# Selenium

实验思路：Selenium IDE安装（Firefox和Chrome插件形式，完成自动化脚本录制）->JDK;Eclipse或者Intellij IDEA安装（供编辑自动化脚本，使用Java语言）-->安装Selenium（导入Jar包或者Marven）-->安装浏览器驱动WebDriver（Chrome，Firefox，IE等） ->在Eclipse或者IntellijIDEA中编辑自动化测试脚本

## 工具准备

1. Selenium官网:https://www.selenium.dev/
2. Selenium IDE: https://www.selenium.dev/downloads/
3. 浏览器驱动

* Firefox浏览器驱动[geckodriver](https://github.com/mozilla/geckodriver/releases)：<https://github.com/mozilla/geckodriver/releases>
* Chrome浏览器驱动[chromedriver](https://sites.google.com/a/chromium.org/chromedriver/home)：<https://npm.taobao.org/mirrors/chromedriver>
* IE浏览器驱动[IEDriverServer](http://selenium-release.storage.googleapis.com/index.html)：

<http://selenium-release.storage.googleapis.com/index.html>

* Edge浏览器驱动[MicrosoftWebDriver](https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver)：<https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/>

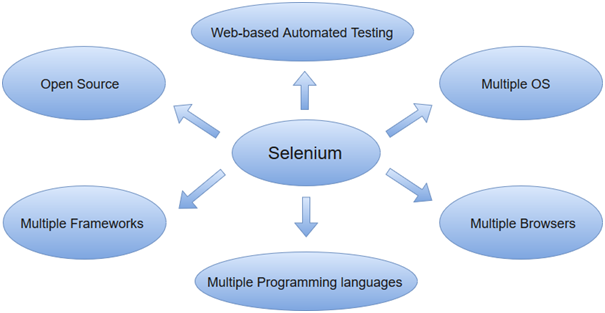
1. JDK安装，Eclipse或者**IntelliJ 安装。**
2. Selenium Java API：<https://seleniumhq.github.io/selenium/docs/api/java/>

## Selenium是什么？

Selenium是最广泛使用的开源Web UI（用户界面）自动化测试套件之一。它最初由Jason Huggins于2004年开发，作为Thought Works的内部工具。 Selenium支持跨不同浏览器，平台和编程语言的自动化。

Selenium可以轻松部署在Windows，Linux，Solaris和Macintosh等平台上。 此外，它支持iOS（iOS，Windows Mobile和Android）等移动应用程序的OS（操作系统）。

Selenium通过使用特定于每种语言的驱动程序支持各种编程语言。Selenium支持的语言包括C#，Java，Perl，PHP，Python和Ruby。目前，Selenium Web驱动程序最受Java和C＃欢迎。 Selenium测试脚本可以使用任何支持的编程语言进行编码，并且可以直接在大多数现代Web浏览器中运行。 Selenium支持的浏览器包括Internet Explorer，Mozilla Firefox，Google Chrome和Safari。



Selenium可用于自动化功能测试，并可与Maven，Jenkins和Docker等自动化测试工具集成，以实现持续测试。 它还可以与TestNG和JUnit等工具集成，以管理测试用例和生成报告。

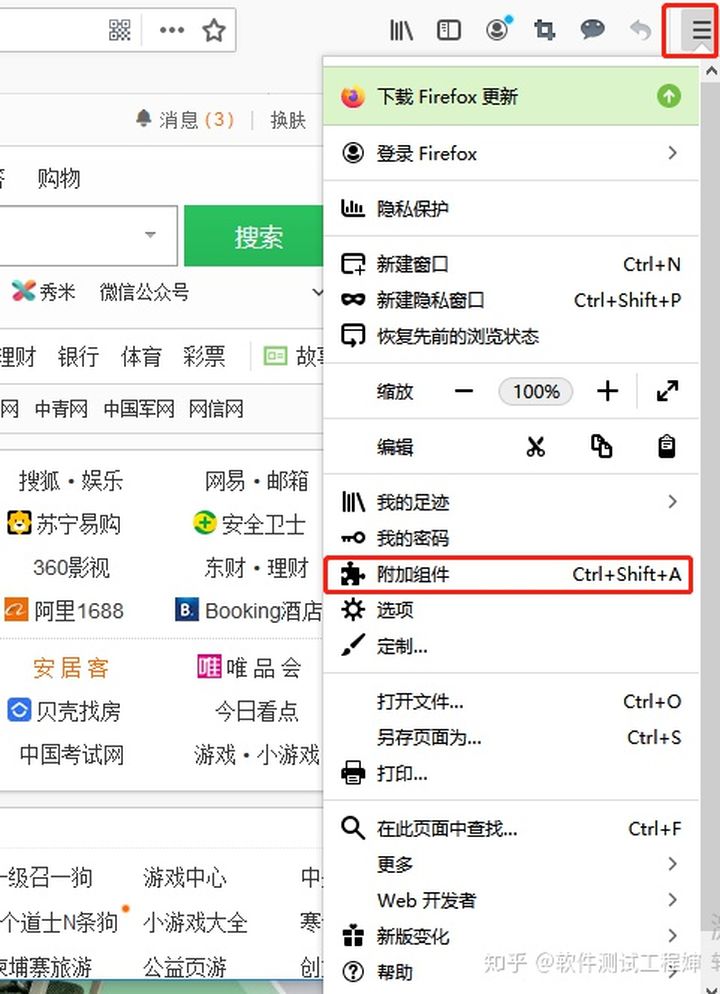
## Selenium IDE使用

虽然自动化测试开发和维护脚本的成本较高，但鉴于它的众多优点，还是有很多项目选择使用自动化测试，尤其在回归测试，验收测试中等。市面上常见的功能自动化测试工具有QTP、Selenium等.QTP功能很强大，不仅可以做Windows应用还可以做Web端应用的自动化测试，但价格昂贵，而且脚本只支持VBS语言,Selenium正好弥补了QTP的不足，开源免费，而且支持三多，大部分企业Web端自动化测试工具都使用的Selenium，可见它的流行程度。支持多种编程语言：Java\Python\C#\Ruby\PHP等；支持多平台：Windows\Linux\Mac等；3、支持多浏览器：Firefox\IE\Chrome\Safari等。

### 1.Firefox Selenium IDE环境配置

IDE是火狐浏览器的插件，要想使用IDE，需要先安装火狐浏览器

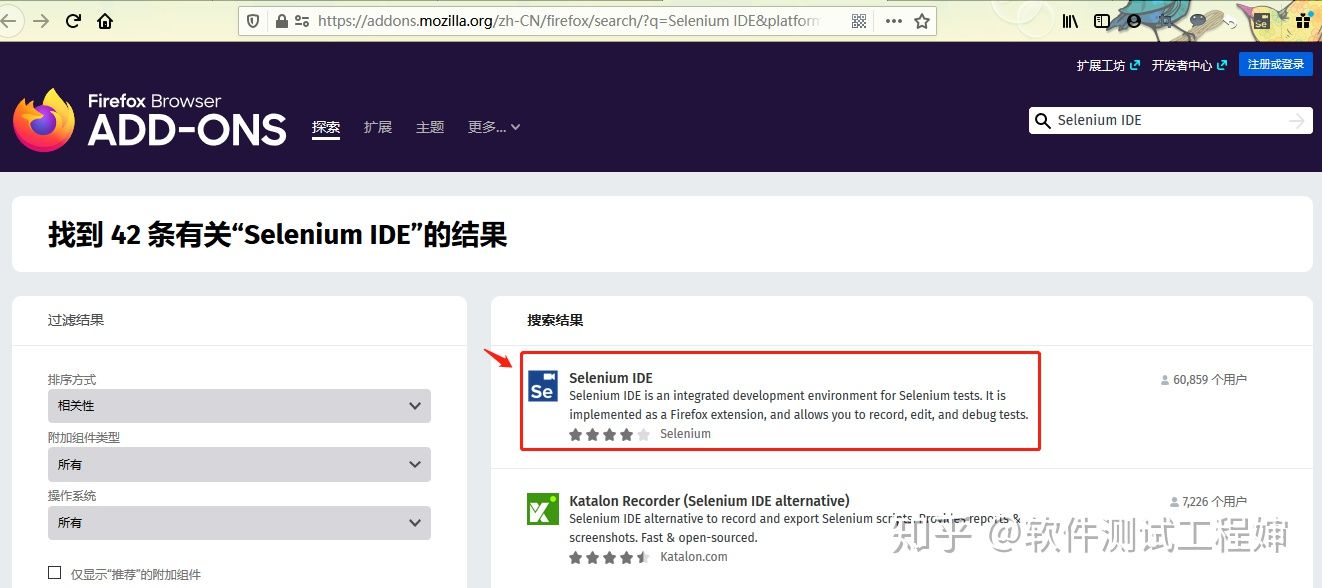
**第一步：**打开火狐浏览器，依次选择“打开菜单”——“附加组件”



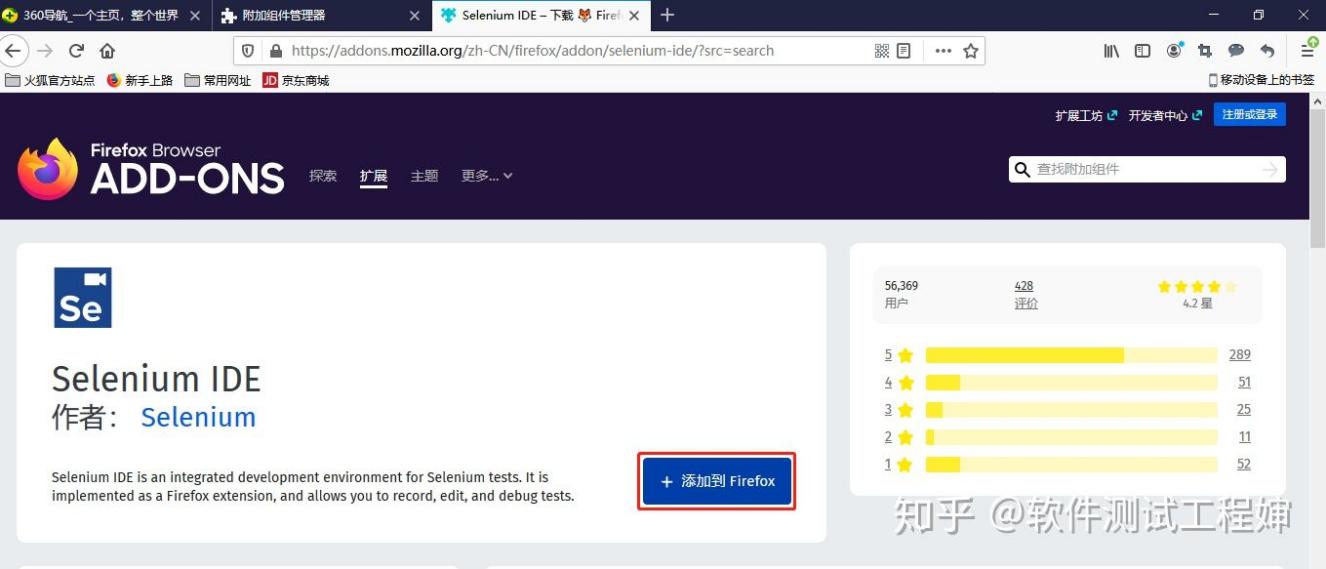
**第二步：**在搜索栏输入“Selenium IDE”回车



**第三步：**点击搜索出来的“Selenium IDE”



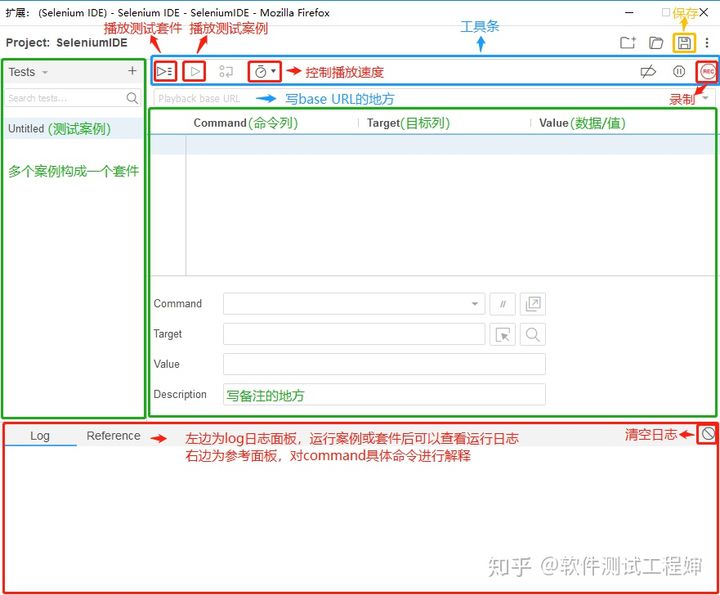
**第四步：**点击“添加到Firefox”按钮



**第五步：**重新启动火狐浏览器，会看到右上角多了一个se图标，表示安装成功



**第六步：**点击se图标，认识一下Selenium IDE的功能界面



之后脚本的录制、编辑、调试、运行都可以使用Selenium IDE来操作了，即使不会写代码，也可以做自动化测试啦

### 2.Chrome浏览器Selenium IDE安装

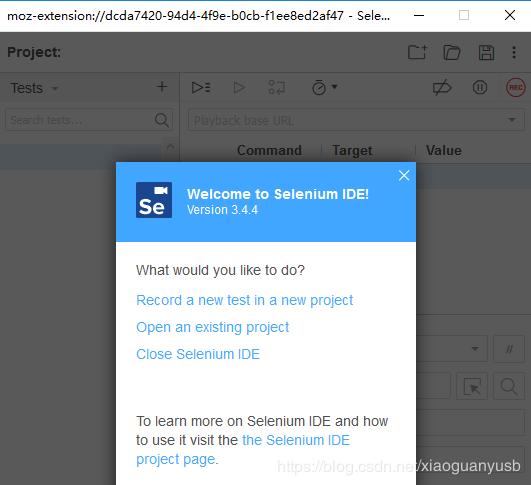
* 打开URL => ：[https://chrome.google.com/webstore/detail/selenium-ide/mooikfkahbdckldjjndioackbalphokd](https://chrome.google.com/webstore/detail/selenium-ide/mooikfkahbdckldjjndioackbalphokd" \t "https://www.pianshen.com/article/4105130534/_blank)

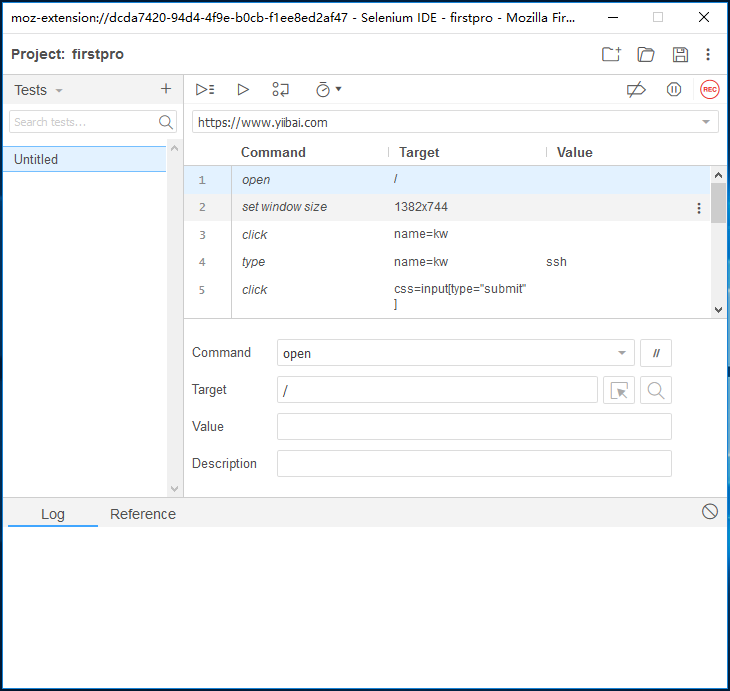


chrome浏览器：

[https://chrome.google.com/webstore/detail/selenium-ide/mooikfkahbdckldjjndioackbalphokd](https://chrome.google.com/webstore/detail/selenium-ide/mooikfkahbdckldjjndioackbalphokd" \t "https://www.pianshen.com/article/4105130534/_blank) 在chrome商城添加selenium IDE。

单击该图标以启动Selenium IDE。

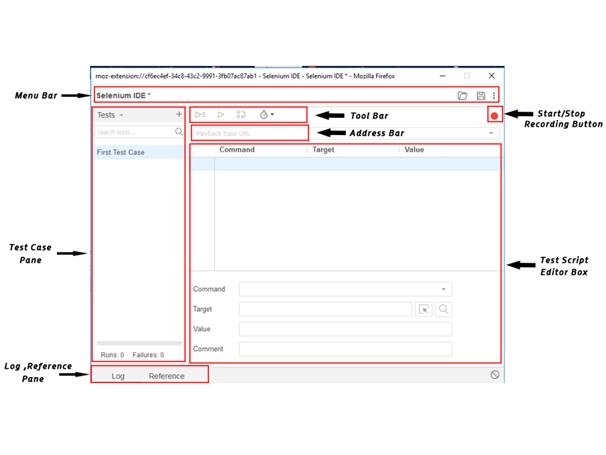




### 3.Selenium IDE功能特性介绍

Selenium IDE分为不同的组件，每个组件都有自己的特性和功能。这里对Selenium IDE的七个不同组件进行了分类，其中包括:

* 菜单栏
* 工具栏
* 地址栏
* 测试案例窗格
* 测试脚本编辑器框
* 开始/停止录制按钮
* 日志，引用窗格



##### 菜单栏

菜单栏位于Selenium IDE界面的最顶部。 最常用的菜单栏模块包括:

* 项目名 - 它用于重命名整个项目。

IMG_263

* **打开项目** - 它用于从个人驱动器加载任何现有项目。

IMG_264

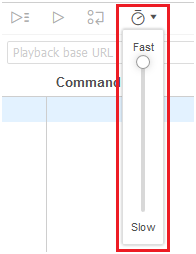
* **保存项目** - 它用于保存当前正在处理的整个项目。

IMG_265

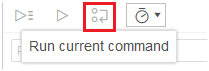
##### 工具栏

工具栏包含用于控制测试用例执行的模块。 此外，它还为提供了调试测试用例的步骤功能。 **Tool Bar** 菜单中最常用的模块包括:

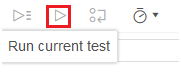
* **速度控制选项** - 它用于控制测试用例的执行速度。



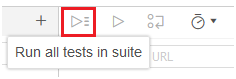
* 步骤功能**- 它用于进入步骤，通过测试用例一次运行一个命令。用于调试测试用例**。



* 运行测试 - 它用于运行当前选定的测试。 只加载一个测试，运行测试， 按钮和运行全部，按钮具有相同的效果。

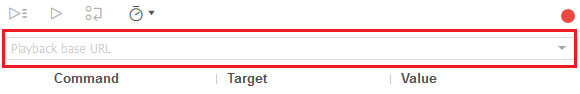


* **全部运行**- 它用于在加载具有多个测试用例的测试套件时运行整个测试套件。



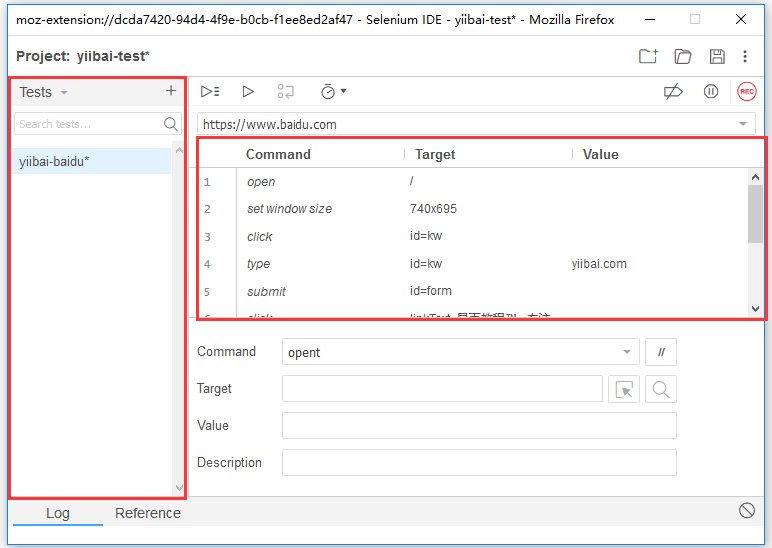
##### 地址栏

此模块提供了一个下拉菜单，可以记住基本URL的所有先前值。 简单来说，基本URL地址栏会记住以前访问过的网站，以便以后导航变得容易。



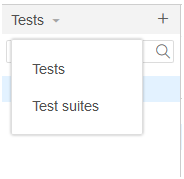
##### 测试用例窗格

此模块包含IDE记录的所有测试用例。 简单来说，它在测试用例窗格下同时提供所有记录的测试用例的列表，以便用户可以轻松地在测试用例之间进行混洗。



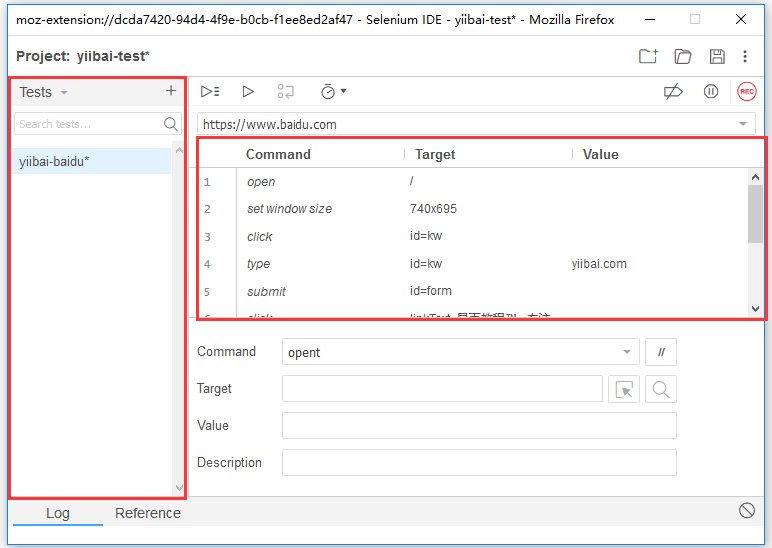
在测试用例窗格的底部，可以看到测试执行结果摘要，其中包括各种测试用例的通过/失败状态。

测试用例窗格还包括导航面板等功能，允许用户在测试用例和测试套件之间导航。

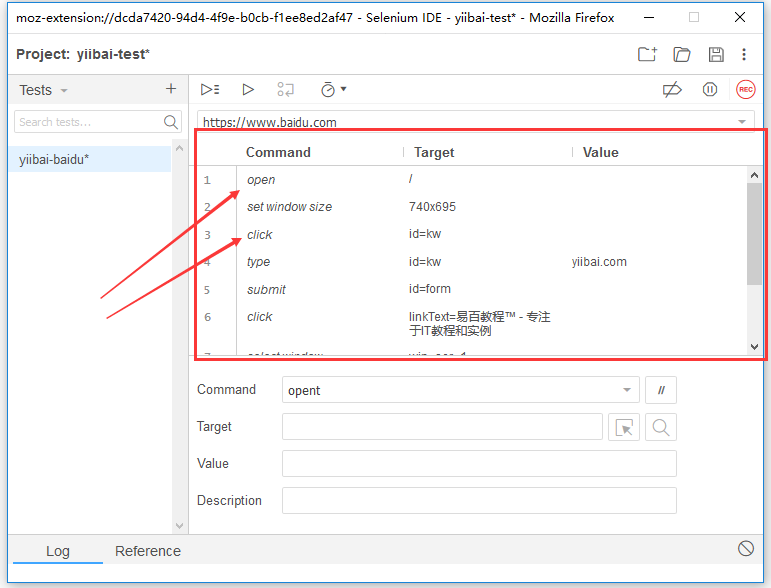


##### 测试脚本编辑器框

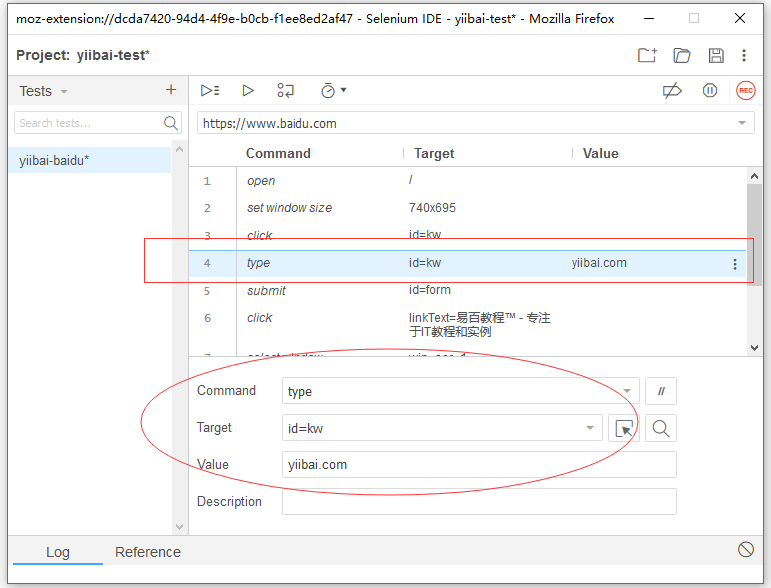
“测试脚本编辑器框”显示IDE记录的所有测试脚本和用户交互。 每个用户交互的显示顺序与执行顺序相同。 “编辑器”框分为三列:命令，目标和值。



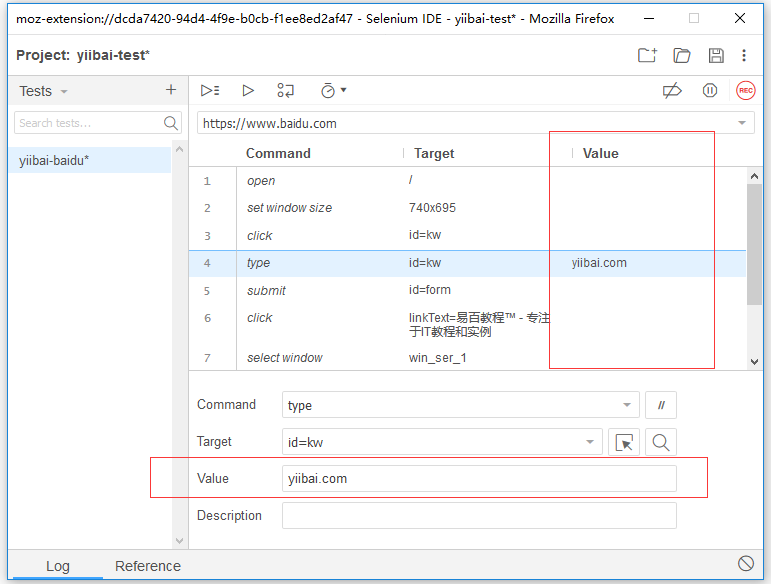
* 命令:
* 可以将命令视为在浏览器元素上执行的实际操作/操作。 例如，如果要打开一个新URL，该命令是:open ,如果单击网页上的链接或按钮，则该命令为:click 。



* 目标:
* Target 指定必须在其上执行操作的web元素以及locator属性。 例如，如果单击名yiibai.com项，那么目标链接是 yiibai.com\* 。

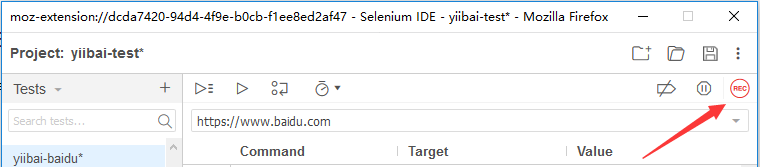


* 值：
* 值被视为可选字段，可在需要发送一些实际参数时使用。 例如，如果在文本框中输入电子邮件地址或密码，则该值将包含实际凭据。



##### 开始/停止录制按钮

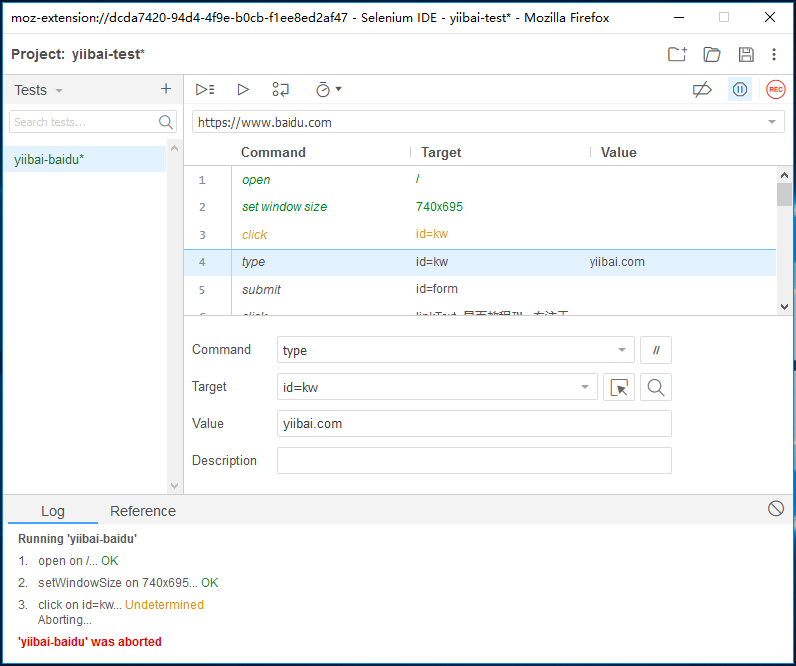
“记录”按钮记录浏览器的所有用户操作。



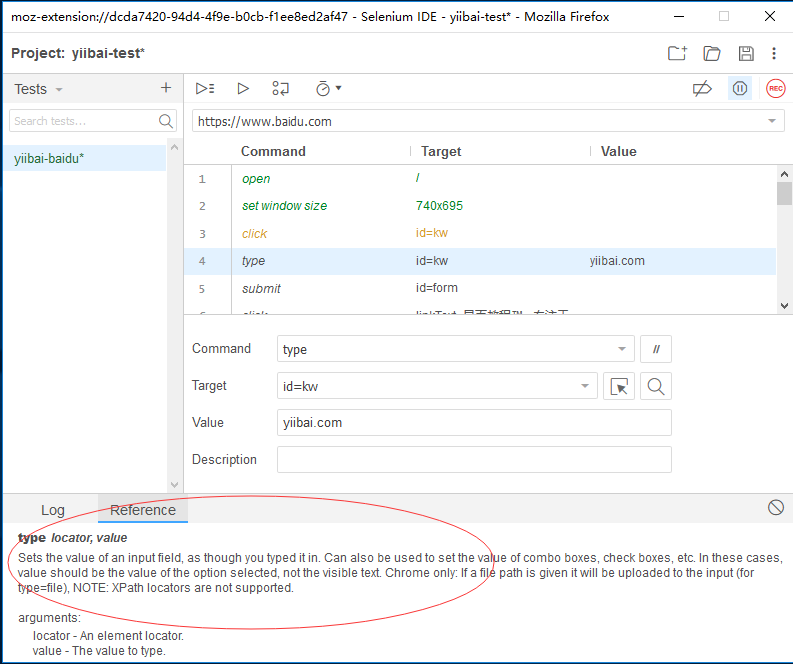
##### 日志，引用窗格

日志窗格在执行期间显示运行时消息。 它提供IDE执行的操作的实时更新。 它可以分为四种类型：信息，错误，调试和警告。

引用窗格在编辑器中显示当前所选selenese命令的完整详细信息。



引用窗格输出如下：



* 点击添加到chrome 
* 添加完成后，在浏览器右上角显示



火狐浏览器最近下掉了firebug和firepath插件，用户即使下载了火狐55以下的版本，也无法查找到这两个插件。

因此，可以用以下方法来获取这两个插件。

1、下载火狐55以内版本安装包，安装时迅速设置禁止自动更新版本，取消勾选自动更新（和之前一样）

54版本火狐浏览器的下载地址：

64位火狐54：[http://ftp.mozilla.org/pub/firefox/releases/54.0.1/win64/zh-CN/](http://ftp.mozilla.org/pub/firefox/releases/54.0.1/win64/zh-CN/" \t "https://www.cnblogs.com/xiaowenshu/p/_blank)

32位火狐54：[http://ftp.mozilla.org/pub/firefox/releases/54.0.1/win32/zh-CN/](http://ftp.mozilla.org/pub/firefox/releases/54.0.1/win32/zh-CN/" \t "https://www.cnblogs.com/xiaowenshu/p/_blank)

2、在火狐随便下载任意一个插件【目的是在配置文件夹里增加一个“extensions”的文件夹】

3、在火狐菜单点击下方小问号，点击“故障排除信息”，新页面的“配置文件夹”那里选择“打开文件夹”。

4、解压我们提供的这个压缩包，看到一个firebug文件夹和两个xpi文件。

压缩包下载：[https://pan.baidu.com/s/1EUrvAJkyIejdfHi0KvB1fQ](https://pan.baidu.com/s/1EUrvAJkyIejdfHi0KvB1fQ" \t "https://www.cnblogs.com/xiaowenshu/p/_blank)

5、将firebug文件夹粘贴到【配置文件夹】里面，两个xpi文件放进【配置文件夹】里面的“extensions”文件夹。

6、在火狐的插件这里把插件选择“启动”

重启一次火狐浏览器，就可以啦！

## Web页面查看器

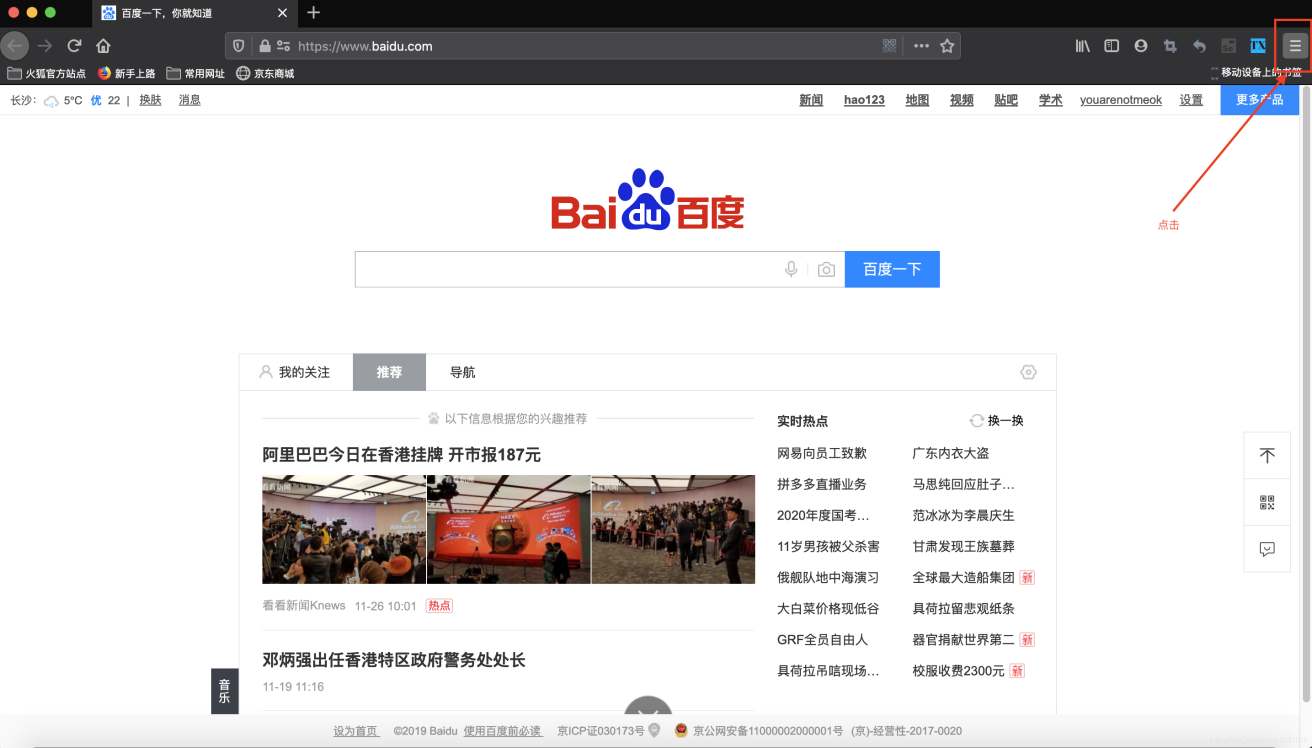
大家都知道，火狐浏览器在48之后就不再支持firebug和firepath插件，但是对于selenium自动化测试来说，准确的定位元素至关重要。失去看这两个组件之后，我们如何定位web端的页面元素呢？这篇文章主要记录了在Firefox v70.0版本上进行元素定位。

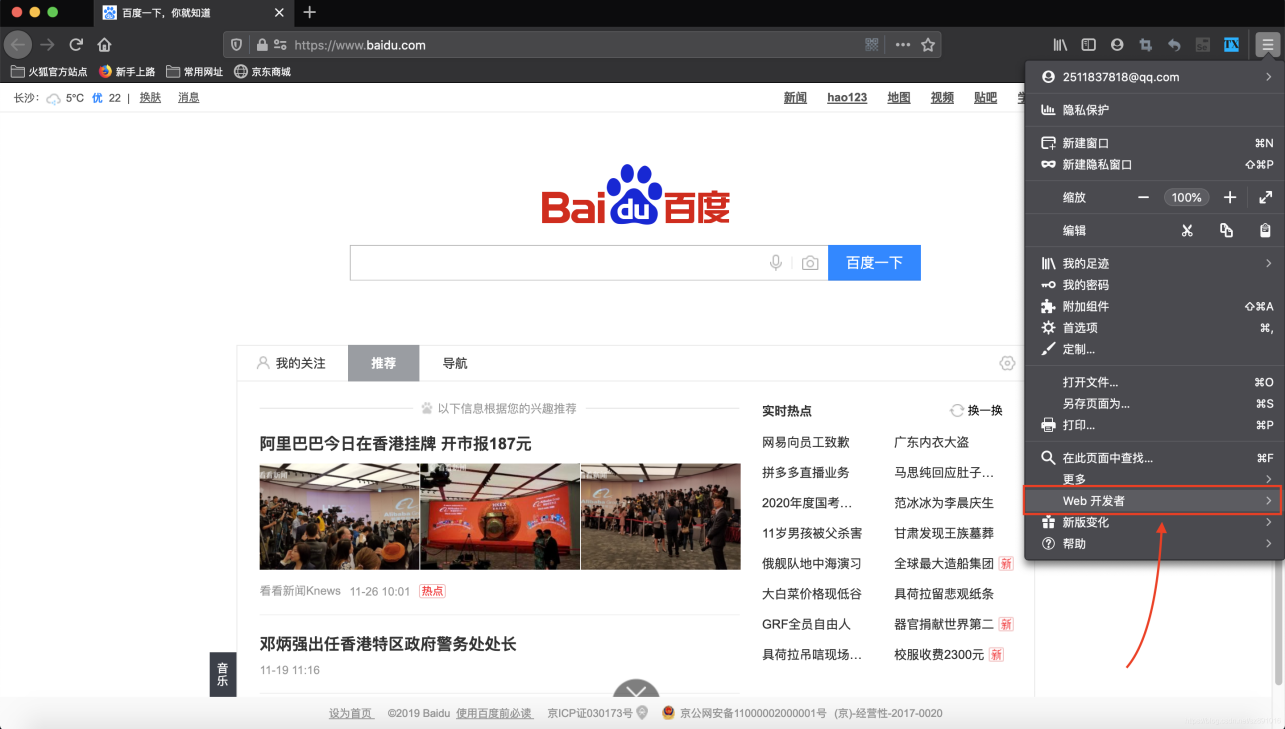
现在，Firefox的firebug和firepath1功能其实都可以在开发者模式下实现。

首先，进入开发者模式。

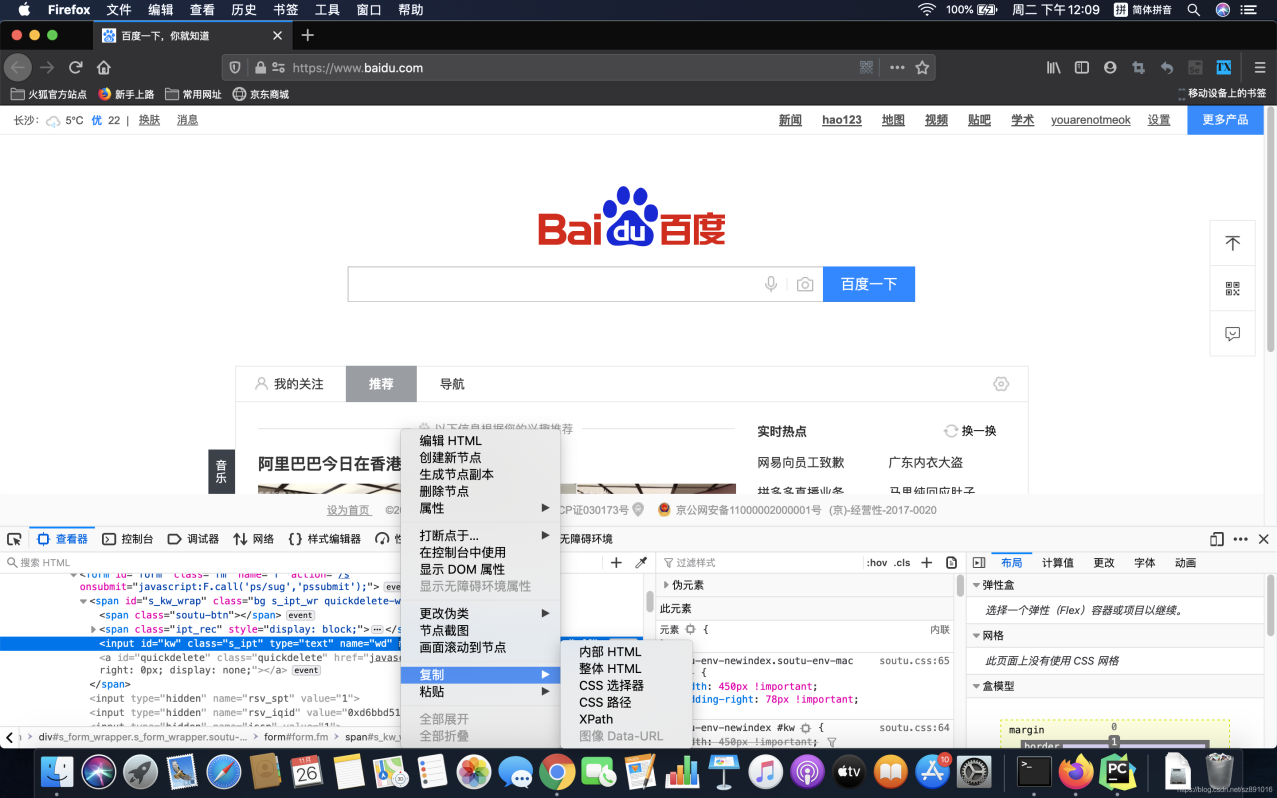
快捷键进入：option+command+i（Mac）

或者，通过以下方式：

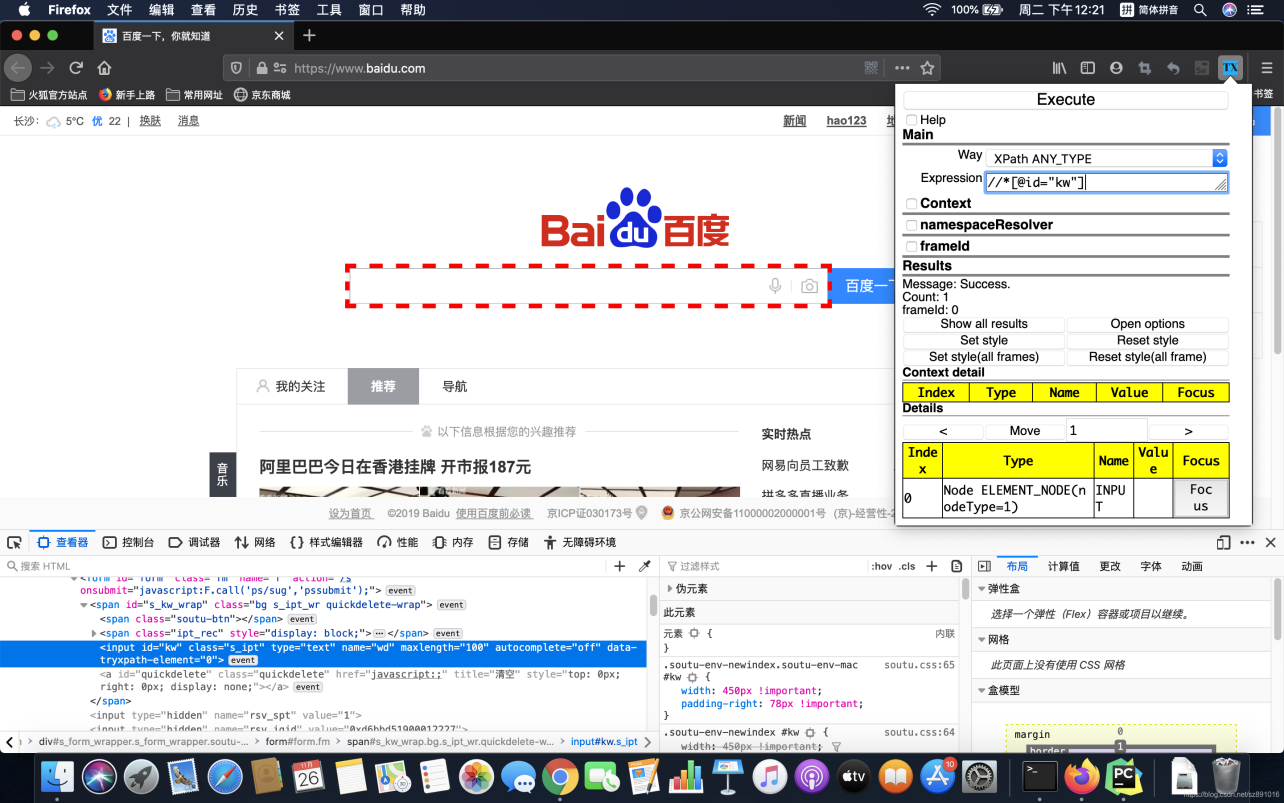




选中你要的元素，右键，复制，然后选择你需要的路径。这里就可以得到xpath了。



如果大家想要进行xpath路径的验证，推荐安装try xpath，Firefox浏览器附加组件里面搜索，然后添加即可。

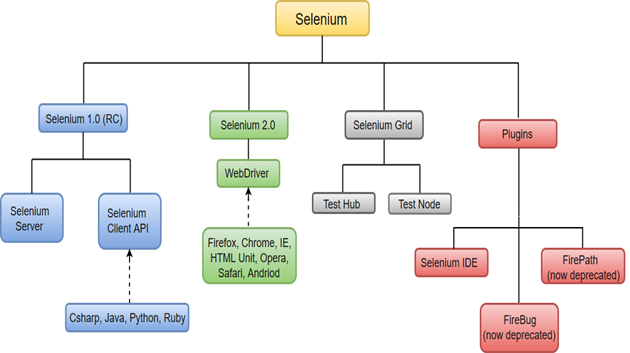


原文链接：https://blog.csdn.net/sz891016/article/details/103253713

## Selenium WebDriver

### 1.WebDriver简介

Selenium WebDriver是Selenium Tool套件中最重要的组件。 最新版本“Selenium 2.0”与WebDriver API集成，提供更简单，更简洁的编程接口。

如下图所示，Selenium组件和测试自动化工具的一些组件关系，让您有一个全面的了解。

Selenium WebDriver最初作为Selenium v2.0的一部分推出。 Selenium的初始版本即Selenium v1仅由IDE，RC和Grid组成。 但是，随着Selenium v3的发布，RC已被弃用并转移到旧版程序包。

在WebDriver中，可以使用任何支持的编程语言开发测试脚本，并且可以在大多数现代Web浏览器中直接运行。WebDriver支持的语言包括C#，Java，Perl，PHP，Python和Ruby。

在学习Selenium WebDriver的概念之前，应该精通至少其中一个支持的编程语言。 目前，Selenium Web驱动程序最受Java和C#欢迎。

与Selenium RC相比，Selenium WebDriver执行速度更快，因为它可以直接调用Web浏览器。 另一方面，RC需要RC服务器与浏览器进行交互，所以相对慢一些。

WebDriver有一个内置的Firefox驱动程序(Gecko Driver)实现。 对于其他浏览器，需要插入其浏览器特定的驱动程序以进行通信并运行测试。最常用的WebDriver如下：

* Google Chrome Driver
* Internet Explorer Driver
* Opera Driver
* Safari Driver
* HTML Unit Driver (一个特殊的无头驱动程序)

### Selenium WebDriver安装

#### （1）下载并安装Java

假设您已在计算机上安装了Java 8或更高版本，并成功配置了运行和编译java程序所需的环境变量。

注意:需要安装Java 8+ 才能使用Selenium 3。

#### 下载和配置Eclipse 或者IntellijIDEA

#### （3）通过Maven安装Selenium

关于Maven安装又是另一个话题了。你可以参考其它资料学习在IntelliJ IDEA创建Maven项目。

[Maven官网](http://maven.apache.org/)

[idea & maven help](https://www.jetbrains.com/help/idea/maven.html)

[Maven仓库](https://mvnrepository.com/)

打开pom.xml 配置Selenium。

**<?**xml version="1.0" encoding="UTF-8"**?>**<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.mvn.demo</groupId>

<artifactId>MyMvnPro</artifactId>

<version>1.0-SNAPSHOT</version>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.6</source>

<target>1.6</target>

</configuration>

</plugin>

</plugins>

</build>

<dependencies>

*<!-- selenium-java -->*

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.4.0</version>

</dependency>

</dependencies>

</project>

#### （4）编写Java版本的HelloSelenium

package javaBase;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class Itest {

public static void main(String[] args) {

WebDriver driver = new ChromeDriver();

driver.get("http://www.itest.info");

String title = driver.getTitle();

System.out.printf(title);

driver.close();

}

}

## Selenium脚本编辑环境安装之Java、IntelliJIDEA、Selenium（选读）

### 安装java

点击 [JDK8下载](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html)，根据自己的平台，选择相应的版本进行下载。

小知识：

Java环境分JDK和JRE ，JDK就是Java Development Kit。简单的说JDK是面向开发人员使用的SDK，它提供了Java的开发

环境和运行环境。JRE是Java Runtime Enviroment是指Java的运行环境，是面向 Java 程序的使用者。

我们以 Windows安装JDK为例，双击下载的JDK，设置安装路径。这里我选择默认安装在 **"D:\Program Files\Java\jdk1.8.0\_101"** 目录下。

下面设置环境变量：

“我的电脑” 右键菜单--->属性--->高级--->环境变量--->系统变量--->新建..

变量名： JAVA\_HOME

变量值： D:\Program Files\Java\jdk1.8.0\_101

变量名： CALSS\_PATH

变量值： .;%JAVA\_HOME%\lib\dt.jar;%JAVA\_HOME%\lib\tools.jar;

找到 path 变量名—>“编辑” 添加：

变量名： PATH

变量值： %JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin;

在Windows命令提示符(cmd)下验证 Java 是否成功：

> java

用法: java [-options] class [args...]

(执行类)

或 java [-options] -jar jarfile [args...]

(执行 jar 文件)

其中选项包括:

-d32 使用 32 位数据模型 (如果可用)

-d64 使用 64 位数据模型 (如果可用)

-client 选择 "client" VM

-server 选择 "server" VM

默认 VM 是 client.

......

> javac

用法: javac <options> <source files>

其中, 可能的选项包括:

-g 生成所有调试信息

-g:none 不生成任何调试信息

-g:{lines,vars,source} 只生成某些调试信息

-nowarn 不生成任何警告

-verbose 输出有关编译器正在执行的操作的消息

-deprecation 输出使用已过时的 API 的源位置

-classpath <路径> 指定查找用户类文件和注释处理程序的位置

.......

* java 命令可以运行 class 文件字节码。
* javac 命令可以将 Java 源文件编译为 class 字节码文件。

可能读者当前下载的 Java 版本与本书不同， 但安装方法是一样的。

### 安装IntelliJ IDEA

你可能会问，为什么不用Eclipse呢？随着发展IntelliJ IDEA有超越Eclipse的势头，JetBrains公司的IDE基本上已经一统了各家主流编程语言的江湖。考虑到 Java IDE的流行趋势，本书中决定选用IntelliJ IDEA。

当然， 选择什么样的IDE充满着个人喜好。你依然可以参考其它资料安装Java IDE。这不会影响你阅读该系列文章。 点击 [IntelliJ IDEA下载](https://www.jetbrains.com/idea)，根据自己的平台，选择相应的版本进行下载。

IntelliJ IDEA安装过程省略...

如果第一次打开IntelliJ IDEA，会看到如下界面。

点击"Create New Project"选项创建新的Java项目。选择项目类型为Java，然后，继续"Next"。

* Project name: 项目名称。
* Project location: 项目在硬盘上的路径。

点击"Finish"结束项目创建完成。

### 编写Hello World！

首先，打开IntelliJ IDEA，点击左侧项目列表，在src下面创建**包**和**类文件**。

1)右键左侧项目列表 src--->New ---> Package 弹出窗口， 输入包的名：javaBase。

2)右键左侧创建的包名：java --->New ---> Java Class 弹出窗口， 输入类的名：HelloWorld。

在 HelloWorld.java 文件中编写第一个 Java 程序。

package com.java.base;

public class HelloWorld {

public static void main(String[] args){

System.out.println("hello world");

}

}

输入完成， 点击工具栏 Run 按钮(或在代码文件中右键选择"Run 'HelloWorld.main()'")运行， 将会在控制台看到“hello word” 的输出。

### 环境安装之Selenium

### 通过jar包安装

点击 [Selenium下载](http://docs.seleniumhq.org/download/) 链接 你会看到Selenium Standalone Server的介绍：

The Selenium Server is needed in order to run Remote Selenium WebDriver. Selenium 3.X is no longer capable of running Selenium RC directly, rather it does it through emulation and the WebDriverBackedSelenium interface.

Download version **3.4.0**

点击版本号进行下载，下载完成将会得到一个 **selenium-server-standalone-3.4.0.jar** 文件。

或者通过 [淘宝镜像](https://npm.taobao.org/mirrors/selenium)。

打开IntelliJ IDEA，导入.jar包。

点击菜单栏 File --> Project Structure（快捷键Ctrl + Alt + Shift + s） ，点击 Project Structure界面左侧的Modules 。在Dependencies 标签界面下，点击右边绿色的“+” 号，选择第一个选项JARs or directories... ，选择相应的 jar 包，点OK ，jar包添加成功。

### 通过Maven安装+

关于Maven安装又是另一个话题了。你可以参考其它资料学习在IntelliJ IDEA创建Maven项目。

[Maven官网](http://maven.apache.org/)

[idea & maven help](https://www.jetbrains.com/help/idea/maven.html)

[Maven仓库](https://mvnrepository.com/)

打开pom.xml 配置Selenium。

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.mvn.demo</groupId>

<artifactId>MyMvnPro</artifactId>

<version>1.0-SNAPSHOT</version>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.6</source>

<target>1.6</target>

</configuration>

</plugin>

</plugins>

</build>

<dependencies>

<!-- selenium-java -->

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.4.0</version>

</dependency>

</dependencies>

</project>

虽然，学习Maven需要增加你的学习成本，但如果你需要长期使用Java编程语言，或者想用Java来做更多事情的话，越早使用Maven越好！因为它会让的第三方包管理变得非常简单。

### Hello Selenium

最后，少不了要写一个简单的Selenium Sample来验证Selenium安装是否成功，打开IntelliJ IDEA 创建一个新类Itest.java

package javaBase;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class Itest {

public static void main(String[] args) {

WebDriver driver = new ChromeDriver();

driver.get("http://www.itest.info");

String title = driver.getTitle();

System.out.printf(title);

driver.close();

}

}

如果执行报错，请看下一节，[Selenium3浏览器驱动](http://www.testclass.net/selenium_java/selenium3-browser-driver/)。

## Selenium3 浏览器驱动

### 下载浏览器驱动

当selenium升级到3.0之后，对不同的浏览器驱动进行了规范。如果想使用selenium驱动不同的浏览器，必须单独下载并设置不同的浏览器驱动。

各浏览器下载地址：（建议只下载Firefox和Chrome驱动，其他驱动有需要自行下载）

Firefox浏览器驱动：[geckodriver](https://github.com/mozilla/geckodriver/releases)

Chrome浏览器驱动：[chromedriver](https://sites.google.com/a/chromium.org/chromedriver/home) [taobao备用地址](https://npm.taobao.org/mirrors/chromedriver)

IE浏览器驱动：[IEDriverServer](http://selenium-release.storage.googleapis.com/index.html)

Edge浏览器驱动：[MicrosoftWebDriver](https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver)

Opera浏览器驱动：[operadriver](https://github.com/operasoftware/operachromiumdriver/releases)

PhantomJS浏览器驱动：[phantomjs](http://phantomjs.org)

### 设置浏览器驱动

设置浏览器的地址非常简单。 我们可以手动创建一个存放浏览器驱动的目录，如： C:\driver , 将下载的浏览器驱动文件（例如：chromedriver、geckodriver）丢到该目录下。

我的电脑-->属性-->系统设置-->高级-->环境变量-->系统变量-->Path，将“C:\driver”目录添加到Path的值中。

* Path
* ;C:\driver

### 测试浏览器驱动

验证不同的浏览器驱动是否正常使用。

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.edge.EdgeDriver;

import org.openqa.selenium.ie.InternetExplorerDriver;

import org.openqa.selenium.opera.OperaDriver;

import org.openqa.selenium.phantomjs.PhantomJSDriver;

……

WebDriver driver = new ChromeDriver(); //Chrome浏览器

WebDriver driver = new FirefoxDriver(); //Firefox浏览器

WebDriver driver = new EdgeDriver(); //Edge浏览器

WebDriver driver = new InternetExplorerDriver(); // Internet Explorer浏览器

WebDriver driver = new OperaDriver(); //Opera浏览器

WebDriver driver = new PhantomJSDriver(); //PhantomJS

练习：请将示例代码在Eclipse或者IEDA中运行，成功后截图到实验报告中。

## Selenium元素定位

Selenium提供了8种定位方式。

* id
* name
* class name
* tag name
* link text
* partial link text
* xpath
* css selector

这8种定位方式在Java selenium中所对应的方法为：

* findElement(By.id())
* findElement(By.name())
* findElement(By.className())
* findElement(By.tagName())
* findElement(By.linkText())
* findElement(By.partialLinkText())
* findElement(By.xpath())
* findElement(By.cssSelector())

### 定位方法的用法

假如我们有一个Web页面，通过前端工具（如，Firebug）查看到一个元素的属性是这样的。

<html>

<head>

<body link="#0000cc">

<a id="result\_logo" href="/" onmousedown="return c({'fm':'tab','tab':'logo'})">

<form id="form" class="fm" name="f" action="/s">

<span class="soutu-btn"></span>

<input id="kw" class="s\_ipt" name="wd" value="" maxlength="255" autocomplete="off">

我们的目的是要定位input标签的输入框。

* 通过id定位:

driver.findElement(By.id("kw"))

* 通过name定位:

driver.findElement(By.name("wd"))

* 通过class name定位:

driver.findElement(By.className("s\_ipt"))

* 通过tag name定位:

driver.findElement(By.tagName("input"))

* 通过xpath定位，xpath定位有N种写法，这里列几个常用写法:

driver.findElement(By.xpath("//\*[@id='kw']"))

driver.findElement(By.xpath("//\*[@name='wd']"))

driver.findElement(By.xpath("//input[@class='s\_ipt']"))

driver.findElement(By.xpath("/html/body/form/span/input"))

driver.findElement(By.xpath("//span[@class='soutu-btn']/input"))

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

driver.findElement(By.xpath("//input[@id='kw' and @name='wd']"))

* 通过css定位，css定位有N种写法，这里列几个常用写法:

driver.findElement(By.cssSelector("#kw")

driver.findElement(By.cssSelector("[name=wd]")

driver.findElement(By.cssSelector(".s\_ipt")

driver.findElement(By.cssSelector("html > body > form > span > input")

driver.findElement(By.cssSelector("span.soutu-btn> input#kw")

driver.findElement(By.cssSelector("form#form > span > input")

接下来，我们的页面上有一组文本链接。

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

<a class="mnav" href="http://www.hao123.com" name="tj\_trhao123">hao123</a>

* 通过link text定位:

driver.findElement(By.linkText("新闻")

driver.findElement(By.linkText("hao123")

* 通过link text定位:

driver.findElement(By.partialLinkText("新")

driver.findElement(By.partialLinkText("hao")

driver.findElement(By.partialLinkText("123")

关于xpath和css的定位比较复杂，请参考： [xpath语法](http://www.w3school.com.cn/xpath/xpath_syntax.asp)、 [css选择器](http://www.w3school.com.cn/cssref/css_selectors.asp)

## Selenium控制浏览器操作

### 控制浏览器窗口大小

有时候我们希望能以某种浏览器尺寸找开，访问的页面在这种尺寸下运行。例如可以将浏览器设置成移动端大小(480\* 800)，然后访问移动站点，对其样式进行评估；WebDriver 提供了 manage().window().setSize()方法来设置浏览器的大小。

* maximize() 设置浏览器最大化
* setSize() 设置浏览器宽高

import org.openqa.selenium.Dimension;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class Browser {

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

WebDriver driver= new ChromeDriver();

driver.get("https://www.baidu.cn");

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

Thread.sleep(2000);

driver.get("https://m.baidu.cn");

driver.manage().window().setSize(new Dimension(480, 800));

Thread.sleep(2000);

driver.quit();

}

}

在 PC 端执行自动化测试脚本大多的情况下是希望浏览器在全屏幕模式下执行， 那么可以使用 maximize()方法使打开的浏览器全屏显示， 其用法与 setSize()相同， 但它不需要任何参数。

练习：请将示例代码在Eclipse或者IEDA中运行，成功后截图到实验报告中。

### 控制浏览器后退、前进

在使用浏览器浏览网页时，浏览器提供了后退和前进按钮，可以方便地在浏览过的网页之间切换，WebDriver也提供了对应的back()和forward()方法来模拟后退和前进按钮。下面通过例子来演示这两个方法的使用。

* back() 模拟浏览器后退按钮
* forward() 模拟浏览器前进按钮

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.By;

public class BrowserGo {

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

WebDriver driver = new ChromeDriver();

//get 到百度首页

driver.get("https://www.baidu.com/");

System.out.printf("now accesss %s \n", driver.getCurrentUrl());

Thread.sleep(2000);

//点击“新闻” 链接

driver.findElement(By.linkText("新闻")).click();

System.out.printf("now accesss %s \n", driver.getCurrentUrl());

Thread.sleep(2000);

//执行浏览器后退

driver.navigate().back();

System.out.printf("back to %s \n", driver.getCurrentUrl());

Thread.sleep(2000);

//执行浏览器前面

driver.navigate().forward();

System.out.printf("forward to %s \n", driver.getCurrentUrl());

Thread.sleep(2000);

driver.quit();

}

}

为了看清脚本的执行过程，下面每操作一步都通过printf()方法来打印当前的URL地址。

练习：请将示例代码在Eclipse或者IEDA中运行，成功后截图到实验报告中

### 刷新页面

有时候需要手动刷新（F5） 页面。

* refresh() 刷新页面（F5）

……

//刷新页面

driver.navigate().refresh();

……

## WebDriver常用方法

前面我们已经学习了定位元素， 定位只是第一步， 定位之后需要对这个元素进行操作， 或单击（按钮） 或 输入（输入框） ， 下面就来认识这些最常用的方法。

### WebDriver 常用方法clear()、sendKeys(\*value)、click()

下面先来认识 WebDriver 中最常用的几个方法：

* clear() 清除文本。
* sendKeys(\*value) 模拟按键输入。
* click() 单击元素

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

public class BaiduDemo {

public static void main(String[] args) {

WebDriver driver = new ChromeDriver();

driver.get("https://www.baidu.com/");

WebElement search\_text = driver.findElement(By.id("kw"));

WebElement search\_button = driver.findElement(By.id("su"));

search\_text.sendKeys("Java");

search\_text.clear();

search\_text.sendKeys("Selenium");

search\_button.click();

driver.quit();

}

}

clear()方法用于清除文本输入框中的内容。

sendKeys()方法模拟键盘向输入框里输入内容。 但是它的作用不仅于此， 我们还可以用它发送键盘按键， 甚至用它来指定上传的文件。

click()方法可以用来单击一个元素，前提是它是可以被单击的对象，它与 sendKeys()方法是Web页面操作中最常用到的两个方法。 其实click()方法不仅仅用于单击一个按钮，它还可以单击任何可以单击的文字/图片链接、复选框、单选框、下拉框等。

练习：请将示例代码在Eclipse或者IEDA中运行，成功后截图到实验报告中

### WebDriver其它常用方法：submit(), getSize()等。

* submit()

submit()方法用于提交表单。 例如，在搜索框输入关键字之后的“回车” 操作， 就可以通过 submit()方法模拟.

……

WebElement search\_text = driver.findElement(By.id("kw"));

search\_text.sendKeys("Selenium");

search\_text.submit();

……

* getSize() 返回元素的尺寸。
* getText() 获取元素的文本。
* getAttribute(name) 获得属性值。
* isDisplayed() 设置该元素是否用户可见。

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

public class BaiduDemo {

public static void main(String[] args) {

WebDriver driver = new ChromeDriver();

driver.get("https://www.baidu.com/");

//获得百度输入框的尺寸

WebElement size = driver.findElement(By.id("kw"));

System.out.println(size.getSize());

//返回百度页面底部备案信息

WebElement text = driver.findElement(By.id("cp"));

System.out.println(text.getText());

//返回元素的属性值， 可以是 id、 name、 type 或元素拥有的其它任意属性

WebElement ty = driver.findElement(By.id("kw"));

System.out.println(ty.getAttribute("type"));

//返回元素的结果是否可见， 返回结果为 True 或 False

WebElement display = driver.findElement(By.id("kw"));

System.out.println(display.isDisplayed());

driver.quit();

}

}

打印结果：

(500, 22)

©2017 Baidu 使用百度前必读 意见反馈 京 ICP 证 030173 号 京公网安备 11000002000001 号

text

True

练习：请将示例代码在Eclipse或者IEDA中运行，成功后截图到实验报告中

## 模拟鼠标操作

### 关于鼠标操的常用方法

通过前面例子了解到，可以使用click()来模拟鼠标的单击操作，现在的Web产品中提供了更丰富的鼠标交互方式， 例如鼠标右击、双击、悬停、甚至是鼠标拖动等功能。在WebDriver中，将这些关于鼠标操作的方法封装在ActionChains类提供。Actions 类提供了鼠标操作的常用方法：

* contextClick() 右击
* clickAndHold() 鼠标点击并控制
* doubleClick() 双击
* dragAndDrop() 拖动
* release() 释放鼠标
* perform() 执行所有Actions中存储的行为

百度首页设置悬停下拉菜单。

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.interactions.Actions;

public class MouseDemo {

public static void main(String[] args) {

WebDriver driver = new ChromeDriver();

driver.get("https://www.baidu.com/");

WebElement search\_setting = driver.findElement(By.linkText("设置"));

Actions action = new Actions(driver);

action.clickAndHold(search\_setting).perform();

driver.quit();

}

}

* import org.openqa.selenium.interactions.Actions;

导入提供鼠标操作的 ActionChains 类

* Actions(driver) 调用Actions()类，将浏览器驱动driver作为参数传入。
* clickAndHold() 方法用于模拟鼠标悬停操作， 在调用时需要指定元素定位。
* perform() 执行所有ActionChains中存储的行为， 可以理解成是对整个操作的提交动作。

练习：请将示例代码在Eclipse或者IEDA中运行，成功后截图到实验报告中

### 关于鼠标操的其它方法

import org.openqa.selenium.interactions.Actions;

……

Actions action = new Actions(driver);

// 鼠标右键点击指定的元素

action.contextClick(driver.findElement(By.id("element"))).perform();

// 鼠标右键点击指定的元素

action.doubleClick(driver.findElement(By.id("element"))).perform();

// 鼠标拖拽动作， 将 source 元素拖放到 target 元素的位置。

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

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

action.dragAndDrop(source,target).perform();

// 释放鼠标

action.release().perform();

## 模拟键盘操作

Keys()类提供了键盘上几乎所有按键的方法。 前面了解到， sendKeys()方法可以用来模拟键盘输入， 除此之 外， 我们还可以用它来输入键盘上的按键， 甚至是组合键， 如 Ctrl+A、 Ctrl+C 等。

import org.openqa.selenium.WebElement;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.By;

import org.openqa.selenium.Keys;

public class Keyboard {

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

WebDriver driver = new ChromeDriver();

driver.get("https://www.baidu.com");

WebElement input = driver.findElement(By.id("kw"));

//输入框输入内容

input.sendKeys("seleniumm");

Thread.sleep(2000);

//删除多输入的一个 m

input.sendKeys(Keys.BACK\_SPACE);

Thread.sleep(2000);

//输入空格键+“教程”

input.sendKeys(Keys.SPACE);

input.sendKeys("教程");

Thread.sleep(2000);

//ctrl+a 全选输入框内容

input.sendKeys(Keys.CONTROL,"a");

Thread.sleep(2000);

//ctrl+x 剪切输入框内容

input.sendKeys(Keys.CONTROL,"x");

Thread.sleep(2000);

//ctrl+v 粘贴内容到输入框

input.sendKeys(Keys.CONTROL,"v");

Thread.sleep(2000);

//通过回车键盘来代替点击操作

input.sendKeys(Keys.ENTER);

Thread.sleep(2000);

driver.quit();

}

}

练习：请将示例代码在Eclipse或者IEDA中运行，成功后截图到实验报告中

需要说明的是，上面的脚本没有什么实际意义，但向我们展示了模拟键盘各种按键与组合键的用法。

* import org.openqa.selenium.Keys;

在使用键盘按键方法前需要先导入 keys 类。

以下为常用的键盘操作：

sendKeys(Keys.BACK\_SPACE) 回格键（BackSpace）

sendKeys(Keys.SPACE) 空格键(Space)

sendKeys(Keys.TAB) 制表键(Tab)

sendKeys(Keys.ESCAPE) 回退键（Esc）

sendKeys(Keys.ENTER) 回车键（Enter）

sendKeys(Keys.CONTROL,'a') 全选（Ctrl+A）

sendKeys(Keys.CONTROL,'c') 复制（Ctrl+C）

sendKeys(Keys.CONTROL,'x') 剪切（Ctrl+X）

sendKeys(Keys.CONTROL,'v') 粘贴（Ctrl+V）

sendKeys(Keys.F1) 键盘 F1

……

sendKeys(Keys.F12) 键盘 F12

## 获取断言信息

不管是在做功能测试还是自动化测试，最后一步需要拿实际结果与预期进行比较。这个比较的称之为**断言**。

我们通常可以通过获取title 、URL和text等信息进行断言。text方法在前面已经讲过，它用于获取标签对之间的文本信息。

* getTitle()： 用于获得当前页面的title。
* getCurrentUrl() ： 用户获得当前页面的URL。
* getText() 获取页面文本信息。

下面同样以百度为例，介绍如何获取这些信息。

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;

public class AssertDemo {

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

WebDriver driver = new ChromeDriver();

driver.get("https://www.baidu.com");

System.out.println("Search before================");

//获取当前的 title 和 url

System.out.printf("title of current page is %s\n", driver.getTitle());

System.out.printf("url of current page is %s\n", driver.getCurrentUrl());

//百度搜索

WebElement search = driver.findElement(By.id("kw"));

search.sendKeys("Selenium");

search.sendKeys(Keys.ENTER);

Thread.sleep(2000);

System.out.println("Search after================");

//获取当前的 title 和 url

System.out.printf("title of current page is %s\n", driver.getTitle());

System.out.printf("url of current page is %s\n", driver.getCurrentUrl());

//获取第一条搜索结果的标题

WebElement result = driver.findElement(By.xpath("//div[@id='content\_left']/div/h3/a"));

System.out.println(result.getText());

driver.quit();

}

}

练习：请将示例代码在Eclipse或者IEDA中运行，成功后截图到实验报告中

打印结果：

Search before================

title of current page is 百度一下， 你就知道

url of current page is https://www.baidu.com/

Search after================

title of current page is Selenium\_百度搜索

url of current page is

https://www.baidu.com/s?ie=utf-8&f=8&rsv\_bp=0&rsv\_idx=1&tn=baidu&wd=Selenium&rsv\_pq=9be

4680700a485c1&rsv\_t=e925U%2F%2B9SBTqmRI%2BuARg0%2BTCzrrZWn4jOBJkb1OS2vUjMrZsq5VblQ7toD8

&rqlang=cn&rsv\_enter=1&rsv\_sug3=8&rsv\_sug2=0&inputT=155&rsv\_sug4=155

Selenium - Web Browser Automation

## 设置元素等待

WebDriver提供了两种类型的等待：**显式等待**和**隐式等待**。

### 显式等待

WebDriver提供了显式等待方法，专门针对某个元素进行等待判断。

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

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

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

public class TimeOut01 {

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

WebDriver driver = new ChromeDriver();

driver.get("https://www.baidu.com");

//显式等待， 针对某个元素等待

WebDriverWait wait = new WebDriverWait(driver,10,1);

wait.until(new ExpectedCondition<WebElement>(){

@Override

public WebElement apply(WebDriver text) {

return text.findElement(By.id("kw"));

}

}).sendKeys("selenium");

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

Thread.sleep(2000);

driver.quit();

}

}

WebDriverWait类是由WebDirver提供的等待方法。在设置时间内，默认每隔一段时间检测一次当前页面元素是否存在，如果超过设置时间检测不到则抛出异常。具体格式如下：

WebDriverWait(driver, 10, 1)

driver： 浏览器驱动。 10： 最长超时时间， 默认以秒为单位。 1： 检测的的间隔（步长） 时间， 默认为 0.5s。

练习：请将示例代码在Eclipse或者IEDA中运行，成功后截图到实验报告中

### 隐式等待

WebDriver 提供了几种方法来等待元素。

* 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会等页面加载完毕再进行后面的操作，所以如果页面超过设置时间依然没有加载完成，那么WebDriver就会抛出异常。

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.By;

import java.util.concurrent.TimeUnit;

public class TimeOut02 {

public static void main(String[] args){

WebDriver driver = new ChromeDriver();

//页面加载超时时间设置为 5s

driver.manage().timeouts().pageLoadTimeout(5, TimeUnit.SECONDS);

driver.get("https://www.baidu.com/");

//定位对象时给 10s 的时间, 如果 10s 内还定位不到则抛出异常

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

driver.findElement(By.id("kw")).sendKeys("selenium");

//异步脚本的超时时间设置成 3s

driver.manage().timeouts().setScriptTimeout(3, TimeUnit.SECONDS);

driver.quit();

}

}

练习：请将示例代码在Eclipse或者IEDA中运行，成功后截图到实验报告中

## 定位一组元素

我们已经学习了8种定位方法， 那8种定位方法是针对单个元素定位的， WebDriver还提供了另外8种用于定位一组元素的方法。

import org.openqa.selenium.By;

......

findElements(By.id())

findElements(By.name())

findElements(By.className())

findElements(By.tagName())

findElements(By.linkText())

findElements(By.partialLinkText())

findElements(By.xpath())

findElements(By.cssSelector())

定位一组元素的方法与定位单个元素的方法类似，唯一的区别是在单词 findElement 后面多了一个 **s** 表示复数。

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import java.util.List;

public class ElementsDemo {

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

WebDriver driver = new ChromeDriver();

driver.get("https://www.baidu.com/");

WebElement search\_text = driver.findElement(By.id("kw"));

search\_text.sendKeys("selenium");

search\_text.submit();

Thread.sleep(2000);

//匹配第一页搜索结果的标题， 循环打印

List<WebElement> search\_result = driver.findElements(By.xpath("//div/div/h3"));

//打印元素的个数

System.out.println(search\_result.size());

// 循环打印搜索结果的标题

for(WebElement result : search\_result){

System.out.println(result.getText());

}

System.out.println("-------我是分割线---------");

//打印第n结果的标题

WebElement text = search\_result.get(search\_result.size() - 10);

System.out.println(text.getText());

driver.quit();

}

}

打印结果：

15

selenium java 教程-90 天从入门到高薪「学习必看」

python selenium 视频-90 天从入门到高薪「学习必看」

Selenium - Web Browser Automation

功能自动化测试工具——Selenium 篇

Selenium Documentation — Selenium Documentation

selenium + python 自动化测试环境搭建 - 虫师 - 博客园

selenium\_百度翻译

Selenium\_百度百科

怎样开始用 selenium 进行自动化测试(个人总结)\_百度经验

Selenium 官网教程\_selenium 自动化测试实践\_Selenium\_领测软件测试网

Selenium - 开源中国社区

selenium 是什么?\_百度知道

selenium-0 基础入学， 先就业后付款!

selenium， 亚马逊官网， 正品低价， 货到付款!

selenium java 教程-90 天从入门到高薪「学习必看」

-------我是分割线---------

selenium + python 自动化测试环境搭建 - 虫师 - 博客园

练习：请将示例代码在Eclipse或者IEDA中运行，成功后截图到实验报告中

## 多表单切换

在 Web 应用中经常会遇到 frame/iframe 表单嵌套页面的应用， WebDriver 只能在一个页面上对元素识别与 定位， 对于 frame/iframe 表单内嵌页面上的元素无法直接定位。 这时就需要通过 switchTo().frame()方法将当前定 位的主体切换为 frame/iframe 表单的内嵌页面中。

<html>

<body>

...

<iframe id="x-URS-iframe" ...>

<html>

<body>

...

<input name="email" >

126邮箱登录框的结构大概是这样子的，想要操作登录框必须要先切换到iframe表单。

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

public class MailLogin {

public static void main(String[] args){

WebDriver driver = new ChromeDriver();

driver.get("http://www.126.com");

WebElement xf = driver.findElement(By.xpath("//\*[@id='loginDiv']/iframe"));

driver.switchTo().frame(xf);

driver.findElement(By.name("email")).clear();

driver.findElement(By.name("email")).sendKeys("username");

driver.findElement(By.name("password")).clear();

driver.findElement(By.name("password")).sendKeys("password");

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

driver.switchTo().defaultContent();

//……

}

}

如果完成了在当前表单上的操作，则可以通过switchTo().defaultContent()方法跳出表单。

练习：请将示例代码在Eclipse或者IEDA中运行，成功后截图到实验报告中

## 多窗口切换

在页面操作过程中有时候点击某个链接会弹出新的窗口， 这时就需要主机切换到新打开的窗口上进行操作。WebDriver提供了switchTo().window()方法可以实现在不同的窗口之间切换。

以百度首页和百度注册页为例，在两个窗口之间的切换如下图。

实现窗口切换的代码如下：

import java.util.Set;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class MoreWindows {

public static void main(String[] arge) throws InterruptedException{

WebDriver driver = new ChromeDriver();

driver.get("https://www.baidu.com");

//获得当前窗口句柄

String search\_handle = driver.getWindowHandle();

//打开百度注册窗口

driver.findElement(By.linkText("登录")).click();

Thread.sleep(3000);

driver.findElement(By.linkText("立即注册")).click();

//获得所有窗口句柄

Set<String> handles = driver.getWindowHandles();

//判断是否为注册窗口， 并操作注册窗口上的元素

for(String handle : handles){

if (handle.equals(search\_handle)==false){

//切换到注册页面

driver.switchTo().window(handle);

System.out.println("now register window!");

Thread.sleep(2000);

driver.findElement(By.name("userName")).clear();

driver.findElement(By.name("userName")).sendKeys("user name");

driver.findElement(By.name("phone")).clear();

driver.findElement(By.name("phone")).sendKeys("phone number");

//......

Thread.sleep(2000);

//关闭当前窗口

driver.close();

}

}

Thread.sleep(2000);

driver.quit();

}

}

在本例中所涉及的新方法如下：

* getWindowHandle()： 获得当前窗口句柄。
* getWindowHandles()： 返回的所有窗口的句柄到当前会话。
* switchTo().window()： 用于切换到相应的窗口，与上一节的switchTo().frame()类似，前者用于不同窗口的切换， 后者用于不同表单之间的切换。

练习：请将示例代码在Eclipse或者IEDA中运行，成功后截图到实验报告中

## 下拉框选择

有时我们会碰到下拉框，WebDriver提供了Select类来处理下接框。

如百度搜索设置的下拉框，如下图： 搜索下拉框实现代码如下：

<select id="nr" name="NR">

<option value="10" selected>每页显示 10 条</option>

<option value="20">每页显示 20 条</option>

<option value="50">每页显示 50 条</option>

<select>

操作下接框代码如下：

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.support.ui.Select;

public class SelectDemo {

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

WebDriver driver = new ChromeDriver();

driver.get("https://www.baidu.com");

driver.findElement(By.linkText("设置")).click();

driver.findElement(By.linkText("搜索设置")).click();

Thread.sleep(2000);

//<select>标签的下拉框选择

WebElement el = driver.findElement(By.xpath("//select"));

Select sel = new Select(el);

sel.selectByValue("20");

Thread.sleep(2000);

driver.quit();

}

}

Select类用于定位select标签。 selectByValue()方法符用于选取<**option**>标签的value值。

练习：请将示例代码在Eclipse或者IEDA中运行，成功后截图到实验报告中

## 警告框处理

在 WebDriver中处理JavaScript所生成的alert、confirm以及prompt十分简单，具体做法是使用switch\_to\_alert()方法定位到alert/confirm/prompt，然后使用text/accept/dismiss/sendKeys等方法进行操作。

* getText()： 返回 alert/confirm/prompt 中的文字信息。
* accept()： 接受现有警告框。
* dismiss()： 解散现有警告框。
* sendKeys(keysToSend)： 发送文本至警告框。
* keysToSend： 将文本发送至警告框。

如下图，百度搜索设置弹出的窗口是不能通过前端工具对其进行定位的，这个时候就可以通过switchTo().alert()方法接受这个弹窗。

接受一个警告框的代码如下：

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class AlertDemo {

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

WebDriver driver = new ChromeDriver();

driver.get("https://www.baidu.com");

driver.findElement(By.linkText("设置")).click();

driver.findElement(By.linkText("搜索设置")).click();

Thread.sleep(2000);

//保存设置

driver.findElement(By.className("prefpanelgo")).click();

//接收弹窗

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

Thread.sleep(2000);

driver.quit();

}

}

## 文件上传（选做）

对于通过input标签实现的上传功能，可以将其看作是一个输入框，即通过sendKeys()指定本地文件路径的方式实现文件上传。

创建upfile.html文件，代码如下：

<html>

<head>

<meta http-equiv="content-type" content="text/html;charset=utf-8" />

<title>upload\_file</title>

<link href="http://cdn.bootcss.com/bootstrap/3.3.0/css/bootstrap.min.css" rel="stylesheet" />

</head>

<body>

<div class="row-fluid">

<div class="span6 well">

<h3>upload\_file</h3>

<input type="file" name="file" />

</div>

</div>

</body>

<script src="http://cdn.bootcss.com/bootstrap/3.3.0/css/bootstrap.min.js"></scrip>

</html>

通过浏览器打开upfile.html文件，功能如下图。

接下来通过sendKeys()方法来实现文件上传。

import java.io.File;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class UpFileDemo {

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

WebDriver driver = new ChromeDriver();

File file = new File("./HTMLFile/upfile.html");

String filePath = file.getAbsolutePath();

driver.get(filePath);

//定位上传按钮， 添加本地文件

driver.findElement(By.name("file")).sendKeys("D:\\upload\_file.txt");

Thread.sleep(5000);

driver.quit();

}

}

## 浏览器cookie操作（选做）

有时候我们需要验证浏览器中Cookie是否正确， 因为基于真实Cookie的测试是**无法通过白盒测试和集成测试进行的**。WebDriver提供了操作Cookie的相关方法可以读取、 添加和删除Cookie信息。

WebDriver 操作Cookie的方法：

* getCookies() 获得所有 cookie 信息。
* getCookieNamed(String name) 返回字典的key为“name”的Cookie信息。
* addCookie(cookie dict) 添加Cookie。“cookie\_dict”指字典对象，必须有 name和value值。
* deleteCookieNamed(String name) 删除Cookie 信息。 “name”是要删除的 cookie的名称； “optionsString” 是该Cookie的选项，目前支持的选项包括“路径” ， “域” 。
* deleteAllCookies() 删除所有 cookie 信息。

下面通过 geCookies()来获取当前浏览器的 cookie 信息。

import java.util.Set;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.Cookie;

public class CookieDemo {

public static void main(String[] args){

WebDriver driver = new ChromeDriver();

driver.get("https://www.baidu.com");

Cookie c1 = new Cookie("name", "key-aaaaaaa");

Cookie c2 = new Cookie("value", "value-bbbbbb");

driver.manage().addCookie(c1);

driver.manage().addCookie(c2);

//获得 cookie

Set<Cookie> coo = driver.manage().getCookies();

System.out.println(coo);

//删除所有 cookie

//driver.manage().deleteAllCookies();

driver.quit();

}

}

打印结果：

[BIDUPSID=82803D3E2DAD0F5342D22C8F96B9E088; expires=星期六, 24 二月 208512:40:10 CST; path=/; domain=.baidu.com, name=key-aaaaaaa; path=/;domain=www.baidu.com, PSTM=1486301167; expires=星期六, 24 二月 2085 12:40:10 CST;path=/; domain=.baidu.com,H\_PS\_PSSID=1437\_21094\_21943\_22023; path=/;domain=.baidu.com, BD\_UPN=12314753; expires=星期三, 15 二月 2017 09:26:04 CST;path=/; domain=www.baidu.com, value=value-bbbbbb; path=/;domain=www.baidu.com,BAIDUID=82803D3E2DAD0F5342D22C8F96B9E088:FG=1; expires=星期六, 24 二月 208512:40:10 CST; path=/; domain=.baidu.com, BD\_HOME=0; path=/;domain=www.baidu.com, \_\_bsi=16852840641557463410\_00\_0\_I\_R\_1\_0303\_C02F\_N\_I\_I\_0;expires=星期日, 05 二月 2017 09:26:10 CST; path=/; domain=.www.baidu.com]

## 调用JavaScript代码（选做）

虽然WebDriver提供了操作浏览器的前进和后退方法，但对于浏览器滚动条并没有提供相应的操作方法。在这种情况下，就可以借助JavaScript来控制浏览器的滚动条。WebDriver提供了executeScript()方法来执行JavaScript代码。

用于调整浏览器滚动条位置的JavaScript代码如下：

<!-- window.scrollTo(左边距,上边距); -->

window.scrollTo(0,450);

window.scrollTo()方法用于设置浏览器窗口滚动条的水平和垂直位置。方法的第一个参数表示水平的左间距，第二个参数表示垂直的上边距。其代码如下：

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.Dimension;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.JavascriptExecutor;

public class JSDemo {

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

WebDriver driver = new ChromeDriver();

//设置浏览器窗口大小

driver.manage().window().setSize(new Dimension(700, 600));

driver.get("https://www.baidu.com");

//进行百度搜索

driver.findElement(By.id("kw")).sendKeys("webdriver api");

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

Thread.sleep(2000);

//将页面滚动条拖到底部

((JavascriptExecutor)driver).executeScript("window.scrollTo(100,450);");

Thread.sleep(3000);

driver.quit();

}

}

通过浏览器打开百度进行搜索，并且提前通过window().setSize()方法将浏览器窗口设置为固定宽高显示，目的是让窗口出现水平和垂直滚动条。然后通过executeScript()方法执行JavaScripts代码来移动滚动条的位置。

## 获取窗口截图（选做）

自动化用例是由程序去执行，因此有时候打印的错误信息并不十分明确。如果在脚本执行出错的时候能对当前窗口截图保存，那么通过图片就可以非常直观地看出出错的原因。 WebDriver提供了截图函数getScreenshotAs()来截取当前窗口。

import java.io.File;

import java.io.IOException;

import org.apache.commons.io.FileUtils;

import org.openqa.selenium.OutputType;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.TakesScreenshot;

public class GetImg {

public static void main(String[] arge){

WebDriver driver = new ChromeDriver();

driver.get("https://www.baidu.com");

File srcFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

try {

FileUtils.copyFile(srcFile,new File("d:\\screenshot.png"));

} catch (IOException e) {

e.printStackTrace();

}

driver.quit();

}

}

脚本运行完成后打开D盘，就可以找到screenshot.png图片文件了。

# 参考资料

1. Selenium IDE环境配置https://zhuanlan.zhihu.com/p/103768251
2. Selenium IDE介绍https://www.pianshen.com/article/4105130534/
3. Selenium教程ttps://www.yiibai.com/selenium
4. Selenium Java <http://www.testclass.net/selenium_java>
5. 测试教程网Selenium Java <http://www.testclass.net/selenium_java>