下载最新的且兼容当前火狐浏览器的 firebug 组件。 <https://addons.mozilla.org/zh-cn/firefox/addon/firebug/versions/>

firebug 组件能完成 HTML 查看、xpath 获取、xpath 验证，提高页面元素确定的效率。

firebug 组件安装：



firebug 组件功能验证： 按 F12



## 4.5 谷歌浏览器

1. 安装谷歌浏览器

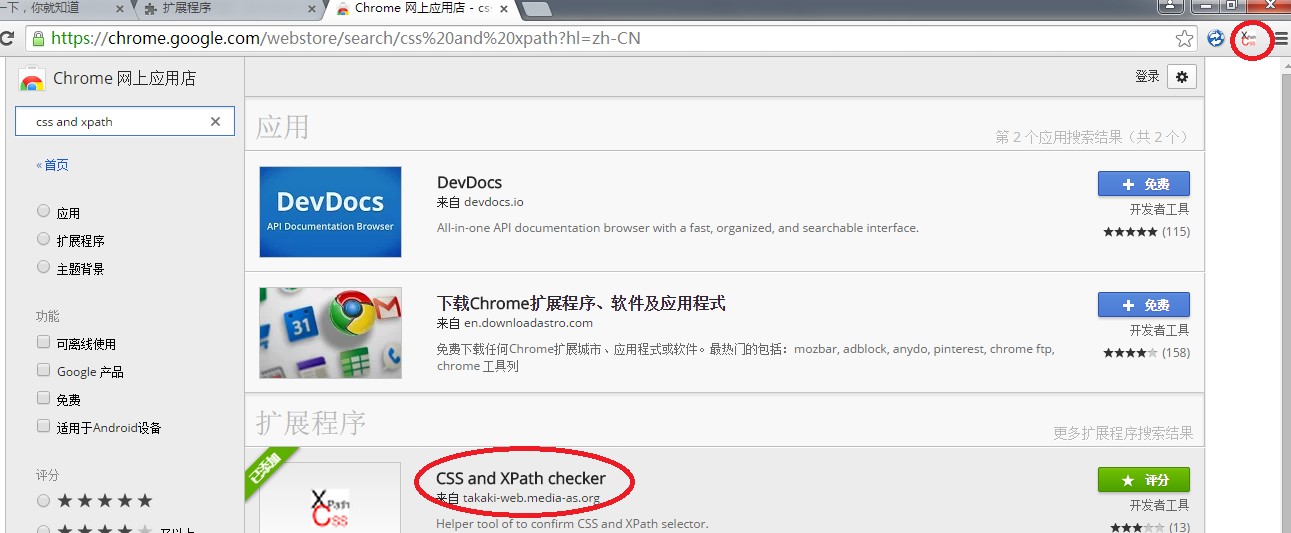
在线安装或者下载 chrome 离线安装程序：

<http://www.google.cn/intl/zh-CN/chrome/browser/desktop/index.html?standalone=1> <https://www.google.com/intl/zh-CN/chrome/browser/?standalone=1>

当前 chrome 最新版本是 39.0.2171.95 m

1. 安装 CSS and XPath checker 插件（验证 xpath）

1）在 chrome 商店或者 google 搜索中搜索插件（需要翻墙），然后安装。



2）安装成功验证：



3.获取页面元素的 xpath

按 F12，点击页面元素，通过右键的 copy xpath 获取 xpath 值。



# selenium 自动化测试基础

<http://www.w3.org/TR/webdriver> <http://code.google.com/p/selenium/wiki> <http://selenium.googlecode.com/git/docs/api/java/index.html>

## 5.1 选择浏览器

在 Selenium 2 中，一共支持以下几种浏览器的测试。

Firefox (FirefoxDriver)。

IE (InternetExplorerDriver)。 Chrome (ChromeDriver)。 Opera (OperaDriver)。 Android (AndroidDriver)。 iPhone (IPhoneDriver)。

* Firefox

//如果启动出现问题，可以使用 System.setProperty 指出 firefox.exe 的路径

//System.setProperty("webdriver.firefox.bin","D:\\ProgramFiles(x86)\\MozillaFirefox\\firefox.exe ");

WebDriver driver=new FirefoxDriver();

* Ie

//System.setProperty("webdriver.ie.driver","D:\\working\\Selenium\\down\\IEDriverServer\_Win 32\_2.44.0\\IEDriverServer.exe");

System.setProperty("webdriver.ie.driver","IEDriverServer.exe");

DesiredCapabilities ieCapabilities = DesiredCapabilities.internetExplorer();

ieCapabilities.setCapability(InternetExplorerDriver.INTRODUCE\_FLAKINESS\_BY\_IGNORING\_SECU RITY\_DOMAINS,true);

WebDriver driver = new InternetExplorerDriver(ieCapabilities);

* Chrome

//System.setProperty("webdriver.chrome.driver","D:\\work\\Selenium\\chromedriver.exe"); System.setProperty("webdriver.chrome.driver","chromedriver");

WebDriver driver = new ChromeDriver();

## 5.2 浏览器导航对象

打开了浏览器器之后，就可以打开指定的页面来进行测试了。要导航页面，需要用到

navigate().to 或者 get 方法来打开页面。

driver.get("[http://www.baidu.com](http://www.baidu.com/)"); driver.navigate().to("[http://www.baidu.com](http://www.baidu.com/)");

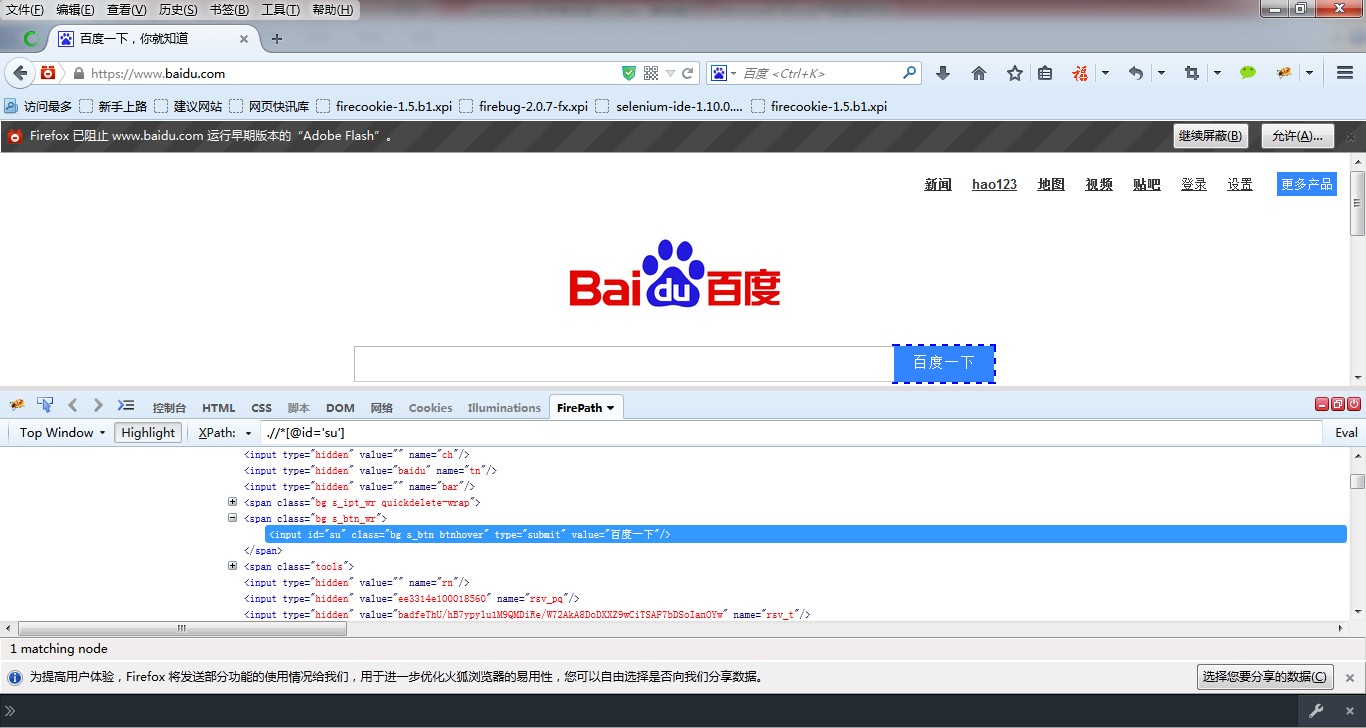
## 5.3 查找页面元素

在导航到对应页面后，就可以对页面上的元素进行操作了。然而，在进行操作之前，必 须要找到相应的元素

根据 HTML 的不同，查找条件也各有不同。例如，可以按 HTML 元素的 ID 进行查找， 也可以按 Name 属性查找，或者直接按 HTML 标签查找，接下来将列举常用的查找条件。

## id

可以按照 HTML 元素的 ID 属性进行查找。例如，百度首页的搜索按钮。



其 HTML 代码如下：

<input id="su" class="bg s\_btn btnhover" type="submit" value="百度一下"/>

如要操作该按钮，则可以通过 ID（id="su"）作为查找条件获取该对象

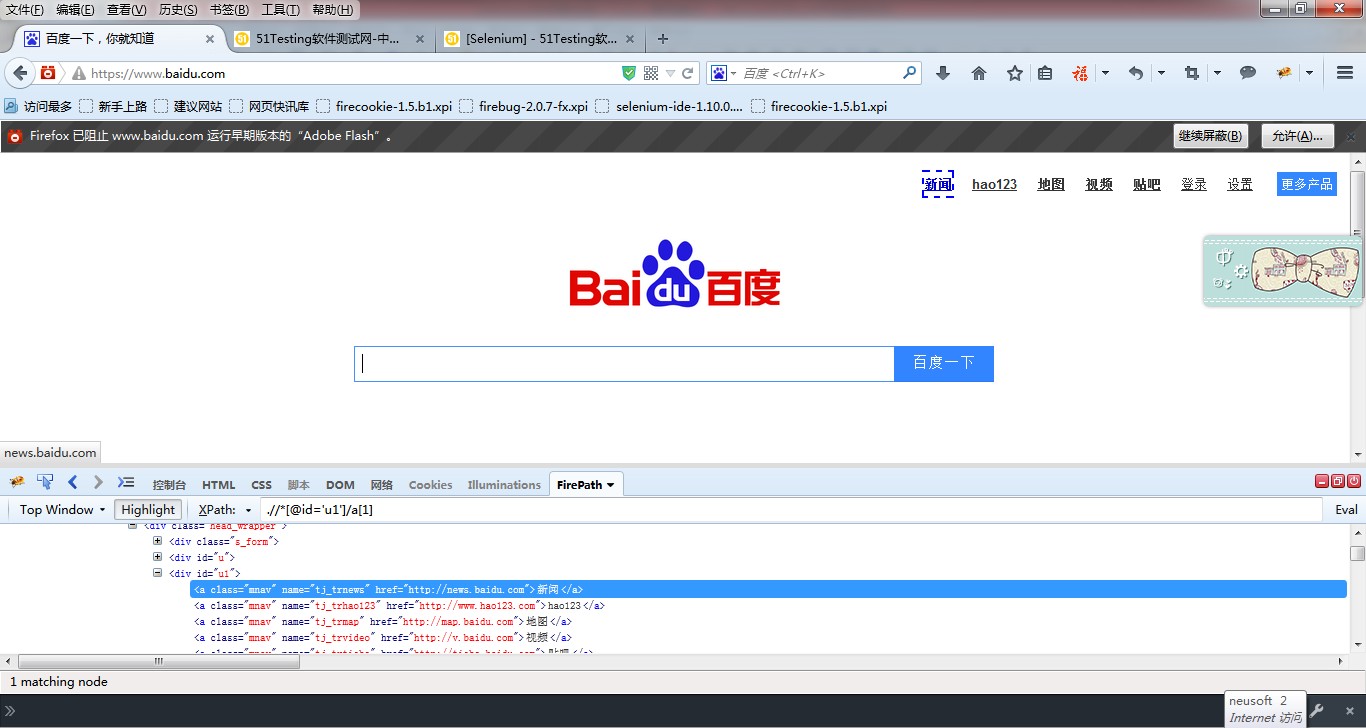
WebElement baiduSearch = driver.findElement(By.id("su"));

另外，还可以执行 js 的 document.getElementById 来获取对象，脚本如下：

WebElement elementC=(WebElement) ((JavascriptExecutor) driver).executeScript("return document.getElementById('su')");

## name

按 Name 进行查找与按 ID 进行查找类似，是根据 HTML 元素的 name 属性进行查找。例如， 百度首页的新闻链接。



其 HTML 代码如下：

<a class="mnav" name="tj\_trnews" href="[http://news.baidu.com](http://news.baidu.com/)">新闻</a>

注意，它的 name 属性为“tj\_trnews”，可以用其作为查找条件来获取登录链接对象

WebElement news= driver.findElement(By.name("tj\_trnews "));

另外，还可以执行 js 的 document. getElementsByName 来获取对象，脚本如下： WebElement elementC=(WebElement) ((JavascriptExecutor) driver).executeScript("return document.getElementsByName('tj\_trnews')[0]");

## LinkText 和 PartialLinkText

LinkText 方法按链接的文本进行查找，PartialLinkText 方法按链接的文本进行模糊查找。 例如，百度首页上的“登录”超级链接



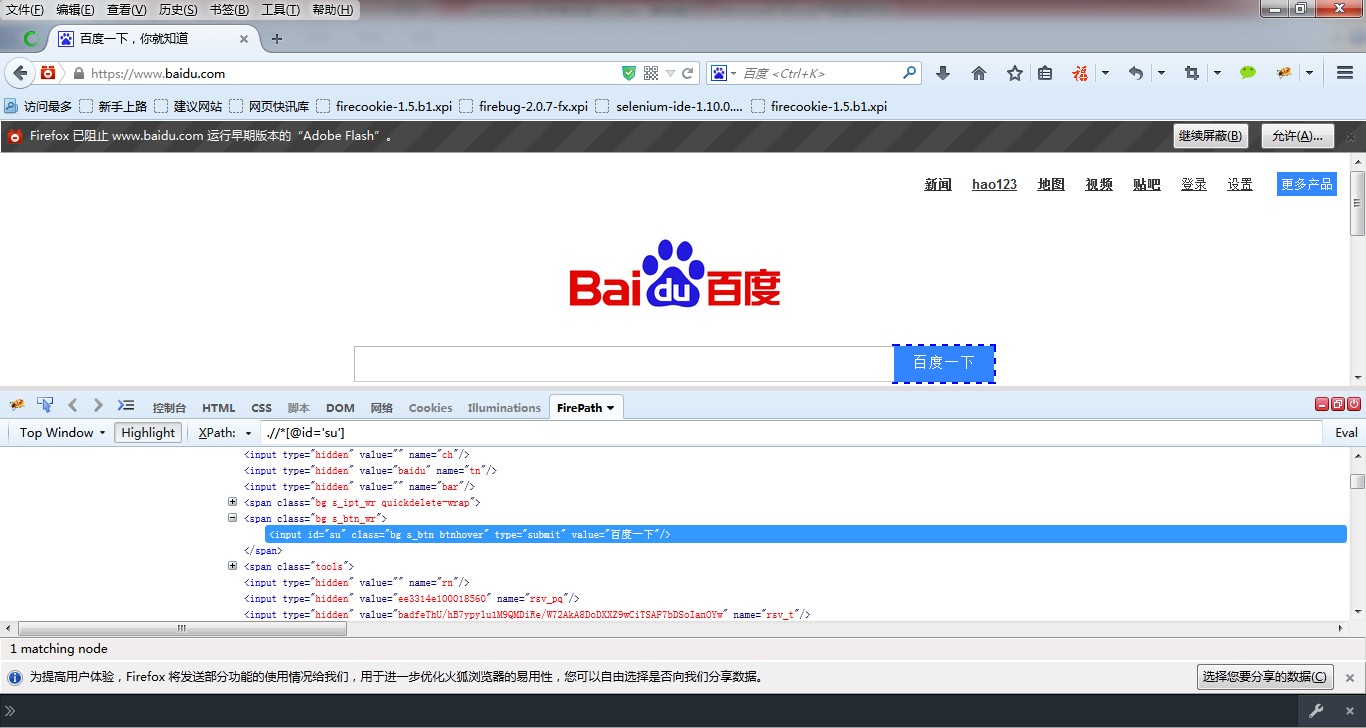
Html:

<a class="pf pfhover" name="tj\_settingicon" href[="h](http://www.baidu.com/gaoji/preferences.html)tt[p://www.baidu.com/gaoji/preferences.html"](http://www.baidu.com/gaoji/preferences.html)>设置</a>

它的链接文本属性为“设置” driver.findElement(By.linkText("设置")).click(); driver.findElement(By.partialLinkText("设")).click();

## XPath

如果以上查找方法都无法定位到指定对象，那么可以按 XPath 进行查找。例如，百度首页的 搜索按钮。



其 HTML 代码如下：

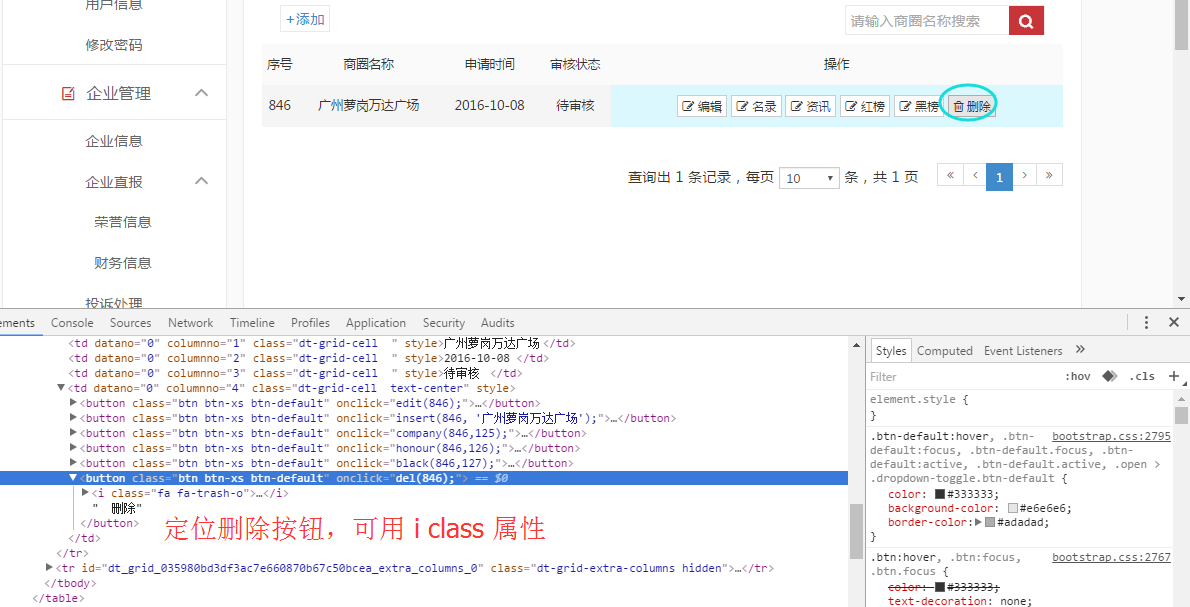
<input id="su" class="bg s\_btn btnhover" type="submit" value="百度一下"/>

如要操作该按钮，则可以通过其 XPath 表达式“//input[@id='su']”作为查找条件获取该对 象，XPath 的值可以通过 firebug 组件获取，也可以自己编写。Xpath 的知识建议自行找相关 资料学习。

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

## *cssSelector*

对于Xpath定位不到的元素，可以尝试使用CssSelector，css选择器的使用方法如下：



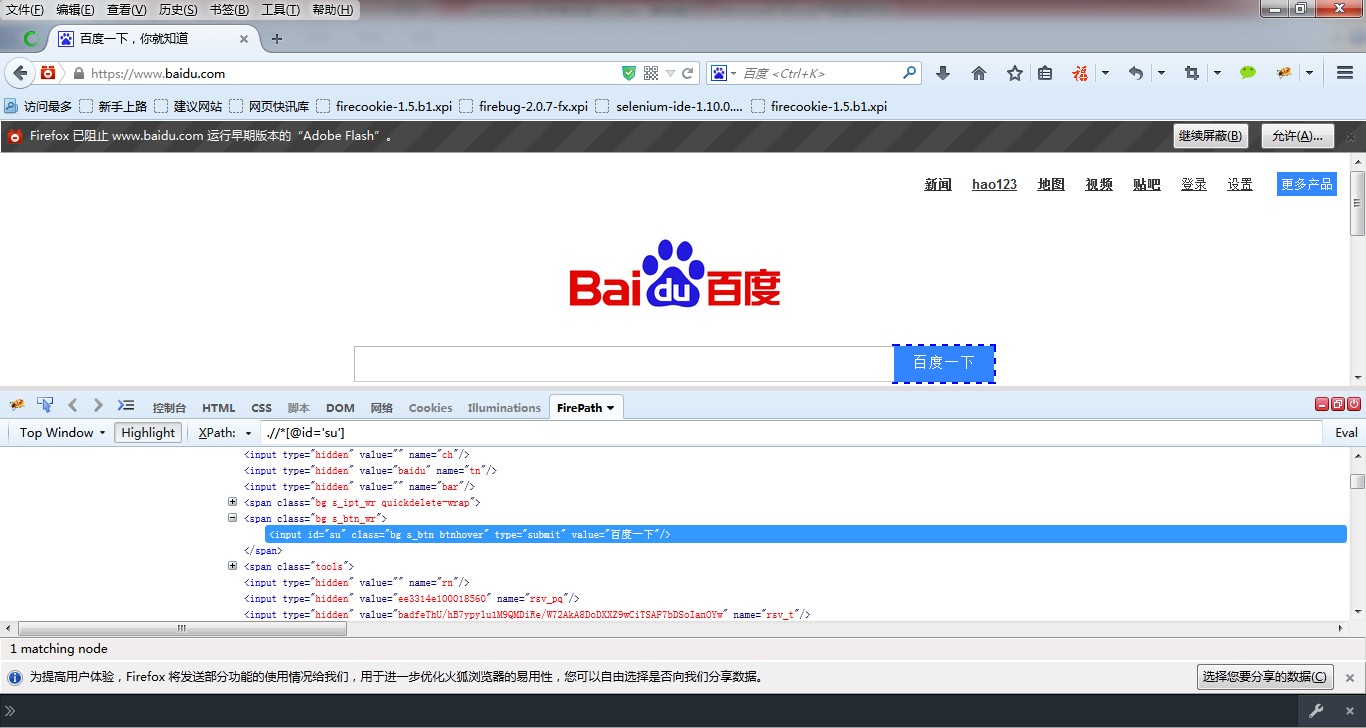
代码实现：driver.findElement(By.cssSelector("i[class='fa fa-trash-o']")).click();//点击删除按钮

## 5.4 操作页面元素

定位到对应的页面元素，然后调用这个页面元素的相关方法对页面进行各种操作。

## 5.4.1 单击

通过 Click()方法和调用 js 来完成单击页面元素的功能。例如，百度首页的搜索按钮。



其 HTML 代码如下：

<input id="su" class="bg s\_btn btnhover" type="submit" value="百度一下"/> 1.Click()方法用于执行单击元素的操作

WebElement baiduSearch = driver.findElement(By.id("su")); baiduSearch.click();

或者

driver.findElement(By.xpath("//input[@id='su']")).click();

2. 键盘模拟单击

Actions actions = new Actions(driver); actions.click(driver.findElement(By.id("su"))).build().perform();

1. 用 javaScript 模拟手工单击操作

使用 WebDriver 点击界面上 Button 元素时，如果当前 Button 元素被界面上其他元素遮住 了，或没出现在界面中(比如 Button 在页面底部，但是屏幕只能显示页面上半部分)，使用默 认的 WebElement.Click()可能会触发不了 Click 事件。

修正方案是找到该页面元素后直接发送一条 Click 的 JavaScript 指令。

((JavascriptExecutor)driver).executeScript("arguments[0].click();",driver.findElement(By.id("su")) )

;

4.调用 js 函数

Html：<input id="queryBt" type="button" onclick="doQuery();" value="查 询" name="queryBt"> ((JavascriptExecutor)driver).executeScript("doQuery ()");

## 5.4.2 输入值

通过 SendKeys()、键盘模拟、调用 js 的方式设置页面元素的值。例如，百度首页有搜索文本 框。

<input id="kw" class="s\_ipt" autocomplete="off" maxlength="100" value="" name="wd"/>

1.SendKeys()方法用于给 input 元素输入文本。 要操作该文本框，需先定位该页面元素，然后使用 SendKeys()方法输入指定内容

WebElement baiduTextBox = driver.findElement(By.id("kw")); baiduTextBox.sendKeys("selenium 书籍");

2. 键盘模拟输入

Actions actiona = new Actions(driver); actiona.sendKeys(baiduTextBox, "qtp"); actiona.build().perform();

3. 用 javaScript 模拟手工输入 ((JavascriptExecutor)driver).executeScript("arguments[0].value=arguments[1]", driver.findElement(By.id("kw")), "selenium 书籍");

或者

((JavascriptExecutor)driver).executeScript("arguments[0].setAttribute('value',arguments[1]);", driver.findElement(By.id("kw")),"selenium 书籍");

4.调用 js 的 document. getElementById 设置属性值

((JavascriptExecutor)driver).executeScript("document.getElementById(\"kw\").value='selenium

书籍'");

## 5.4.3 双击

Actions action = new Actions(driver);

action.doubleClick();// 鼠标在当前停留的位置做双击操作 action.doubleClick(driver.findElement(By.name(element)))// 鼠标双击指定的元素 action.build().perform();

## 5.4.4 鼠标悬停

Actions action = new Actions(driver);

//action.clickAndHold();// 鼠标悬停在当前位置，既点击并且不释放 action.clickAndHold(onElement);// 鼠标悬停在 onElement 元素的位置 action.build().perform();

action.clickAndHold(onElement) 这个方法实际上是执行了两个动作，首先是鼠标移动到元素 onElement ， 然 后 再 clickAndHold, 所 以 这 个 方 法 也 可 以 写 成 action.moveToElement(onElement).clickAndHold()。

## 5.4.5 鼠标移动操作

Actions action = new Actions(driver); action.moveToElement(toElement);// 将鼠标移到 toElement 元素中点 action.build().perform();

## 5.4.6 鼠标右键点击

Actions action = new Actions(driver);

action.contextClick();// 鼠标右键在当前停留的位置做单击操作 action.contextClick(driver.findElement(By.name(element)))// 鼠标右键点击指定的元素 action.build().perform();

## 5.4.7 模拟键盘操作

在实际的自动化测试项目中，经常会在无法获取元素属性的情况下，根据焦点的切换完成输 入、回车或者快捷键等操作

1）利用 Keyboard

Keyboard keyboard = ((RemoteWebDriver) driver).getKeyboard(); keyboard.pressKey(Keys.TAB);

2）利用 Action

Actions actionk = new Actions(driver); actionk.sendKeys(Keys.TAB);// 模拟按下并释放 TAB 键 actionk.build().perform();

ctrl+a，可以直接 actions.sendkeys(Keys.CONTROL+"a").build().perform() 3）利用 Robot

Robot robot=new Robot();

robot.keyPress(KeyEvent.VK\_TAB);

## 5.4.8 上传图片

**分析HTML**，发现无论是普通前台还是Extjs，他们的html有一个共同的特点，具有以下代码：

　　<input ..... type="file" ...../>

ok,只要我们使用selenium自带的sendkeys把文件路径（相对绝对无所谓）传给该元素，就可绕过选择窗口直接上传文件.

**解决方案诞生了**

driver.findElement(XXXXX).sendKeys("filePath");

**Have a Try**

博客园上传图片：driver.findElement(By.name("file")).sendKeys("C:\Users\123\Pictures\a.png");

QQ邮箱上传附件：driver.findElement(By.name("UploadFile")).sendKeys("C:\Users\123\Pictures\a.png");

定位元素根据情况而定，不一定用name；文件路径可以使相对路径，大家自行修改就好

**注意：**执行sendKeys的元素一定要符合input和 type="file"条件,否则就是你没找对上传文件的对象，会上传失败的。

http://images.cnitblog.com/blog2015/553409/201504/161138438704933.png像这样的，一定注意不要上传到前面的那个input上，否则会失败的

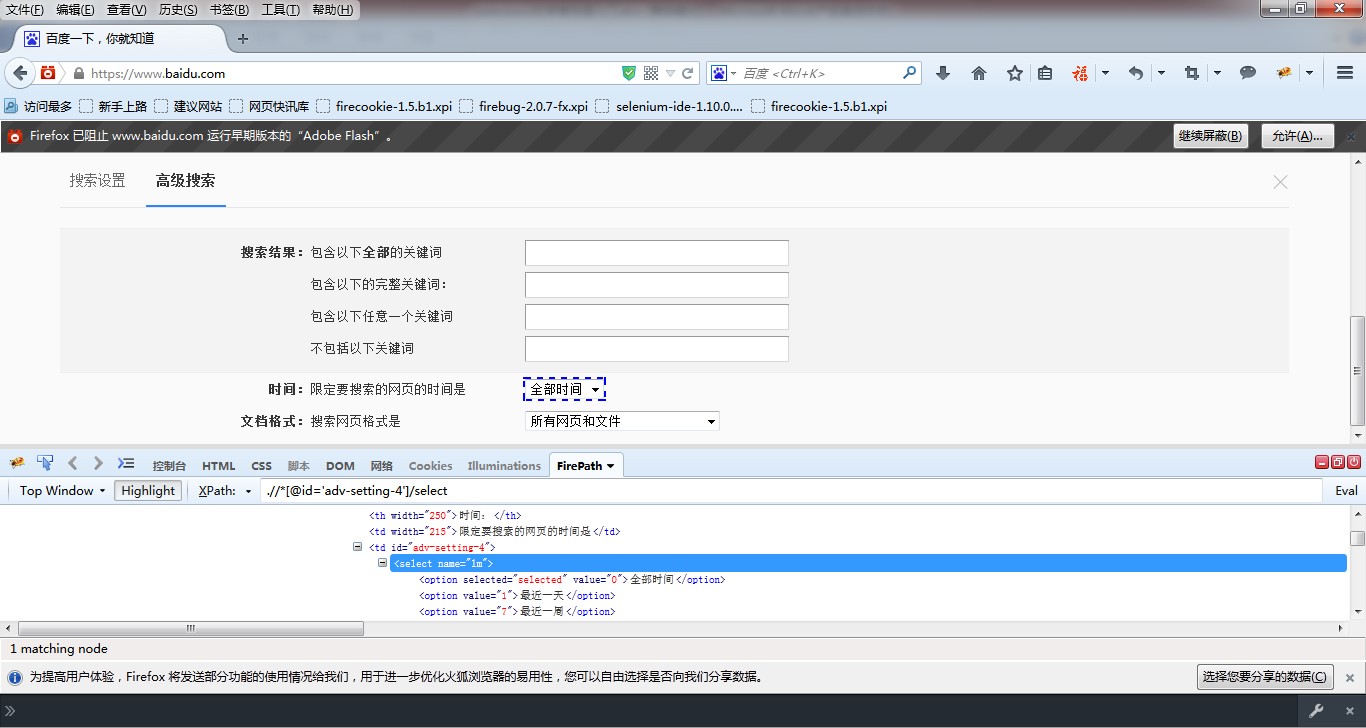


上面的上传图片，代码实现为:

driver.findElement(By.name("file")).sendKeys("C:/Users/shen.jl/Downloads/万达广场.jpg");

## 5.4.9 下拉列表选择

通过 Select 类完成标准的 select 下拉列表选择，通过调用 js 完成非 select 下拉列表选择。

1. select 下拉列表选择。例如百度的设置的高级搜索的时间选择。

Html：

<select name="lm">

<option selected="selected" value="0">全部时间</option>

<option value="1">最近一天</option>

<option value="7">最近一周</option>

<option value="30">最近一月</option>

<option value="360">最近一年</option>

</select

Selenium 代码：

Select seList = new Select(driver.findElement(By.name("lm"))); seList.selectByVisibleText("最近一周");

或者：

Select seList=new Select(driver.findElement(By.name("lm")));

seList.selectByValue("360");//通过value选择

2.非 select 下拉列表框

Html：

<ul name=”o1”>

<li>省份 1</li>

<li>省份 2</li>

...

</ul>

Selenium 脚本：

1）通过 js 实现

WebElement inputs= (WebElement)((JavascriptExecutor)driver).executeScript(

"var tags=document.getElementsByTagName(\"li\");var list = [];

for( var k in tags){"+

"var tag = tags[k];if(tag.innerText== \"省份 2\") {return tag;}}");

inputs.click();

2）通过 java 实现

WebElement select1=driver.findElement(By.name("o1"));

String targetText = "最近一月";

List <WebElement> options = select1.findElements(By.tagName("li")); for (int i = 0; i < options.size(); i++)

{

if (options.get(i).getText().equals(targetText))

{

options.get(i).click();

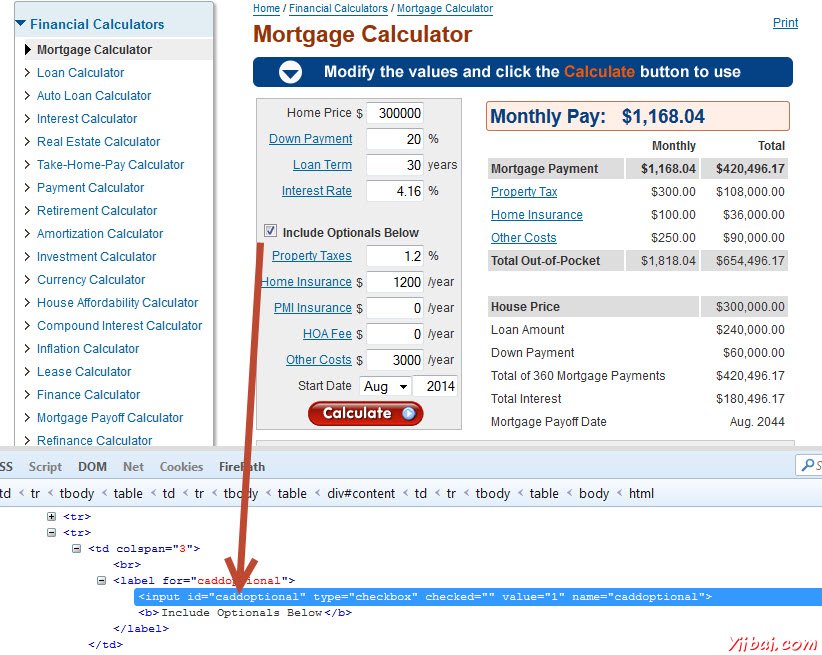
}

}

## 5.5.9 复选框交互

在本节中，我们将了解如何与交互复选框。我们可以使用“click”方法选择入住按钮选项，然后取消选择使用相同的“click”方法。

让我们明白，如何使用交互复选框 - http://www.calculator.net/mortgage-calculator.htmll。我们还可以检查是否选择/启用/可见光的复选框。



import java.util.concurrent.TimeUnit;

import org.openqa.selenium.\*;

import org.openqa.selenium.firefox.FirefoxDriver;

public class webdriverdemo

{

public static void main(String[] args) throws InterruptedException

{

WebDriver driver = new FirefoxDriver();

//Puts a Implicit wait, Will wait for 10 seconds before throwing exception

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

//Launch website

driver.navigate().to("http://www.calculator.net/mortgage-calculator.htmll");

driver.manage().window().maximize();

// Click on check Box

driver.findElement(By.id("caddoptional")).click();

System.out.println("The Output of the IsSelected " + driver.findElement(By.id("caddoptional")).isSelected());

System.out.println("The Output of the IsEnabled " + driver.findElement(By.id("caddoptional")).isEnabled());

System.out.println("The Output of the IsDisplayed " + driver.findElement(By.id("caddoptional")).isDisplayed());

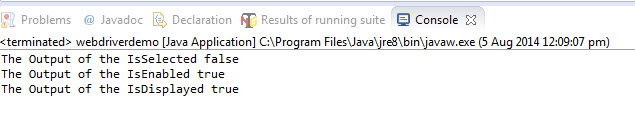
driver.close();

}

}

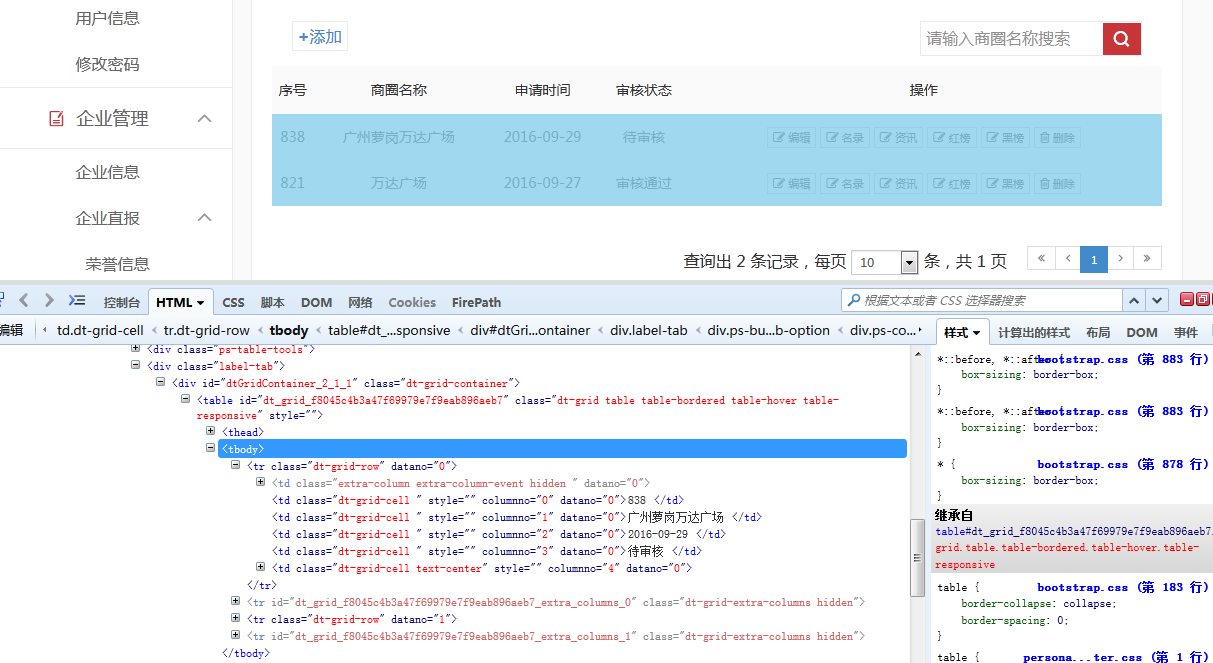
输出

在执行时，该复选框被点击命令后，选中(因为它是默认选中)和命令的输出显示在控制台中。



## 5.5.9 循环取到tbody的值

比如添加操作，添加完成后，需要判断列表中是否含有添加的内容，实现方式如下：



//定位表格

WebElement tbody=driver.findElement(By.xpath("/html/body/div[2]/div/div/div[2]/div/div[2]/div[1]/table/tbody"));

//把行存放到list里面

List<WebElement> trlist=driver.findElements(By.xpath("/html/body/div[2]/div/div/div[2]/div/div[2]/div[1]/table/tbody/tr"));

for(int r=0;r<trlist.size();r++){

String trText=trlist.get(r).getText(); //取得行的text内容

if(trText.contains(name)){ //判断商圈名称是否含有刚才添加的商圈名称，存在则退出循环

System.out.printf("添加的商圈名称为："+name);

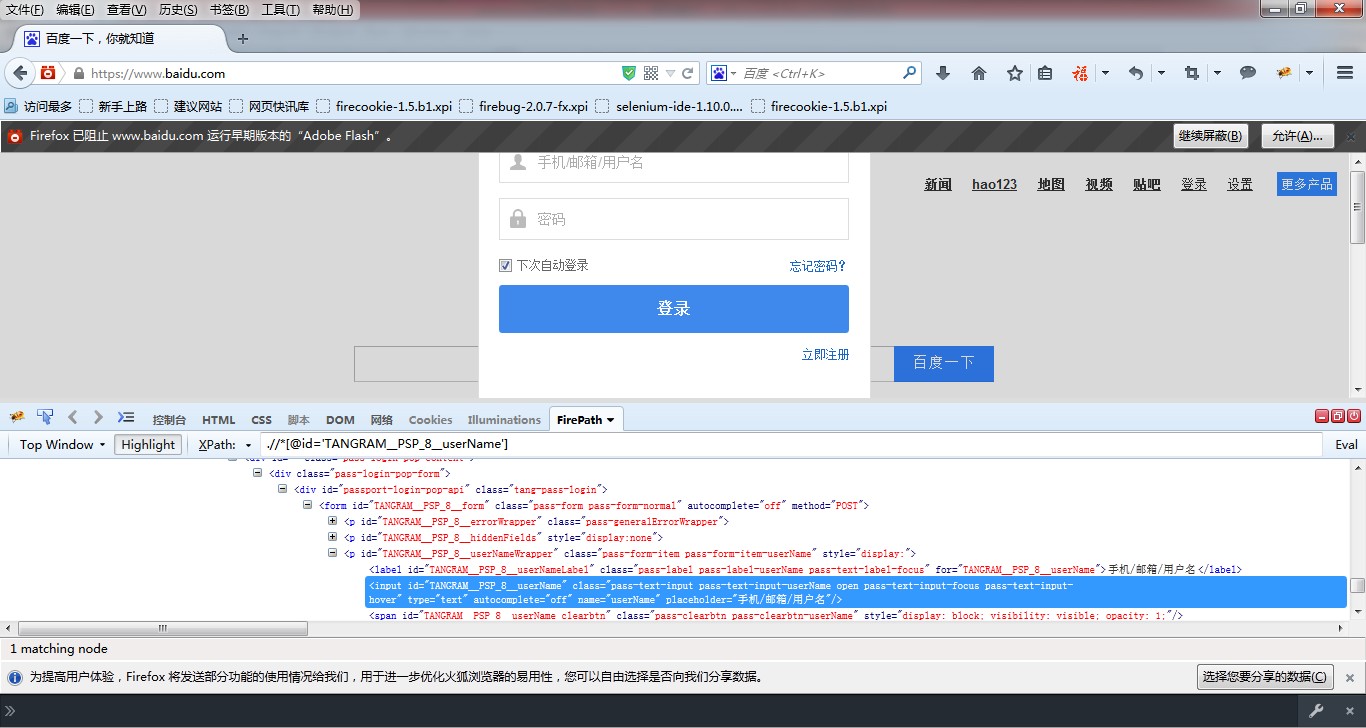
System.out.printf("商圈列表显示为:"+trText+"\n"+"商圈添加成功！");

break;

}

## 5.5.9 Submit()

Submit()方法用于对指定元素所在的 form 元素进行提交操作。例如百度的登录。



Html：

<form id="TANGRAM PSP\_8 form" class="pass-form pass-form-normal" autocomplete="off" method="POST">

<p id="TANGRAM PSP\_8 errorWrapper" class="pass-generalErrorWrapper">

<p id="TANGRAM PSP\_8 hiddenFields" style="display:none">

<p id="TANGRAM PSP\_8 userNameWrapper" class="pass-form-item pass-form-item-userName" style="display:">

<p id="TANGRAM PSP\_8 passwordWrapper" class="pass-form-item pass-form-item-password" style="display:">

<p id="TANGRAM PSP\_8 verifyCodeImgWrapper" class="pass-form-item pass-form-item-verifyCode" style="display:none">

<p id="TANGRAM PSP\_8 memberPassWrapper" class="pass-form-item pass-form-item-memberPass">

<p id="TANGRAM PSP\_8 submitWrapper" class="pass-form-item pass-form-item-submit">

<input id="TANGRAM PSP\_8 submit" class="pass-button pass-button-submit" type="submit" value="登录">

<a class="pass-reglink" target="\_blank" href="https://passport.baidu.com/v2/?reg&tpl=mn&u=https%3A%2F%2Fwww.baidu.com%2F"> 立即注册</a>

<a class="pass-fgtpwd" target="\_blank" href="https://passport.baidu.com/?getpassindex&tpl=mn&u=https%3A%2F%2F[www.baidu.com](http://www.baidu.com/)

%2F">忘记密码？</a>

</p>

</form>

可以看到，它由一个 form 组成，只要对这个 from 中的任何元素使用 Submit()方法，都 会提交这个 form。

Selenium 脚本

driver.findElement(By.id("TANGRAM PSP\_8 password")).submit();

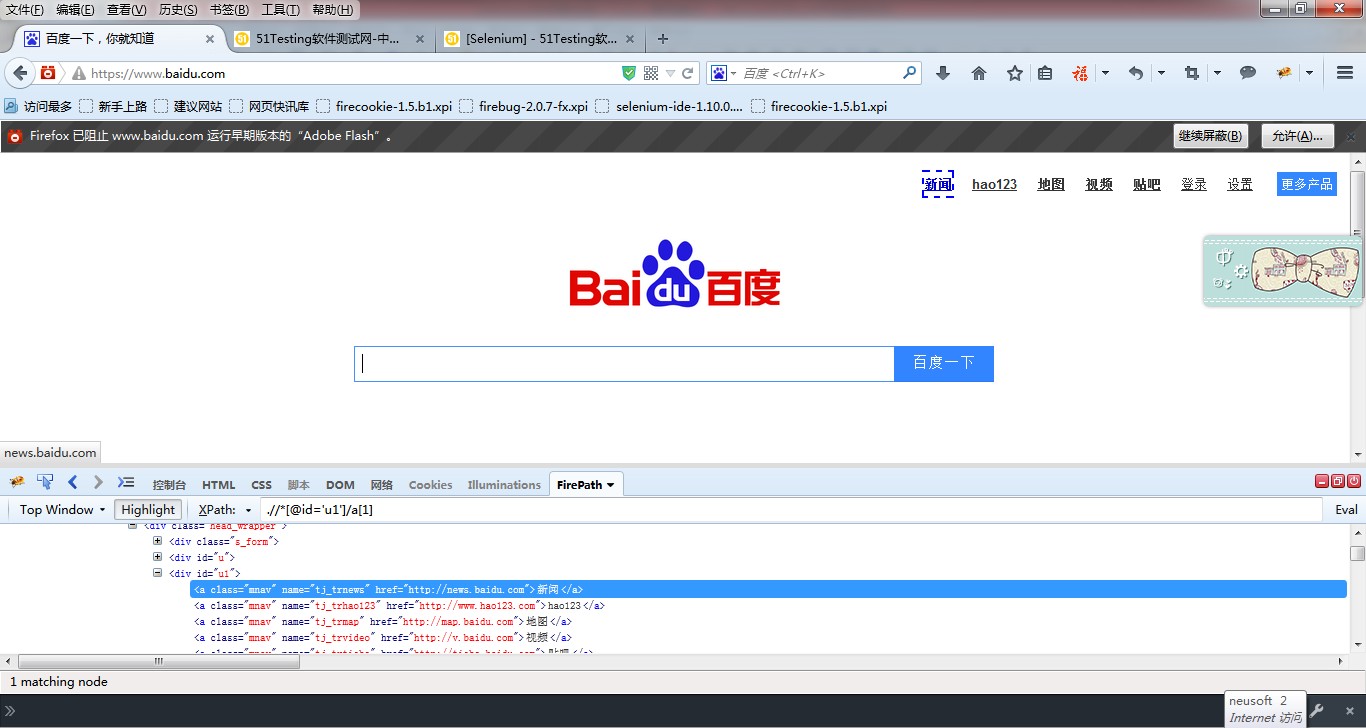
//driver.findElement(By.id("TANGRAM PSP\_8 submit")).click();

## 5.5 获取页面及页面元素的内容

除了对页面元素进行操作，还可以获取它的内容，以比较该其内容是否符合预期。

## getText

getText ( )用于存储某个元素的文本值，例如链接，纯文本等。例如百度的新闻链接。



其 HTML 代码如下：

<a class="mnav" name="tj\_trnews" href="[http://news.baidu.com](http://news.baidu.com/)">新闻</a>

driver.findElement(By.name("tj\_trnews ")).getText()

//driver.findElement(By.name("tj\_trnews ")).getAttribute("innerText")

## getAttribute

getAttribute(String name)：获取元素中名为 name 的属性的值

GetAttribute( )方法用于获取指定属性的值，attributeName 为属性的名称。

而对于页面上输入类型的元素（input，textarea 等），则不能通过 getText()获取到相应的文本， 需要用 getAttribute。例如百度的搜索按钮。



Html：

<input id="su" class="bg s\_btn btnhover" type="submit" value="百度一下"/>

Selenium 代码：

driver.findElement(By.xpath("//input[@id='su']")).getAttribute("value")

## 5.6 窗口处理

selenium 需要对弹出的对话框、窗口进行切换，否则无法识别新窗口的页面元素。

## 5.6.1 弹出对话框

通过 Accept()完成单击弹出对话框的确认按钮，可以同时对 Alert、Confirmation 以及 Prompt

使用。

driver.switchTo().alert().accept();

通过 getText()用于获取弹出对话框的文本内容 driver.switchTo().alert().getText();

## Windows 窗口

在进行 Web 测试时，还会弹出一些子窗口，并且在多个窗口之间进行切换操作 要在多个窗口之间进行切换， 首先必须获取每个窗口的唯一标识符（ 句柄 ）， 通过 getWindowHandles()（用于 Java）可以获取所有打开窗口的标识符，并将其以集合的形式返 回。

新窗口弹出后，可以通过它的标识符（句柄）切换到该窗口，再对该窗口的元素进行操 作。

例如，在百度首页，单击“登录”超级链接

1.窗口切换

1）通过当前窗口标识切换到新窗口

for (String wd : driver.getWindowHandles()) {

if (!wd.endsWith(driver.getWindowHandle())) { driver.switchTo().window(wd);

}

}

要在多个窗口直接切换，首先获取每个窗口的唯一标示符（句柄），通过窗口属性可以获取所有打开窗口的标示符，以集合的形式返回；以下示例：

    Set<String> winHandels = driver.getWindowHandles(); // 得到当前窗口的set集合

    List<String> it = new ArrayList<String>(winHandels); // 将set集合存入list对象

    driver.switchTo().window(it.get(1)); // 切换到弹出的新窗口

    Thread.sleep(1000);

     String url=driver.getCurrentUrl(); //获取新窗口的url

     System.out.println(url);

     driver.switchTo().window(it.get(0)); // 返回至原页面

2）通过窗口标题来切换

for (String wd : driver.getWindowHandles()) {

driver.switchTo().window(wd);

if (driver.getTitle().equals(WINDOWNTITLE)) break;

}

3）通过窗口的唯一标识符（句柄）来切换

String[] handles = new String[driver.getWindowHandles().size()];

driver.getWindowHandles().toArray(handles); for (int i = 0; i < handles.length; i++) {

String loop = String.valueOf(i); if (loop.equals(switchValue)) {

driver.switchTo().window(handles[i]);

}

}

2.关闭窗口

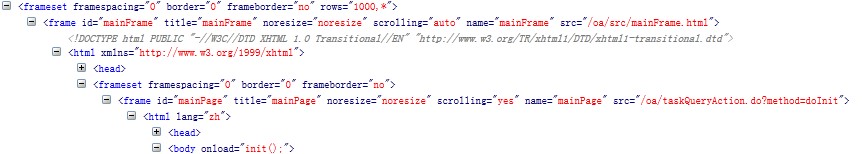
1）通过 driver 关闭当前的浏览器窗口。

driver.close() ; 2）调用关闭浏览器窗口 js 代码

((JavascriptExecutor)driver).executeScript("window.opener = null;window.open('','\_self');window.close();");

## Frame 窗口

在 Selenium 里，是需要先切换到这个 frame 上，才能继续定位里面的元素的。当关闭这 个 frame 之后，需要再切换里到主页面上才能继续运行接下来的语句。



1.切换到指定 frame

1）根据 frame 名称和 ID 切换（frame 或 iframe 的 name 或 Id 属性）

driver.switchTo().frame("mainFrame"); driver.switchTo().frame("mainPage "); 或者

driver.switchTo().frame("mainFrame").switchTo().frame("mainPage").switchTo().frame("ta bIframe").findElement(By.xpath(".//\*[@id='menuTr']")).click();

2）根据 index 来切换

页面中每个层级 frame 的 Index 是从 0 开始，该层级的第一个 frame 为 0，第二个 frame

为 1。一般不建议用 index，如果页面的 frame 多，index 的值不好确定且可维护性差。

driver.switchTo().frame("mainFrame"); driver.switchTo().frame(0);

3）根据 WebElement 的对象来切换

当一个 iFrame 既没有 id，也没有 name 时，可以考虑用 WebElement 方式。

Html：

<iframe frameborder="0" dojoattachpoint="frame" title="Fill Quote" src="https://tssstrpms501.corp.trelleborg.com:12001">

通过以下语句切换：

driver.switchTo().frame(driver.findElement(By.xpath(".//iframe[@title='Fill Quote']")));

2. 切到默认的 content 通过以下语句切换到默认的 content： driver.switchTo().defaultContent();

当已经切换到某个 frame，如果想跳出该 frame 返回到猪页面或者切换到其他的 frame

时，一定需要切换到默认的 content。

## 5.7 检查点

当前数据与期望数据进行比较叫做检查点。

## 5.7.1 判断元素是否存在

判断当前页面中是否存在某个期望查找的元素。 方法代码：

public boolean isElementExsit(WebDriver driver, By locator) {

boolean flag = false;

try {

WebElement element=driver.findElement(locator);

flag=null!=element;

} catch (NoSuchElementException e) {

System.out.println("Element:" + locator.toString()

+ " is not exsit!");

}

return flag;

}

方法调用：

WebDriver driver = new InternetExplorerDriver();

By locator = By.id("id");

isElementExsit(driver,locator);

注： 此处没有考虑页面加载时间等因素。使用时可以酌情增加等待等操作。

## 5.7.2 断言

断言是检查测试数据是否满足预期结果，当检查失败时，会停止测试，常应用在出现重 要的错误需要退出测试的情况。常用 assertEquals 方法

例如：

Assert.assertEquals(true, isElementExsit(driver,locatorSearch));

第一个参数是预期值，第二个参数是实际值。

## 5.7.3 验证页面元素的值

通过获取页面元素的值与预期结果比较。 例如：

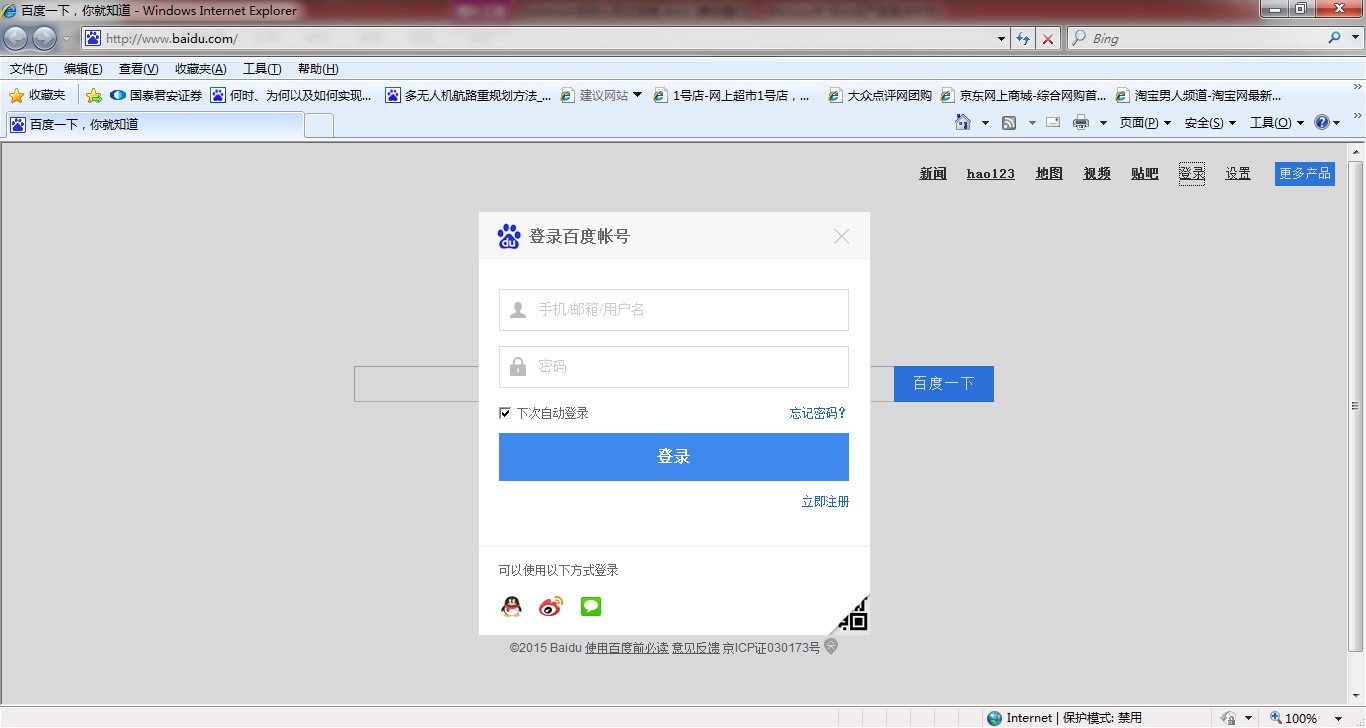
System.out.printf(" 请 您 填 写 手 机 / 邮 箱 / 用 户 名:"+driver.findElement(By.id("TANGRAM PSP\_8 error")).getText().contains(" 请您填写手机/ 邮箱/用户名")+"\n");

# 6 实例

在自动化测试的流程中，需要先设计才能进行正式编码。设计文档一般是测试用例，它 们将作为编写自动化测试代码的依据。

接下来以百度的登陆功能为例，详细说明如何设计自动化测试用例。

## 6.1 登陆功能的用例设计



要测试的功能点：

1. 用户名和密码未输入，直接点击登录，会提示输入用户名

2. 只输入用户名，不输入密码，点击登录，会提示输入密码

3. 输入不存在的用户名和密码，点击登录，会提示帐号不存在

。。。。。。。。。。。。。。。。 输入正确的用户名和密码，点击登录，登陆成功后会显示登陆帐号和天气预报信息。



登录的功能测试用例：

|  |  |  |
| --- | --- | --- |
| 步 骤 序 号 操 作 步 骤 检查点 | | |
| 1 | 打开浏览器，在地址栏输入 [http://www.baidu.com](http://www.baidu.com/)，然 后回车 | 检查是否成功进入百 度 |
| 2 | 点击右上角的登录链接 |  |
| 3 | 不输入任何值，直接点击登录界面中的登录按钮 | 给出提示“请您填写  手机/邮箱/用户名” |
| 4 | 只输入用户名，不输入密码，点击登录 | 给出提示“请您填写  密码” |
| 5 | 输入不存在的用户名和错误的密码，点击登录，会提示  帐号或密码错误 | 给出提示“您输入的  帐号不存在” |
| 6 | 输入正确的用户名和错误的密码，点击登录 | 登陆成功后右上角的  “登录”会显示为登 陆帐号、左上角会显 示天气预报信息 |
| 7 | 点击退出，注销登录用户 | 注销成功后右上角会  从登录帐号变为显示 “登录” |
| 8 | 关闭浏览器 |  |

## 6.2 自动化测试代码

package testpack;

import java.util.concurrent.TimeUnit; import org.junit.Assert;

import org.openqa.selenium.By;

import org.openqa.selenium.NoSuchElementException; import org.openqa.selenium.NoSuchWindowException; import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.htmlunit.HtmlUnitDriver; import org.openqa.selenium.ie.InternetExplorerDriver; import org.openqa.selenium.interactions.Actions; import org.openqa.selenium.remote.DesiredCapabilities;

public class testbaidu {

public static void main(String[] args) {

// TODO Auto-generated method stub

//创建一个 WebDriver 实例

// ie

//System.setProperty("webdriver.ie.driver","D:\\working\\Selenium\\down\\IEDriverServer

\_Win32\_2.44.0\\IEDriverServer.exe"); System.setProperty("webdriver.ie.driver","IEDriverServer.exe"); DesiredCapabilities ieCapabilities = DesiredCapabilities.internetExplorer();

ieCapabilities.setCapability(InternetExplorerDriver.INTRODUCE\_FLAKINESS\_BY\_IGNORING\_SECU RITY\_DOMAINS,true);

WebDriver driver = new InternetExplorerDriver(ieCapabilities);

//脚本在查找元素时的最大等待时间即超时时间 driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

//fixfox

//WebDriver driver=new FirefoxDriver();

/\* chrome

//System.setProperty("webdriver.chrome.driver","D:\\working\\Selenium\\down\\chromed river\_win32\\chromedriver.exe");

System.setProperty("webdriver.chrome.driver","chromedriver.exe"); WebDriver driver = new ChromeDriver();

\*/

//访问百度

//driver.get("[http://www.baidu.com](http://www.baidu.com/)"); driver.navigate().to("[http://www.baidu.com](http://www.baidu.com/)");

//等待 try {

Thread.sleep(10000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block e.printStackTrace();

}

//百度搜索按钮

By locatorSearch = By.id("su");

//验证 百度搜索按钮元素是否出现

Assert.assertEquals(true, isElementExsit(driver,locatorSearch));

//点击右上角的登陆链接 driver.findElement(By.partialLinkText("登")).click();

try {

Thread.sleep(5000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block e.printStackTrace();

}

//切换到新弹出的登陆窗口

for (String wd : driver.getWindowHandles()) {

if (!wd.endsWith(driver.getWindowHandle())) { try {

driver.switchTo().window(wd);

} catch (NoSuchWindowException e) { System.out.print("Window: "

+ " cound not found!"

+ e.fillInStackTrace());

}

}

}

//清空帐号文本框

driver.findElement(By.id("TANGRAM PSP\_8 userName")).clear();

//点击登录

//driver.findElement(By.id("TANGRAM PSP\_8 password")).submit(); driver.findElement(By.id("TANGRAM PSP\_8 submit")).click();

try {

Thread.sleep(5000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block e.printStackTrace();

}

//验证 给出提示“请您填写手机/邮箱/用户名”

System.out.printf(" 请 您 填 写 手 机 / 邮 箱 / 用 户 名:"+driver.findElement(By.id("TANGRAM PSP\_8 error")).getText().contains(" 请您填写手机/ 邮箱/用户名")+"\n");

//清空帐号和密码文本框

driver.findElement(By.id("TANGRAM PSP\_8 userName")).clear(); driver.findElement(By.id("TANGRAM PSP\_8 password")).clear();

//输入用户名

driver.findElement(By.id("TANGRAM PSP\_8 userName")).sendKeys("ddddddd");

try {

Thread.sleep(2000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block e.printStackTrace();

}

//点击登录

// driver.findElement(By.id("TANGRAM PSP\_8 password")).submit(); driver.findElement(By.id("TANGRAM PSP\_8 submit")).click();

try {

Thread.sleep(5000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block e.printStackTrace();

}

//验证 给出提示“请您填写密码”

System.out.printf(" 请 您 填 写 密 码 :"+driver.findElement(By.id("TANGRAM PSP\_8 error")).getText().contains(" 请 您 填 写 密 码 ")+"\n");

//清空帐号和密码文本框

driver.findElement(By.id("TANGRAM PSP\_8 userName")).clear(); driver.findElement(By.id("TANGRAM PSP\_8 password")).clear();

//输入不存在的帐号

driver.findElement(By.id("TANGRAM PSP\_8 userName")).sendKeys("dddddddxxx1");

//输入密码

driver.findElement(By.id("TANGRAM PSP\_8 password")).sendKeys("11111111");

try {

Thread.sleep(2000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block e.printStackTrace();

}

//点击登录

// driver.findElement(By.id("TANGRAM PSP\_8 password")).submit(); driver.findElement(By.id("TANGRAM PSP\_8 submit")).click();

try {

Thread.sleep(5000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block e.printStackTrace();

}

//验证 给出提示“您输入的帐号不存在”

System.out.printf(" 您 输 入 的 帐 号 不 存 在:"+driver.findElement(By.id("TANGRAM PSP\_8 error")).getText().contains("您输入的帐号不 存在")+"\n");

//清空帐号和密码文本框

driver.findElement(By.id("TANGRAM PSP\_8 userName")).clear(); driver.findElement(By.id("TANGRAM PSP\_8 password")).clear();

//输入正确的帐号

driver.findElement(By.id("TANGRAM PSP\_8 userName")).sendKeys("wsryyffs");

//输入正确的密码

driver.findElement(By.id("TANGRAM PSP\_8 password")).sendKeys("487666");

//点击登录

// driver.findElement(By.id("TANGRAM PSP\_8 password")).submit(); driver.findElement(By.id("TANGRAM PSP\_8 submit")).click();

try {

Thread.sleep(10000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block e.printStackTrace();

}

//验证 登陆成功后右上角的“登录”会显示为登陆帐号

System.out.printf(" 登 陆 成 功 - 帐 号 显 示 :"+driver.findElement(By.xpath(".//\*[@id='s\_username\_top']/span[@class='user-name']")).get Text().contains("wsryyffs")+"\n");

//验证 登陆成功后左上角会显示天气预报信息

System.out.printf(" 登 陆 成 功 - 天 气 预 报 城 市 :"+driver.findElement(By.xpath(".//em[@class='show-city-name']")).getText().contains(" 沈阳 ")+"\n");

//鼠标移动到右上角的帐号上 Actions actionm = new Actions(driver);

actionm.moveToElement(driver.findElement(By.xpath(".//\*[@id='s\_username\_top']/span[@class

='user-name']")));// 将鼠标移到 toElement 元素中点

actionm.build().perform();

try {

Thread.sleep(2000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block e.printStackTrace();

}

//选择退出 driver.findElement(By.partialLinkText("退出")).click();

try {

Thread.sleep(8000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block e.printStackTrace();

}

//点击弹出的退出确认的 div 框

driver.findElement(By.xpath(".//\*[@id='tip\_con\_wrap']//a[normalize-space(text())=' 确 定']")).click();

// driver.findElement(By.xpath(".//\*[@id='tip\_con\_wrap']/div[3]/a[3]")).click();

try {

Thread.sleep(8000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block e.printStackTrace();

}

//关闭浏览器 driver.quit();

}

public static boolean isElementExsit(WebDriver driver, By locator) { boolean flag = false;

try {

WebElement element=driver.findElement(locator); flag=null!=element;

} catch (NoSuchElementException e) { System.out.println("Element:" + locator.toString()

+ " is not exsit!");

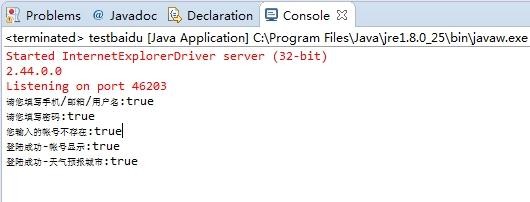
}

return flag;

}

}

## 6.3 执行结果

在 eclipse 控制台上清晰打印测试结果。

# selenium 测试框架

通过以上介绍，会发现 Selenium 只是帮我们解决了执行层面的问题，在大规模的自动化 测试项目中，我们还需要一个基于 Selenium 平台的测试框架，这个框架需要有以下的特性：

 支持数据驱动

 对象库的分离和管理

 操作和检查点的封装

 基于前台灵活配置、后台引擎自动生成脚本，使用者只关心数据、对象和检查点

 基于场景、分布式的、无人值守的自动化测试

 较好的可重用和可维护性

 高效、可靠的元素抓取插件

目前开源的主要是 Selenium+TestNG 的测试框架，但距上面的需求还差很多。 然后再就是各公司基于各自产品特点开发的基于 Selenium 的测试框架，如淘宝自己开发的 基于 selenium 的自动化测试框架。

# selenium 参考网站

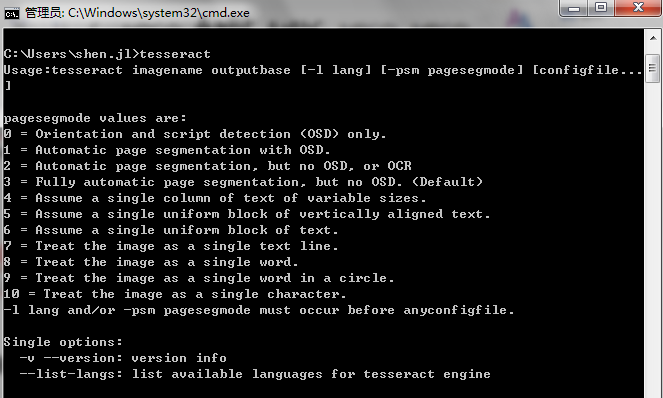
* 1. selenium API <http://selenium.googlecode.com/git/docs/api/java/index.html>（需要翻墙）
  2. selenium 英文论坛

<http://code.google.com/p/selenium/issues/list>（需要翻墙）

# 9 验证码识别工具

## Tesseract-OCR命令行使用

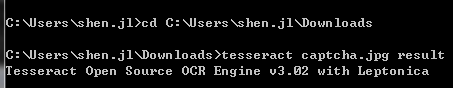
1.打开DOS界面，输入tesseract



如果出现如上输出，表示安装正常。

**2.** 我准备了一张验证码captcha.jpg放在C:\Users\shen.jl\Downloads目录下http://img.blog.csdn.net/20140811163359656?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvd2hhdGRheQ==/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast，简单的执行验证码识别

命令如下：tesseract captcha.jpg result



执行成功后会在验证码图片所在位置生成result.txt 打开结果为:

