Selenium入门

一．元素定位方式

Selenium提供了8种定位方式

1. id定位：
2. class定位
3. Name定位(name元素一般不唯一,可通过getindex下标或for循环遍历获取)
4. Tagname定位
5. xpath定位
6. css定位
7. linkText定位(根据text文本内容进行定位)
8. partial\_link定位(模糊文本定位)

注：

1. 当页面元素有id属性时，最好尽量用id来定位。但由于现实项目中很多程序员其实写的代码并不规范，会缺少很多标准属性，这时就只有选择其他定位方法。

2. xpath很强悍，但定位性能不是很好，当脚本存在大量xpath定位时速度会缓慢，所以还是尽量少用。如果确实少数元素不好定位，可以选择xpath或css。

3. 当要定位一组元素相同元素时，可以考虑用tagName或name。

4. 当有链接需要定位时，可以考虑linkText或partialLinkText方式。

二．iframe切换

三．句柄切换

四．键盘，鼠标模拟操作

 //生成Actions实例对象     Actions actions=new Actions(driver);

鼠标

* Perform：执行所有ActionChains存储的行为
* context\_click()：右击
* double\_click()：双击
* drag\_and\_drop()：拖动
* move\_to\_element()：悬停

键盘(有Robot,awt两种)

* **Keys.BACK\_SPACE**：删除键
* **Keys.SPACE**：空格键
* **Keys.TAB**：Tab键
* **Keys.ESCAPE**：回退键
* **Keys.ENTER**：回车键
* **Keys.CONTROL,”a”**：组合键，Ctrl + A
* **Keys.CONTROL,”x”**：组合键，Ctrl + X
* **Keys.CONTROL,”v”**：组合键，Ctrl + V
* **Keys.CONTROL,”c”**：组合键，Ctrl + C
* **Keys.F1**：F1键
* **Keys.F12**：F12键

1. JS操作

1.输入

((JavascriptExecutor) driver).executeScript("arguments[0].value=\"北京\"", from\_inpox);

2.去掉只读属性

((JavascriptExecutor)driver).executeScript("arguments[0].removeAttribute(\"+"readonly"+\")", e);

1. 滚动条回到顶部

((JavascriptExecutor) driver).executeScript("window.scrollTo(document.body.scrollHeight,0)");

4.滚动条拉到底部

((JavascriptExecutor) driver).executeScript("window.scrollTo(0, document.body.scrollHeight)");

5.移动到指定坐标(相对坐标)  
((JavascriptExecutor) driver).executeScript("window.scrollBy(0, 700)");

5.移动到指定坐标(绝对坐标)  
((JavascriptExecutor) driver).executeScript("window.scrollTo(0, 1600)");

六．弹窗处理

七．元素等待

隐式等待

显示等待

强制等待

八．TestNg框架

九．分层思想

十．持续集成

十一．Selenium操作原理

在WebDriver中有类似的三个角色：

自动化测试代码：自动化测试代码发送请求给浏览器的驱动（比如火狐驱动、谷歌驱动）

浏览器的驱动：它来解析这些自动化测试的代码，解析后把它们发送给浏览器

浏览器：执行浏览器驱动发来的指令，并最终完成工程师想要的操作。

所以在这个类比中：

工程师写的自动化测试代码就相当于是乘客

浏览器的驱动就相当于是出租车司机

浏览器就相当于是出租车

面再从技术上解释下WebDriver的工作原理：

从技术上讲，也同样是上面的三个角色：

1.WebDriver API（基于Java、Python、C#等语言）

对于java语言来说，就是下载下来的selenium的Jar包，比如selenium-java-3.8.1.zip包，代表Selenium3.8.1的版本

2.浏览器的驱动（browser driver）

每个浏览器都有自己的驱动，均以exe文件形式存在

比如谷歌的chromedriver.exe、火狐的geckodriver.exe、IE的IEDriverServer.exe

3.浏览器

浏览器当然就是我们很熟悉的常用的各种浏览器。

那在WebDriver脚本运行的时候，它们之间是如何通信的呢？为什么同一个browser driver即可以处理java语言的脚本，也可以处理python语言的脚本呢？让我们来看一下，一条Selenium脚本执行时后端都发生了哪些事情：

对于每一条Selenium脚本，一个http请求会被创建并且发送给浏览器的驱动

浏览器驱动中包含了一个HTTP Server，用来接收这些http请求

HTTP Server接收到请求后根据请求来具体操控对应的浏览器

浏览器执行具体的测试步骤

浏览器将步骤执行结果返回给HTTP Server

HTTP Server又将结果返回给Selenium的脚本，如果是错误的http代码我们就会在控制台看到对应的报错信息。

为什么使用HTTP协议呢？

因为HTTP协议是一个浏览器和Web服务器之间通信的标准协议，而几乎每一种编程语言都提供了丰富的http libraries，这样就可以方便的处理客户端Client和服务器Server之间的请求request及响应response，WebDriver的结构中就是典型的C/S结构，WebDriver API相当于是客户端，而小小的浏览器驱动才是服务器端。

WebDriver基于的协议：JSON Wire protocol。

JSON Wire protocol是在http协议基础上，对http请求及响应的body部分的数据的进一步规范。

我们知道在HTTP请求及响应中常常包括以下几个部分：http请求方法、http请求及响应内容body、http响应状态码等。

常见的http请求方法：

GET：用来从服务器获取信息。比如获取网页的标题信息

POST：向服务器发送操作请求。比如findElement，Click等

http响应状态码：

在WebDriver中为了给用户以更明确的反馈信息，提供了更细化的http响应状态码，比如：

7：NoSuchElement

11：ElementNotVisible

200：Everything OK

现在到了最关键的http请求及响应的body部分了：

body部分主要传送具体的数据，在WebDriver中这些数据都是以JSON的形式存在并进行传送的，这就是JSON Wire protocol。

Selenium 是将各个浏览器的API封装成" Selenium自己设计定义的协议，名字叫做The WebDriver Wire Protocol " 的webdriver API

操作层面：

1、测试人员编写UI自动化测试脚本（java,python等等），运行脚本后，程序会打开指定的webdriver浏览器

webdriver浏览器作为一个remote-server 接受脚本的命令，同时webservice会打开一个端口：http://localhost:9515 浏览器则会监听这个端口

2、webservice会将脚本语言翻译成json格式传递给浏览器执行操作命令

逻辑层面：

1、测试人员执行测试脚本后，就创建了一个session, 通过http 请求向webservice发送了restfull的请求。

2、webservice翻译restfull的请求为浏览器能懂的脚本，然后接受脚本执行结果。

3、webservice将结果进行封装--json 给到客户端client/测试脚本 ，然后client就知道操作是否成功，同时测试也可以进行校验了。