Selenium2.0使用说明

目录

[第1章 Webdirver基础 2](#_Toc342576465)

[1.1   介绍WebDriver 2](#_Toc342576466)

[1.2   安装selenium webdriver 2](#_Toc342576467)

[1.2.1安装jdk并配置环境变量 2](#_Toc342576468)

[1.2.2安装Firefox，Selenium IDE，Firebug和xpahter 2](#_Toc342576469)

[1.2.3安装eclipse 2](#_Toc342576470)

[1.2.4安装selenium webdriver 2](#_Toc342576471)

[1.3   工具使用 2](#_Toc342576472)

[1.3.1 Selenium-IDE使用 2](#_Toc342576473)

[1.3.2 Firebug的使用 2](#_Toc342576474)

[1.3.3 xpath checker的使用 2](#_Toc342576475)

[1.3.4 Selenium 2.0的使用 2](#_Toc342576476)

[第2章 Webdirver对浏览器的支持 2](#_Toc342576477)

[2.1   HtmlUnit Driver 2](#_Toc342576478)

[2.2   FireFox Driver 2](#_Toc342576479)

[2.3   InternetExplorer Driver 2](#_Toc342576480)

[第3章 对浏览器的简单操作 2](#_Toc342576481)

[3.1   用webdriver打开一个浏览器 2](#_Toc342576482)

[3.2   打开测试页面 2](#_Toc342576483)

[3.3   关闭浏览器 2](#_Toc342576484)

[3.4   返回当前页面的url 和title 2](#_Toc342576485)

[第4章 使用操作 2](#_Toc342576486)

[4.1   如何找到页面元素 2](#_Toc342576487)

[4.1.1 By ID 2](#_Toc342576488)

[4.1.2 By Name 2](#_Toc342576489)

[4.1.3 By XPATH 2](#_Toc342576490)

[4.1.4 By Class Name 2](#_Toc342576491)

[4.1.5 By Link Text 2](#_Toc342576492)

[4.2   如何对页面元素进行操作 2](#_Toc342576493)

[4.2.1 输入框（text field or textarea） 2](#_Toc342576494)

[4.2.2 下拉选择框(Select) 2](#_Toc342576495)

[4.2.3 单选项(Radio Button) 2](#_Toc342576496)

[4.2.4 多选项(checkbox) 2](#_Toc342576497)

[4.2.5 按钮(button) 2](#_Toc342576498)

[4.2.6 左右选择框 2](#_Toc342576499)

[4.2.7 弹出对话框(Popup dialogs) 2](#_Toc342576500)

[4.2.8 表单(Form) 2](#_Toc342576501)

[4.2.9 上传文件 (Upload File) 2](#_Toc342576502)

[4.2.10 Windows 和 Frames之间的切换 2](#_Toc342576503)

[4.2.11 拖拉(Drag andDrop) 2](#_Toc342576504)

[4.2.12 导航 (Navigationand History) 2](#_Toc342576505)

[4.3   高级使用 2](#_Toc342576506)

[4.3.1 改变user agent 2](#_Toc342576507)

[4.3.2 读取Cookies 2](#_Toc342576508)

[4.3.3 调用Java Script 2](#_Toc342576509)

[4.3.4 Webdriver截图 2](#_Toc342576510)

[4.3.5 页面等待 2](#_Toc342576511)

[第5章 RemoteWebDriver 2](#_Toc342576512)

[5.1 使用RemoteWebDriver 2](#_Toc342576513)

[5.2 SeleniumServer 2](#_Toc342576514)

[5.3 How to setFirefox profile using RemoteWebDriver 2](#_Toc342576515)

# 第1章 Webdirver基础

## 1.1   介绍WebDriver

Selenium 2.0主要的特性就是与WebDriver API的集成。WebDriver旨在提供一个更简单更简洁的编程接口以及解决一些Selenium-RC API的限制。Selenium-Webdriver更好的支持页面本身不重新加载而页面的元素改变的动态网页。WebDriver的目标是提供一个良好设计的面向对象的API提供了对于现代先进web应用程序测试问题的改进支持。Selenium Webdriver 支持很多语言，如：C#，Java，Python，Ruby 等，本文使用java。

## 1.2   安装selenium webdriver

### 1.2.1安装jdk并配置环境变量

jdk安装  
jdk需要1.6版本及以上的，安装目录为默认路径。  
jdk下载地址： <http://www.oracle.com/technetwork/java/javase/downloads/index.html>

环境变量配置   
CLASSPATH=.;%JAVA\_HOME%\lib\dt.jar;%JAVA\_HOME%\lib\tools.jar  
JAVA\_HOME=C:\Program Files\Java\jdk1.6.0\_10  
PATH=%JAVA\_HOME%\bin

### 1.2.2安装Firefox，Selenium IDE，Firebug和xpahter

安装FireFox

Firefox版本有一定限制，需要和selenium IDE相匹配。  
下载地址： <http://www.firefox.com.cn/download/>

安装Selenium IDE  
Selenium IDE是基于FIREFOX浏览器的一个插件，提供GUI界面来运行Selenium测试。Selenium IDE提供脚本录制和回放功能，可以将用户在浏览器中执行的操作记录下来，生成各种形式的脚本，可以将这些脚本保存供selenium使用。Selenium IDE主要是用在Selenium 1.0中，在Selenium 2.0中基本不使用。  
1）下载Selenim IDE  
下载地址：<http://seleniumhq.org/projects/ide/>

2）安装：直接把下载的Selenium IDE文件拖到FireFox浏览器窗口中，按提示操作即可安装成功。

安装Firebug  
1）打开Firefox浏览器  
2）点击菜单“工具（T）”，下拉列表中选择“附加组件”。  
3）“获取附加组件”  
4）在搜索里输入“firebug”，稍等即可。  
5）点击“添加至Firefox”  
6）OK，重启浏览器即可。

安装xpahter  
1）打开Firefox浏览器  
2）点击菜单“工具（T）”，下拉列表中选择“附加组件”。  
3）“获取附加组件”  
4）在搜索里输入“xpahter”，稍等即可。  
5）点击“添加至Firefox”  
6）OK，重启浏览器即可。

安装xpath checker

1）打开Firefox浏览器  
2）点击菜单“工具（T）”，下拉列表中选择“附加组件”。  
3）“获取附加组件”  
4）在搜索里输入“xpath checker”，稍等即可。  
5）点击“添加至Firefox”  
6）OK，重启浏览器即可。

### 1.2.3安装eclipse

安装eclipse，安装在默认安装目录下。

### 1.2.4安装selenium webdriver

1）下载地址： <http://code.google.com/p/selenium/downloads/list>

官方UserGuide：<http://seleniumhq.org/docs/>

2）下载：selenium-server-standalone-2.25.0.jar和selenium-java-2.25.0.zip（使用java语言的下载该包）。   
3）解压下载的selenium-java-2.25.0.zip文件

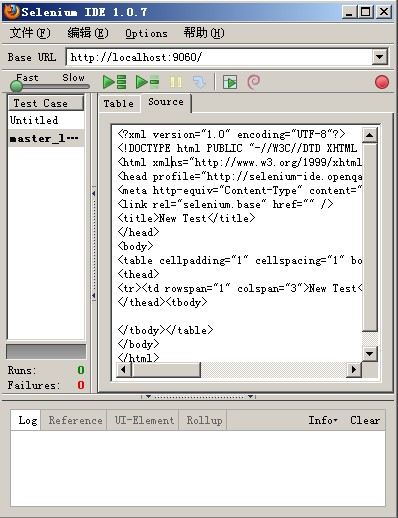
## 1.3   工具使用

### 1.3.1 Selenium-IDE使用

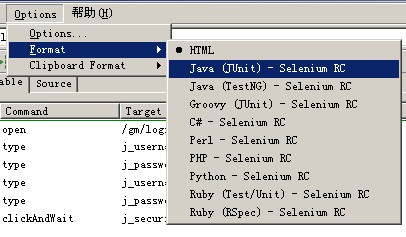
1）Firefox工具栏，打开Selenium-IDE插件，如下图：

****

2）在“Base URL”中输入需要录制脚本的网址地址，然后点击界面中右上角红色录制按钮（开始录制、停止录制都是此按钮），如下图。

****

3）录制完成后，点击回放按钮可以对刚刚录制的脚本进行回放，这里可以调整回放速度。 回放按钮就是上图中的绿色三角按钮。  
4）还可以将录制的脚本转换成C#，Java，PHP，Ruby，Prel，Groovy，Python等语言，如下图所示。转换完以后可以直接复制到Eclipse中使用，在Selenium 1.0中稍加修改就可使用，在Selenium 2.0则不能直接使用，只能手动编制。



### 1.3.2 Firebug的使用

打开Firebug有两种基本的方法：

1）当需要获取页面上元素的属性时，可以使用Firebug进行获取，例如下图所示输入框：



获取方法：右击输入框，在菜单中选择“查看元素”



此时将会在页面下方出现该元素的属性，如下图所示：



然后右击“编辑”后面的内容，在菜单中选择“复制xpath”，即可使用。



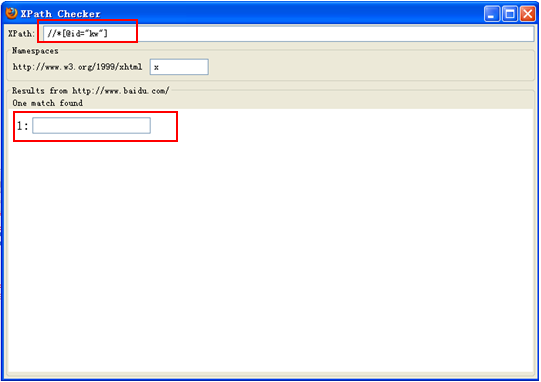
2）Firebug也可以通过点击Firefox浏览器上的“工具”菜单->“Firebug”->“打开Firebug”进行打开。

### 1.3.3 xpath checker的使用

元素的属性还可以通过xpath checker来查看。对于上面百度输入框的属性可以通过右击并选择菜单中的“view xpath“按钮



此时会打开如下窗口，“XPath”中的内容就是该输入框的属性，在下面会显示对应的元素。



### 1.3.4 Selenium 2.0的使用

1）在Eclipse里新建一个project，然后引用selenium-java-2.25.0.zip解压后的selenium-java-2.25.0.jar。

2）新建一个class“Seleniumcn”把代码贴进去，如果代码没错误就可以运行了。例如下面：

**package** 辅助测试;

**import** java.util.Set;

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

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

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

**import** org.openqa.selenium.firefox.FirefoxDriver;

**public** **class** Seleniumcn {

**public** **static** **void** main(String[] args) {

System.*setProperty*("javax.xml.parsers.DocumentBuilderFactory","com.sun.org.apache.xerces.internal.jaxp.DocumentBuilderFactoryImpl");

WebDriver driver = **new** FirefoxDriver();

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

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

element.sendKeys("hello Selenium!");

element.submit();

**try** {

Thread.*sleep*(3000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

System.*out*.println("Page title is: " + driver.getTitle());

driver.quit();

}

}

正常运行后，这几行代码将会打开firefox 浏览器，然后转跳到google 首页。在搜索框中输入hello Selenium 并提交搜索结果。等待3 秒后会在命令行打印出当前页面的title，输出如下：

Page title is: hello Selenium! - Google 搜索

并关闭ff 浏览器。

# 第2章 Webdirver对浏览器的支持

## 2.1   HtmlUnit Driver

优点：HtmlUnit Driver不会实际打开浏览器，运行速度很快。对于用FireFox等浏览器来做测试的自动化测试用例，运行速度通常很慢，HtmlUnit Driver无疑是可以很好地解决这个问题。

缺点：它对JavaScript的支持不够好，当页面上有复杂JavaScript时，经常会捕获不到页面元素。

使用：

WebDriver driver = new HtmlUnitDriver();

## 2.2   FireFox Driver

优点：FireFox Dirver对页面的自动化测试支持得比较好，很直观地模拟页面的操作，对JavaScript的支持也非常完善，基本上页面上做的所有操作FireFox Driver都可以模拟。

缺点：启动很慢，运行也比较慢，不过，启动之后Webdriver的操作速度虽然不快但还是可以接受的。

使用：

WebDriver driver = new FirefoxDriver();

## 2.3   InternetExplorer Driver

优点：直观地模拟用户的实际操作，对JavaScript提供完善的支持。

缺点：是所有浏览器中运行速度最慢的，并且只能在Windows下运行，对CSS以及XPATH的支持也不够好。

使用：

WebDriver driver = new InternetExplorerDriver();

# 第3章 对浏览器的简单操作

## 3.1   用webdriver打开一个浏览器

我们常用的浏览器有firefox和IE两种，firefox是selenium支持得比较成熟的浏览器，还支持HtmlUnit浏览器，打开语句为：

打开firefox浏览器：

        //Create a newinstance of the Firefox driver

        WebDriver driver = newFirefoxDriver();

打开IE浏览器

        //Create a newinstance of the Internet Explorer driver

        WebDriver driver = newInternetExplorerDriver ();

打开HtmlUnit浏览器

        //Createa new instance of the Internet Explorer driver

        WebDriverdriver = new HtmlUnitDriver();

## 3.2   打开测试页面

对页面测试，首先要打开被测试页面的地址（如：[http://www.baidu.com](http://www.google.com/)）,web driver 提供的get方法可以打开一个页面：

        // And now use thedriver to visit Google

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

## 3.3   关闭浏览器

测试完成后，需要关闭浏览器

//Close the browser

driver.quit();

## 3.4   返回当前页面的url 和title

有时候我们需要返回当前页面的url 或者title 做一些验证性的操作等。代码如

下：

//得到title

String title = driver.getTitle();

//得到当前页面url

String currentUrl = driver.getCurrentUrl();

//输出title 和currenturl

System.out.println(title+"\n"+currentUrl);

# 第4章 使用操作

## 4.1   如何找到页面元素

Webdriver的findElement方法可以用来找到页面的某个元素，最常用的方法是用id和name查找。下面介绍几种比较常用的方法。

### 4.1.1 By ID

假设页面写成这样：

<input type="text" name="passwd"id="passwd-id" />

 那么可以这样找到页面的元素：

通过id查找：

WebElement element = driver.findElement(By.id("passwd-id"));

### 4.1.2 By Name

或通过name查找：

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

### 4.1.3 By XPATH

或通过xpath查找：

WebElement element =driver.findElement(By.xpath("//input[@id='passwd-id']"));

### 4.1.4 By Class Name

假设页面写成这样：

 <div class="cheese">

<span>Cheddar</span>

</div>

<div class="cheese">

<span>Gouda</span>

</div>

可以通过这样查找页面元素：

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

### 4.1.5 By Link Text

假设页面元素写成这样：

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

 那么可以通过这样查找：

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

## 4.2   如何对页面元素进行操作

找到页面元素后，怎样对页面进行操作呢？我们可以根据不同的类型的元素来进行一一说明。

### 4.2.1 输入框（text field or textarea）

   找到输入框元素：

WebElement element = driver.findElement(By.id("passwd-id"));

在输入框中输入内容：

element.sendKeys(“test”);

将输入框清空：

element.clear();

获取输入框的文本内容：

element.getText();

### 4.2.2 下拉选择框(Select)

找到下拉选择框的元素：

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

 选择对应的选择项：

select.selectByVisibleText(“mediaAgencyA”);

或

select.selectByValue(“MA\_ID\_001”);

不选择对应的选择项：

select.deselectAll();

select.deselectByValue(“MA\_ID\_001”);

select.deselectByVisibleText(“mediaAgencyA”);

或者获取选择项的值：

select.getAllSelectedOptions();

select.getFirstSelectedOption();

### 4.2.3 单选项(Radio Button)

找到单选框元素：

WebElement bookMode =driver.findElement(By.id("BookMode"));

选择某个单选项：

bookMode.click();

清空某个单选项：

bookMode.clear();

判断某个单选项是否已经被选择：

bookMode.isSelected();

### 4.2.4 多选项(checkbox)

多选项的操作和单选的差不多：

WebElement checkbox =driver.findElement(By.id("myCheckbox."));

checkbox.click();

checkbox.clear();

checkbox.isSelected();

checkbox.isEnabled();

### 4.2.5 按钮(button)

找到按钮元素：

WebElement saveButton = driver.findElement(By.id("save"));

点击按钮：

saveButton.click();

判断按钮是否enable:

saveButton.isEnabled ();

### 4.2.6 左右选择框

也就是左边是可供选择项，选择后移动到右边的框中，反之亦然。例如：

Select lang = new Select(driver.findElement(By.id("languages")));

lang.selectByVisibleText(“English”);

WebElement addLanguage =driver.findElement(By.id("addButton"));

addLanguage.click();

### 4.2.7 弹出对话框(Popup dialogs)

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

alert.accept();

alert.dismiss();

alert.getText();

### 4.2.8 表单(Form)

Form中的元素的操作和其它的元素操作一样，对元素操作完成后对表单的提交可以：

WebElement approve = driver.findElement(By.id("approve"));

approve.click();

或

approve.submit();//只适合于表单的提交

### 4.2.9 上传文件 (Upload File)

上传文件的元素操作：

WebElement adFileUpload = driver.findElement(By.id("WAP-upload"));

String filePath = "C:\test\\uploadfile\\media\_ads\\test.jpg";

adFileUpload.sendKeys(filePath);

### 4.2.10 Windows 和 Frames之间的切换

一般来说，登录后建议是先：

driver.switchTo().defaultContent();

切换到某个frame：

driver.switchTo().frame("leftFrame");

从一个frame切换到另一个frame：

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

切换到某个window：

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

### 4.2.11 拖拉(Drag andDrop)

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

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

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

### 4.2.12 导航 (Navigationand History)

打开一个新的页面：

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

通过历史导航返回原页面：

driver.navigate().forward();

driver.navigate().back();

## 4.3   高级使用

### 4.3.1 改变user agent

User Agent的设置是平时使用得比较多的操作：

FirefoxProfile profile = new FirefoxProfile();

profile.addAdditionalPreference("general.useragent.override","some UA string");

WebDriver driver = new FirefoxDriver(profile);

### 4.3.2 读取Cookies

我们经常要对的值进行读取和设置。

增加cookie:

// Now set the cookie. This one's valid for the entire domain

Cookie cookie = new Cookie("key", "value");

driver.manage().addCookie(cookie);

获取cookie的值：

// And now output all the available cookies for the current URL

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

for (Cookie loadedCookie : allCookies) {

   System.out.println(String.format("%s -> %s",loadedCookie.getName(), loadedCookie.getValue()));

}

根据某个cookie的name获取cookie的值：

driver.manage().getCookieNamed("mmsid");

删除cookie:

// You can delete cookies in 3 ways

// By name

driver.manage().deleteCookieNamed("CookieName");

// By Cookie

driver.manage().deleteCookie(loadedCookie);

// Or all of them

driver.manage().deleteAllCookies();

### 4.3.3 调用Java Script

Web driver对Java Script的调用是通过JavascriptExecutor来实现的，例如：

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript("(function(){inventoryGridMgr.setTableFieldValue('"+ inventoryId + "','" + fieldName + "','"+ value + "');})()");

### 4.3.4 Webdriver截图

如果用webdriver截图是：

driver = webdriver.Firefox()  
driver.save\_screenshot("C:\error.jpg")

### 4.3.5 页面等待

因为Load页面需要一段时间，如果页面还没加载完就查找元素，必然是查找不到的。最好的方式，就是设置一个默认等待时间，在查找页面元素的时候如果找不到就等待一段时间再找，直到超时。

Webdriver提供两种方法，一种是显性等待，另一种是隐性等待。

显性等待：

WebDriver driver =new FirefoxDriver();

driver.get("http://somedomain/url\_that\_delays\_loading");

WebElement myDynamicElement = (new WebDriverWait(driver, 10))

  .until(new ExpectedCondition<WebElement>(){

  @Override

  public WebElement apply(WebDriver d) {

    returnd.findElement(By.id("myDynamicElement"));

  }});

隐性等待：

WebDriver driver = new FirefoxDriver();

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

driver.get("http://somedomain/url\_that\_delays\_loading");

WebElement myDynamicElement =driver.findElement(By.id("myDynamicElement"));

# 第5章 RemoteWebDriver

当本机上没有浏览器，需要远程调用浏览器进行自动化测试时，需要用到RemoteWebDirver.

## 5.1 使用RemoteWebDriver

import java.io.File;

import java.net.URL;

import org.openqa.selenium.OutputType;

import org.openqa.selenium.TakesScreenshot;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.remote.Augmenter;

import org.openqa.selenium.remote.DesiredCapabilities;

import org.openqa.selenium.remote.RemoteWebDriver;

public class Testing {

    public void myTest()throws Exception {

        WebDriver driver = newRemoteWebDriver(

                               new URL("http://localhost:4446/wd/hub"),

                               DesiredCapabilities.firefox());

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

        // RemoteWebDriverdoes not implement the TakesScreenshot class

        // if the driver doeshave the Capabilities to take a screenshot

        // then Augmenter willadd the TakesScreenshot methods to the instance

        WebDriveraugmentedDriver = new Augmenter().augment(driver);

        File screenshot =((TakesScreenshot)augmentedDriver).

                            getScreenshotAs(OutputType.FILE);

    }

}

## 5.2 SeleniumServer

在使用RemoteDriver时，必须在远程服务器启动一个SeleniumServer:

java -jar selenium-server-standalone-2.25.0.jar -port 4446

## 5.3 How to setFirefox profile using RemoteWebDriver

profile = new FirefoxProfile();

           profile.setPreference("general.useragent.override",testData.getUserAgent());

capabilities = DesiredCapabilities.firefox();

capabilities.setCapability("firefox\_profile", profile);

driver = new RemoteWebDriver(new URL(“http://localhost:4446/wd/hub”),capabilities);

driverWait = new WebDriverWait(driver,TestConstant.WAIT\_ELEMENT\_TO\_LOAD);

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