

利用 Selenium RC 搭建自动化测试框架

最近利用业余时间学习利用 selenium RC 来进行自动化测试的初步框架，并写了几个简单的自动化用例脚本，实现了从外部 excel 档（即自动化测试用例）中读取测试数据，实现检查点校验并将测试结果写入测试用例文档，并对测试结果进行统计。现在将具体实现过程进行归纳如下。

1 selenium 简介

Selenium 是 ThoughtWorks 公司开发的一套基于 WEB 应用的测试工具，直接运行在浏览器中，仿真用户的操作，它可以被用于单元测试，回归测试，冒烟测试，集成测试，验收测试，并且可以运行在各种浏览器和操作系统上。

➤ 为什么不用 qtp

Selenium 可以很好的支持包括 firefox 浏览器在内的各种浏览器，而 qtp 对 firefox 的支持很差。我们公司的 OMS 系统基本上是基于 firefox 开发的，从工具的兼容性及自动化脚本的调试难度来说，用 selenium 更加便利

Qtp 的脚本语言仅支持 vbscript，selenium 可以直接使用 JAVA、python 等语言进行编写，从开发语言上手难易程度来讲，采用 selenium 更加容易上手。

2 Selenium RC 搭建自动测试框架

2.1.配置 selenium 运行环境

2.1.1 安装 JDK

这里我们使用 java 作为自动化脚本开发语言，要安装 JDK，并配置环境变量
目前服务端的各台测试机上应该都安装了，可以从 开始->运行->cmd 中调出命令行窗口，输入 java -version 如果能查看到当前的 jdk 版本信息，代表 jdk 安装成功，见图。

```
C:\Documents and Settings\Administrator>java -version
java version "1.6.0_21"
Java(TM) SE Runtime Environment (build 1.6.0_21-b06)
Java HotSpot(TM) Client VM (build 17.0-b16, mixed mode, sharing)
```

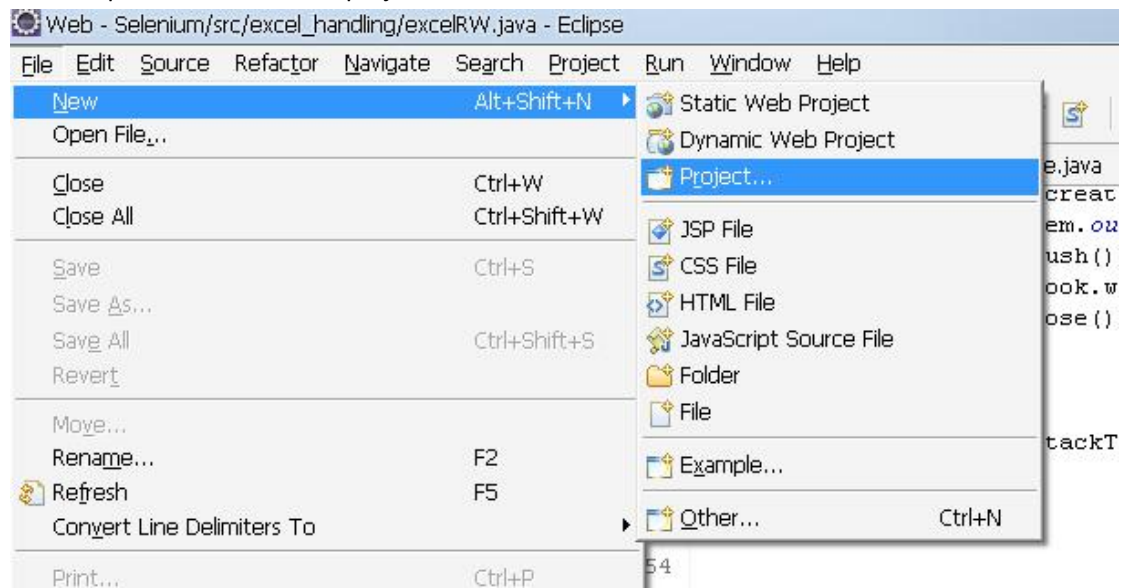
2.1.2 在 eclipse 上部署一个自动化测试工程

2.1.2.1 安装 eclipse

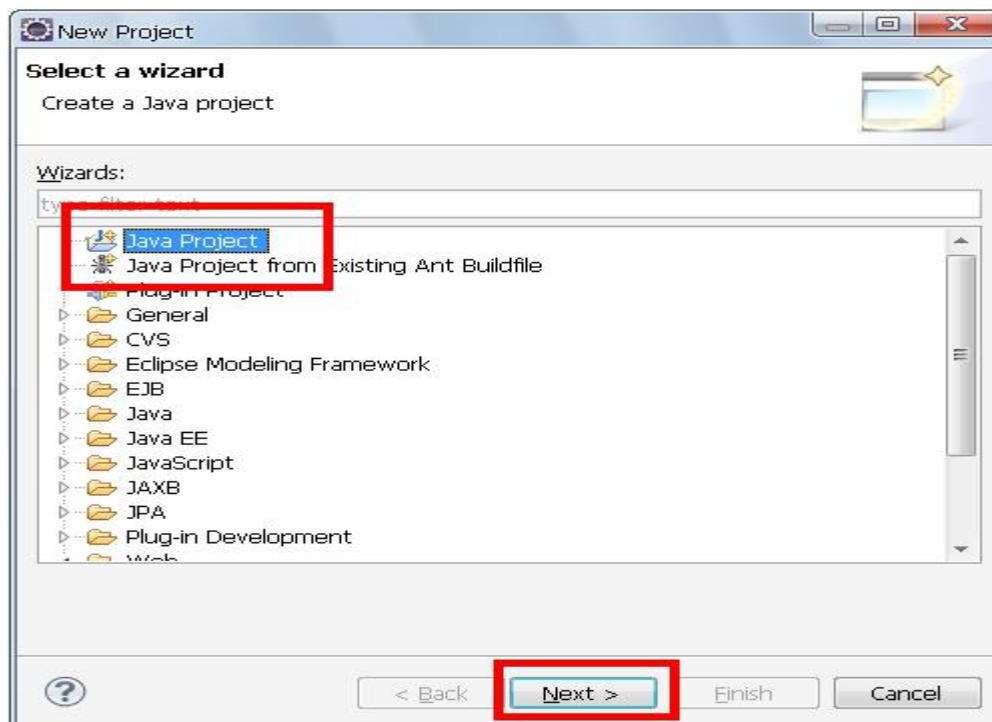
在配置好 jdk 后，下载 eclipse 安装包，然后按照提示信息一步一步安装就行。

2.1.2.2 新建一个自动化测试工程

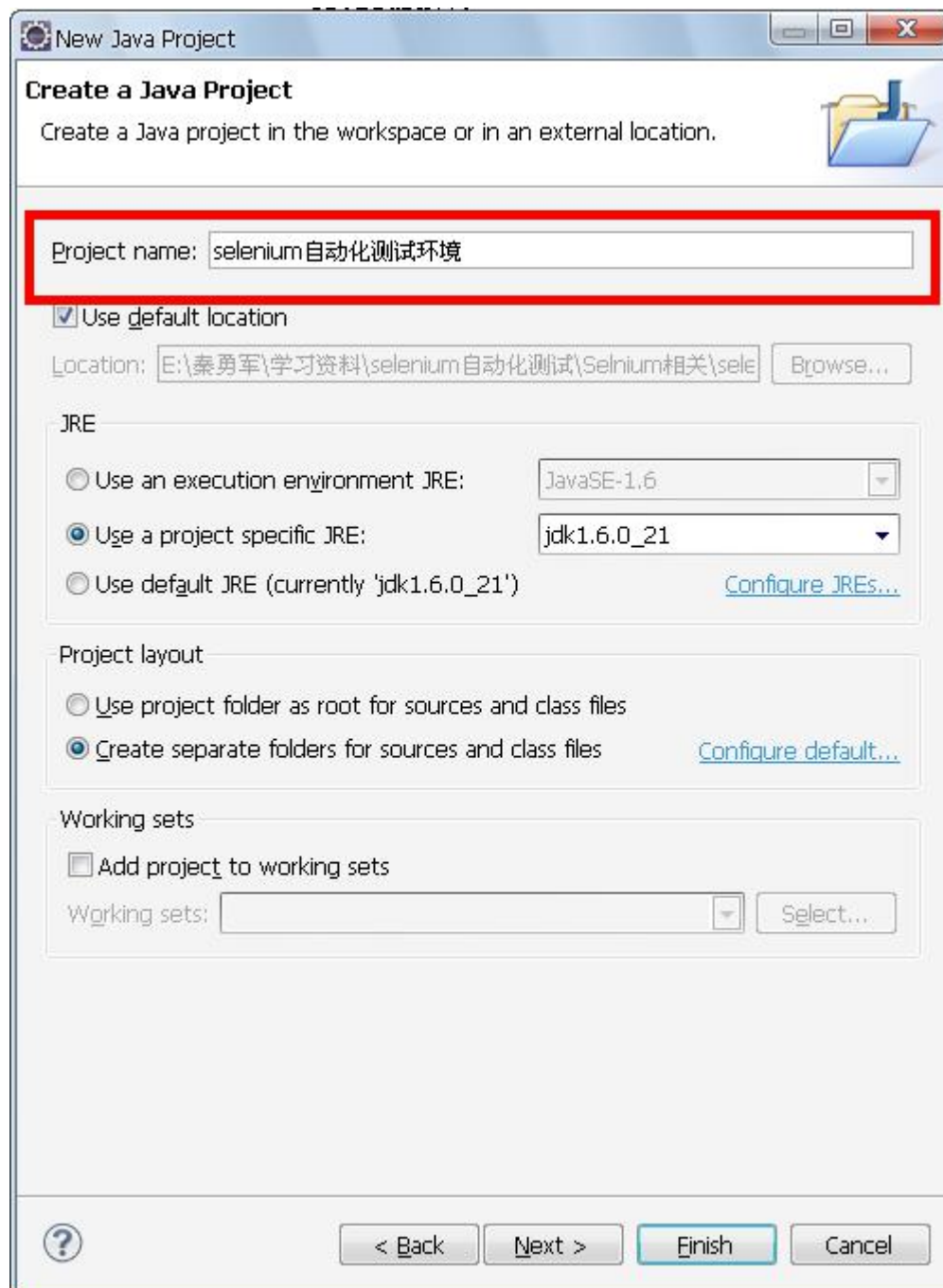
打开 eclipse,点击 file->new->project, 新建一个工程



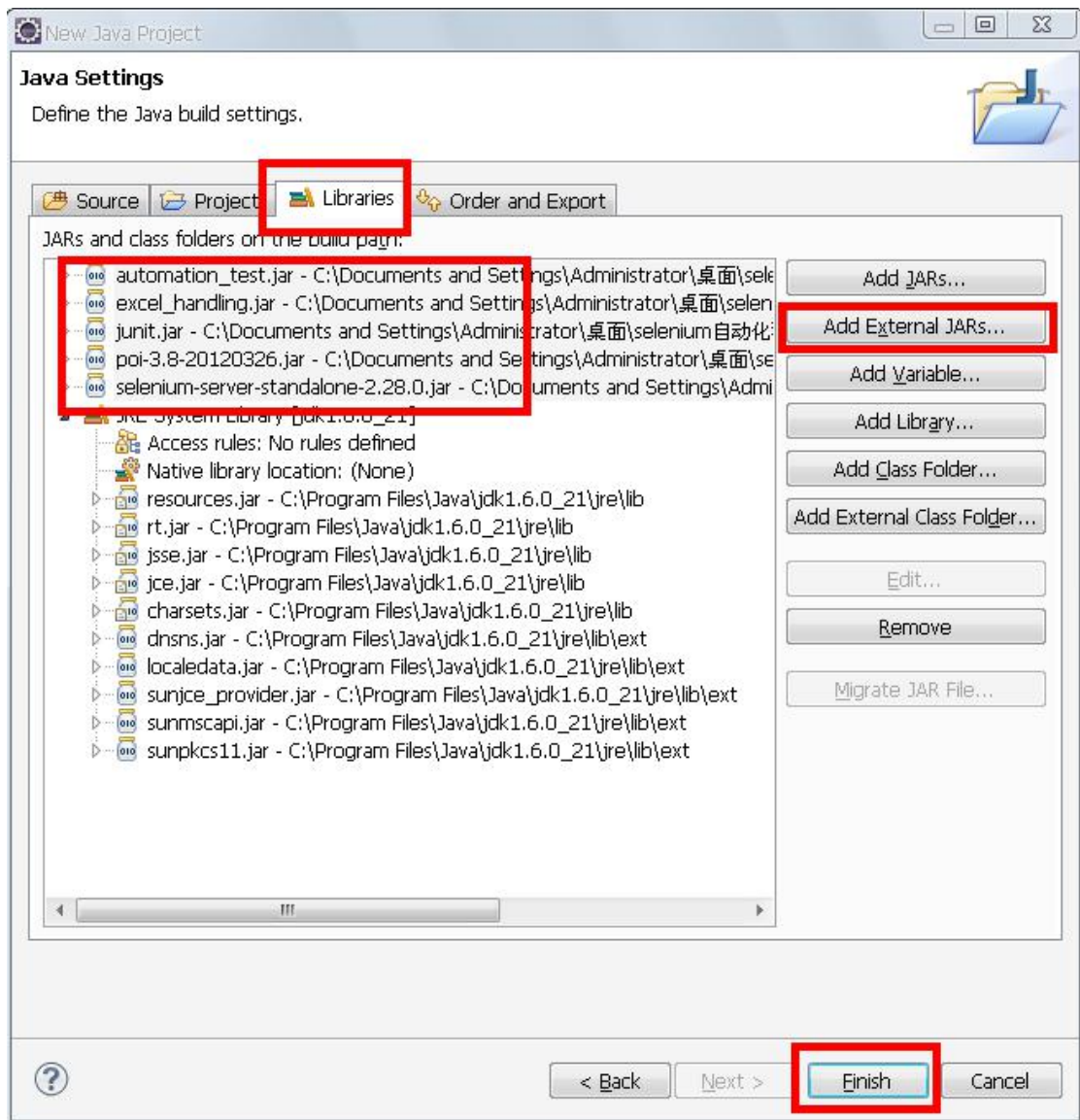
选择 Java Project, 然后 next



输入工程名，next



在这个界面上我们点击到 **libraries** 标签，引入本次自动化测试所需要的 4 个 jar 包：



其中:

excel_handling.jar 是 excel 表格的处理函数, 包括读写等等;

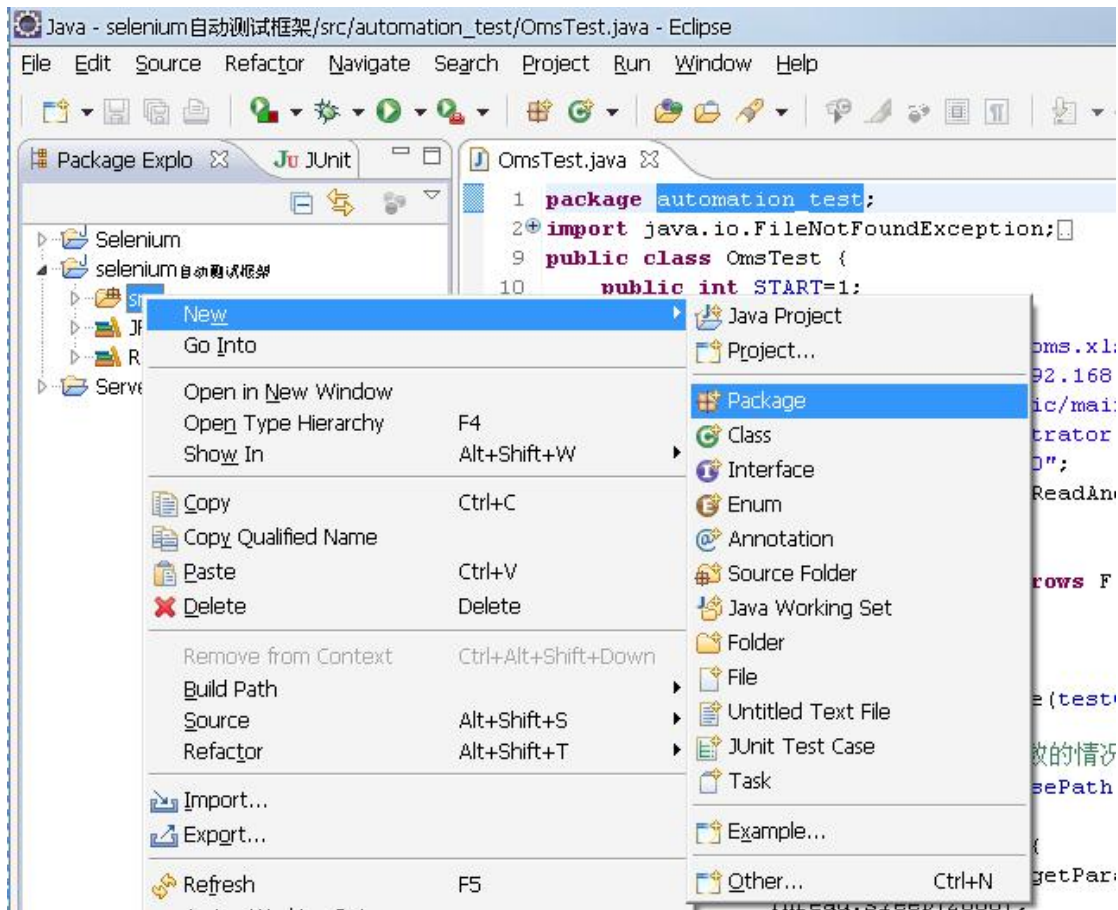
junit.jar 是执行测试的支持包;

poi-3.8-20120326.jar 是 excel 支持包;

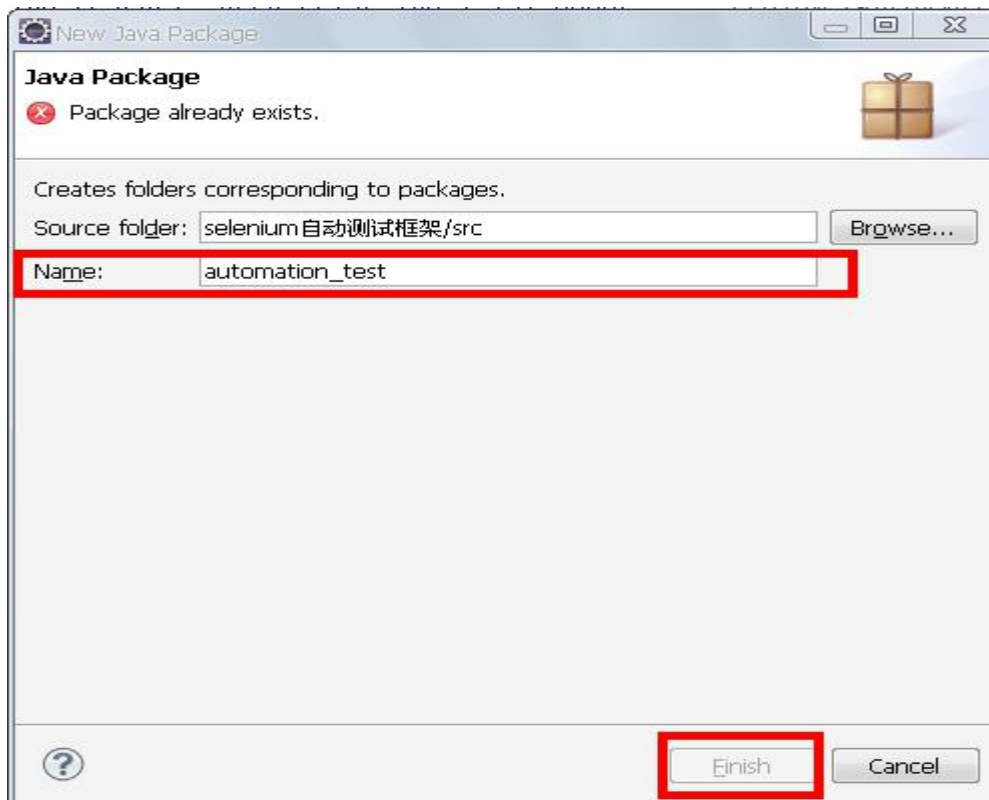
selenium-server-standalone-2.28.0.jar 是 selenium 的支持包。

2.1.2.3 新建一个主程序 JAVA 文件

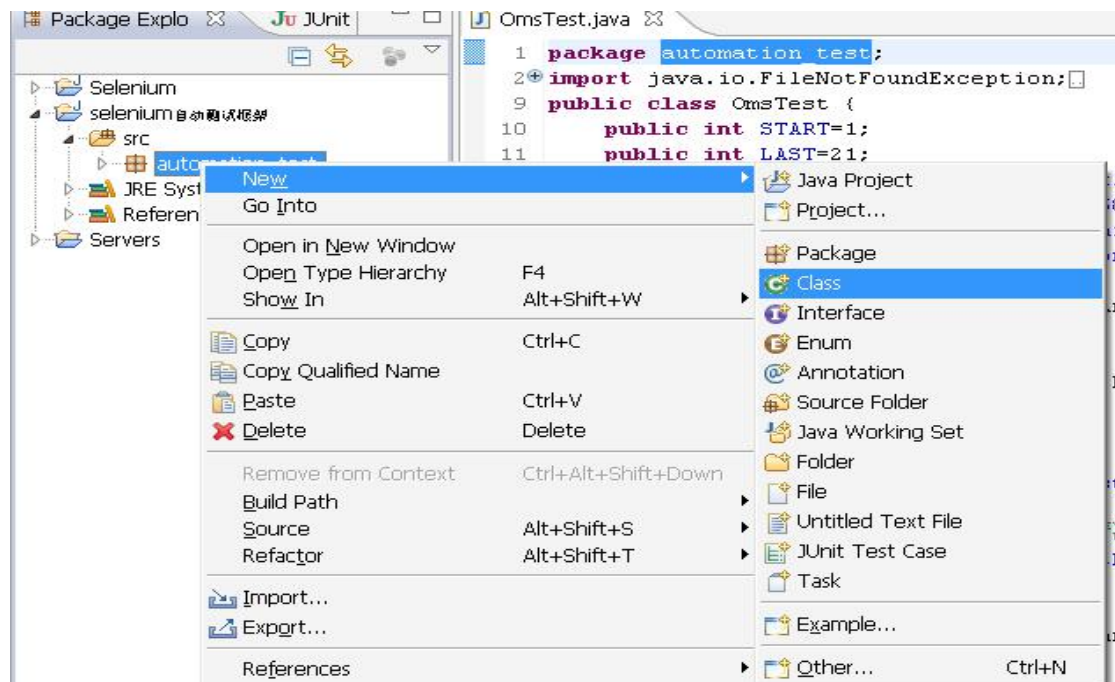
工程建好后, 如图首先建立一个主程序类的包



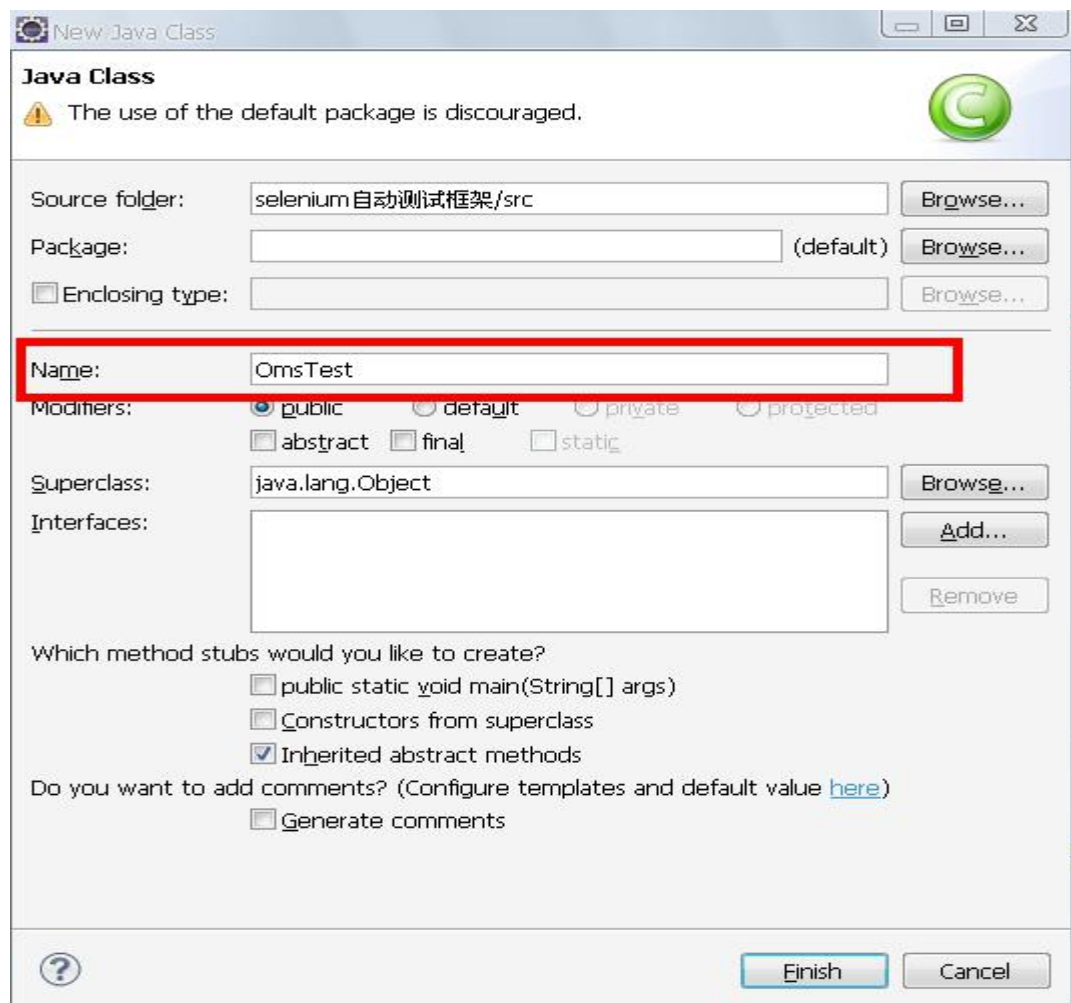
包的名字是 automation_test(如果改成其它名字，则对应的主程序引入的包名也要修改)



在包下新建一个类



类的名字是 OmsTest(如果改其他名字，对应的主程序 java 文件类的名字也要改)



将自动生成的代码清空，并将 OmsTest.java 中的代码拷贝进去即可
主程序代码有注释，大家可以看一下。

主要思路就是通过读取 excel 中的测试用例的参数，然后让 selenium 自动执行，并将执行结果再写入到测试用例，实现自动测试并统计测试结果的目的。

判断测试执行是否正确的依据是元素检查点，当一个测试用例的某个步骤出现与预期结果不一致的情形时，程序中止该条用例的执行，并将错误记录下来，继续进行下一条用例，直到所有用例执行结束。

2.2 编写自动化测试用例

2.2.1 自动化用例模板

我现在设计的用例模板的样式是这样的：

用例编号	用例说明	用例步骤说明	动作函数	动作参数1	动作参数2	动作参数3
1	新建子栏目	展开站点树+号	click	xpath= (//img[contains(@class, 'x-tree-elbow-end-plus')])[1]	无	无
1		点击selenium自动化测试栏目	click	//span[text()='selenium自动化测试']	无	无
1		右键弹出菜单	contextMenu	//span[text()='selenium自动化测试']	无	无
1		点击添加子栏目	click	//span[text()='添加子栏目']	无	无
1		光标点进栏目名输入框	click	//input[@name='nodeNet.node.name']	无	无
1		输入栏目名	type	//input[@name='nodeNet.node.name']	selenium9	无
1		光标点进栏目url输入框	click	//input[@name='nodeNet.node.urlName']	无	无
1		输入栏目url	type	//input[@name='nodeNet.node.urlName']	9	无
1		光标点进栏目序号输入框	click	//input[@name='nodeNet.ranking']	无	无
1		输入栏目序号	type	//input[@name='nodeNet.ranking']	9	无
1		点击保存按钮	click	//button[text()='保存']	无	无
1		点击是确认保存	click	//button[text()='是']	无	无
1		双击selenium自动化测试栏目	doubleClick	//span[text()='selenium自动化测试']	无	无

元素检查点	实际是否存在	测试是否通过	结果统计
//span[text()='selenium自动化测试']	是	是	第1, 条用例执行错误
无	未验证	未验证	
//span[text()='添加子栏目']	是	是	
//input[@name='nodeNet.node.name']	是	是	
无	未验证	未验证	
无	未验证	未验证	
无	未验证	未验证	
无	未验证	未验证	
无	未验证	未验证	
无	未验证	未验证	
//button[text()='是']	是	是	
无	未验证	未验证	
//span[text()='selenium6']	否	否	

字段解释：

用例编号：同一条用例的编号都设置为相同的，主要是为了当某一步执行错误时，中止该错误用例的继续执行，程序可以找出下一条用例的编号所在的 excel 行；

用例说明：同手工用例

用例步骤说明：基本类似手工用例

动作函数：执行某个操作时，所需要用到的 selenium 类方法，目前使用较多的是鼠标左键


```
<input type="button" name="save" value="另存为" />
</xml>
```

保存为 1.html 后，用 firefox 打开如下：



比如说我们要定位【测试输入】这个文本框，可以用以下的 xpath 语句：

```
//input[@name="testname"]
```

其中 // 表示在整个页面元素中查找，input 代表元素的标签，@name 表示这个元素的属性名，testname 表示属性值；

由此可见，xpath 的第一种定位方式即 //元素卷标名["@属性名"=属性值]。

<input name="testname" value="测试输入"/> 这段代码中，input 表示这个文本输入框的标签名，name 为属性名，testname 为属性值。

同理，还可以用 //input[@value="测试输入"] 来定位该输入框。

//input[@value="保存 1"] 定位“保存 1”这个按钮； //button[@type="button"] 定位 “保存 2” 这个按钮。

【思考】可否用 //input[@name="save"] 来定位“另存为”这个按钮？为什么，如果要定位“另存为”，该用什么表达式？

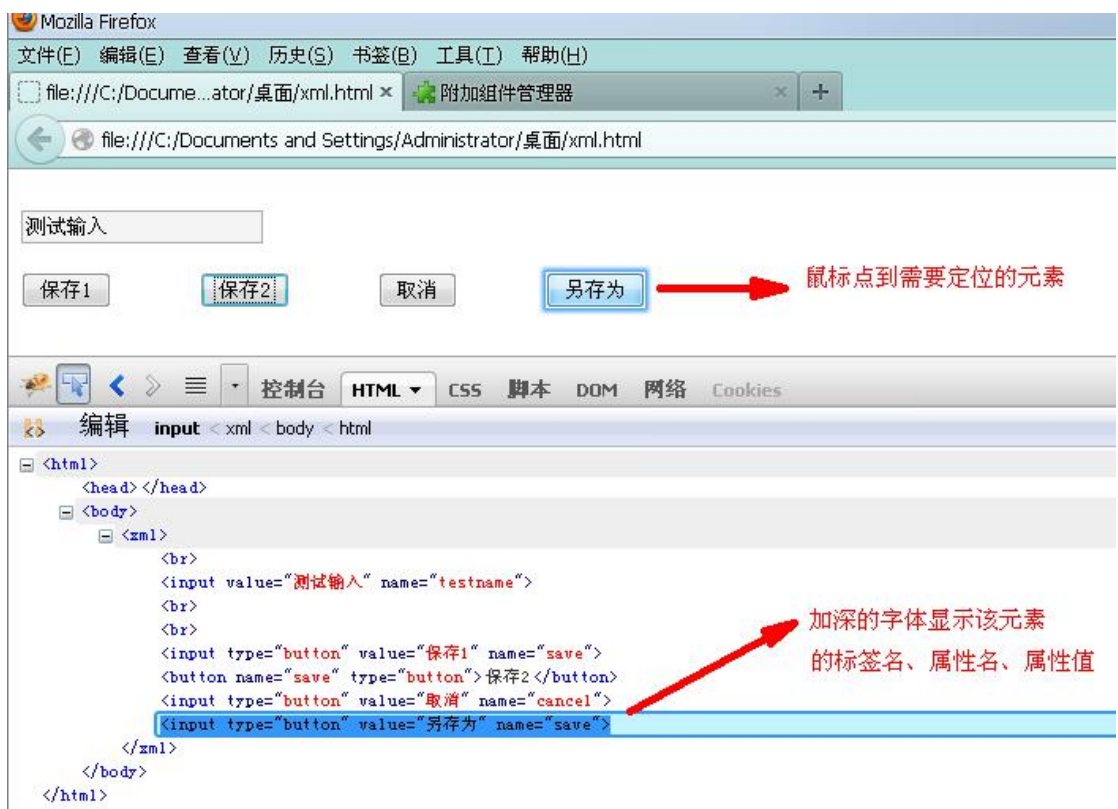
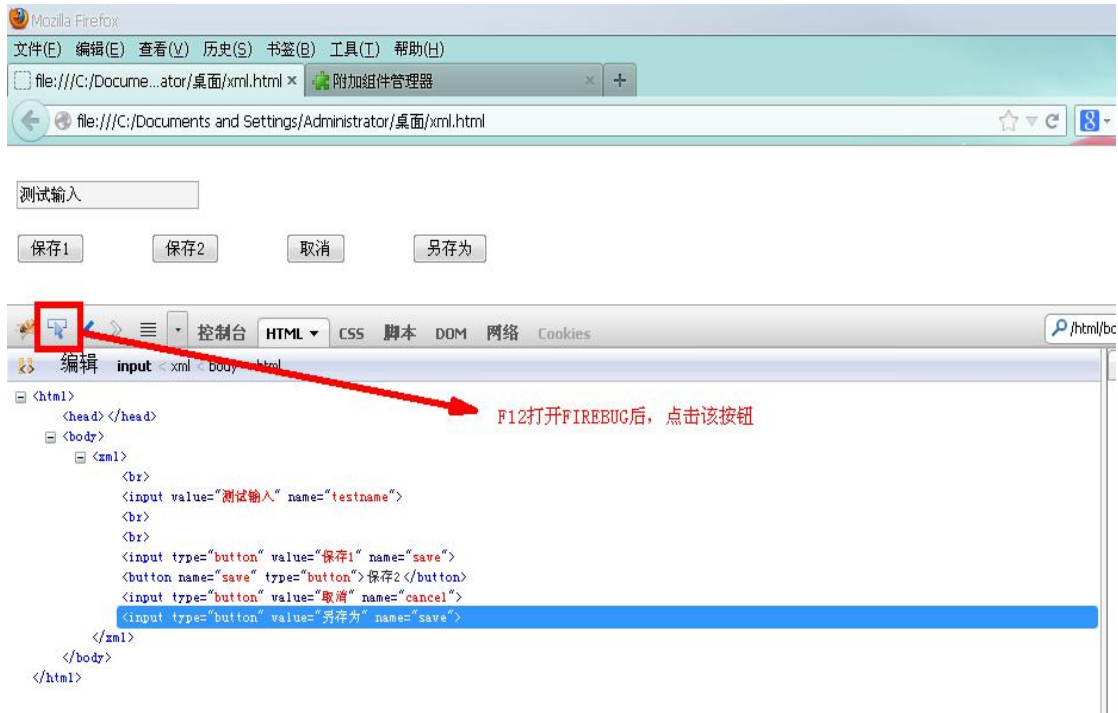
2.2.2.2 定位 xml 元素的辅助插件

综上所述，要定位页面的一个元素，只需找到一个//元素卷标名["@属性名"=属性值]的表达式可以唯一的指向该元素即可，关键是我们该如何判断一个元素卷标、属性名和属性值，以及如何确定该表达式确实可以正确指向该元素。需要用到 3 个 firefox 插件：Firebug、xpath checker 以及 selenium IDE。

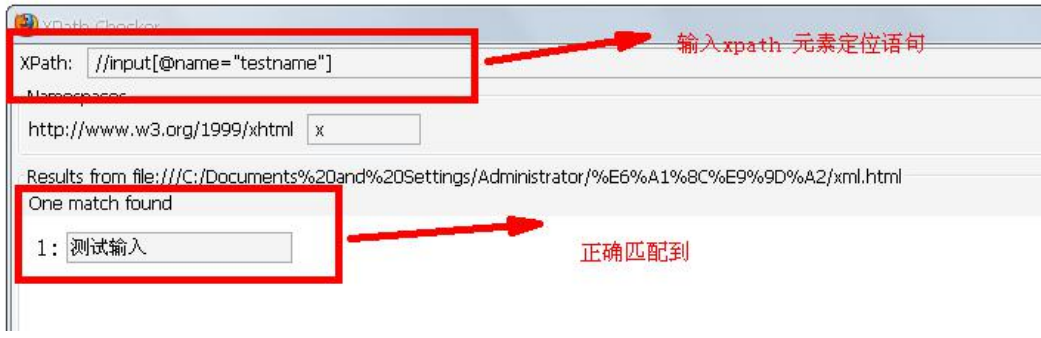
2.2.2.2.1 FireBUG 查找元素的标签名、属性名、属性值

Firebug 大家应该比较熟悉了，我们要用到它的“查看元素”功能，如果我们不清楚一个元素的卷标、属性或者属性值时，可以通过该功能进行查找。

具体使用步骤看截图

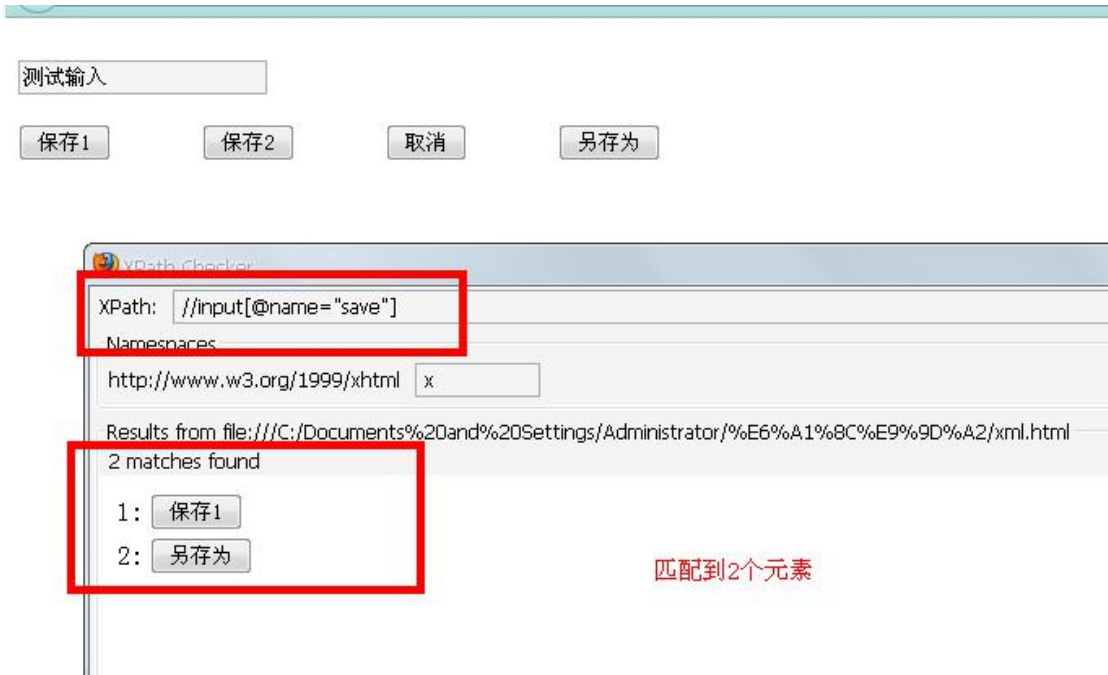


通过该按钮, 可以很方便的找到页面元素的各个属性, 为使用 xpath 语法定位元素创造基础。



可以看到//input[@name="testname"] 可以正确的定位到该文本框。

如果输入 `//input[@name="save"]`，看看是什么情况。



看看这两个元素的 xml 语言就知道为什么了。

```
<input type="button" name="save" value="保存 1" />
```

```
<input type="button" name="save" value="另存为" />
```

可以看到，如果以 `//input[@name="save"]` 作为 xpath 路径定位语句，由于这两个元素的标签都是“input”，且都存在 name 这个属性，而且属性值都是“save”，因此就会匹配出 2 个，这种情况下，就不能唯一的定位一个元素，如果要定位“保存 1”和“另存为”这两个元素，就需要采用另外的属性了。可以看到，两个元素虽然都有相同的 value 属性名，但是属性值是不同的，因此可以分别用 `//input[@value="保存 1"]`；`//input[@value="另存为"]` 来定位这两个元素，用 xpath checker 检查一下。



测试输入

保存1

保存2

取消

另存为

XPath Checker

XPath: `//input[@value='另存为']`

Namespaces

http://www.w3.org/1999/xhtml x

Results from file:///C:/Documents%20and%20Settings/Administrator/%E6%A1%8C%E9%9D%A2/xml

One match found

1: 另存为

定位成功

我们在做自动化测试时，要确保元素的 xpath 都只能指向唯一的一个，否则如果指向多个的话，程序将随机点击，那样自动测试的结果就不会准确。

另外，对于<button type="button" name="save">保存 2</button> 这种有文本内容在标签对之外的，可以直接使用`//button[text()='保存 2']`来进行定位，十分方便，特别是 OMS 系统中很多按钮的标签名、属性名、属性值都一样的，只有按钮名字不同，也只有用这个方式才能定位到唯一元素。

测试输入

保存1

保存2

取消

另存为

XPath Checker

XPath: `//button[text()='保存2']`

Namespaces

http://www.w3.org/1999/xhtml x

Results from file:///C:/Documents%20and%20Settings/Administrator/%E6%A1%8C%E9%9D%A2/xml.html

One match found

1: 保存2

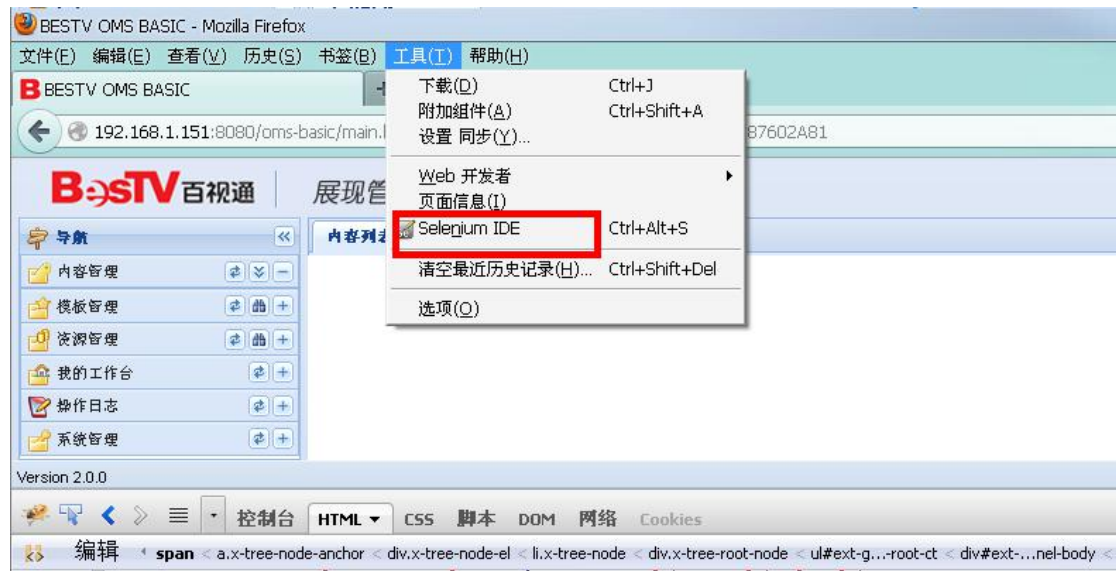
【注】`//标签名[text()='文本内容']` 这个 xpath 定位语句在编写用例中会经常用到，应该予以牢记。

2.2.2.2.3 Selenium IDE 进行单步调试

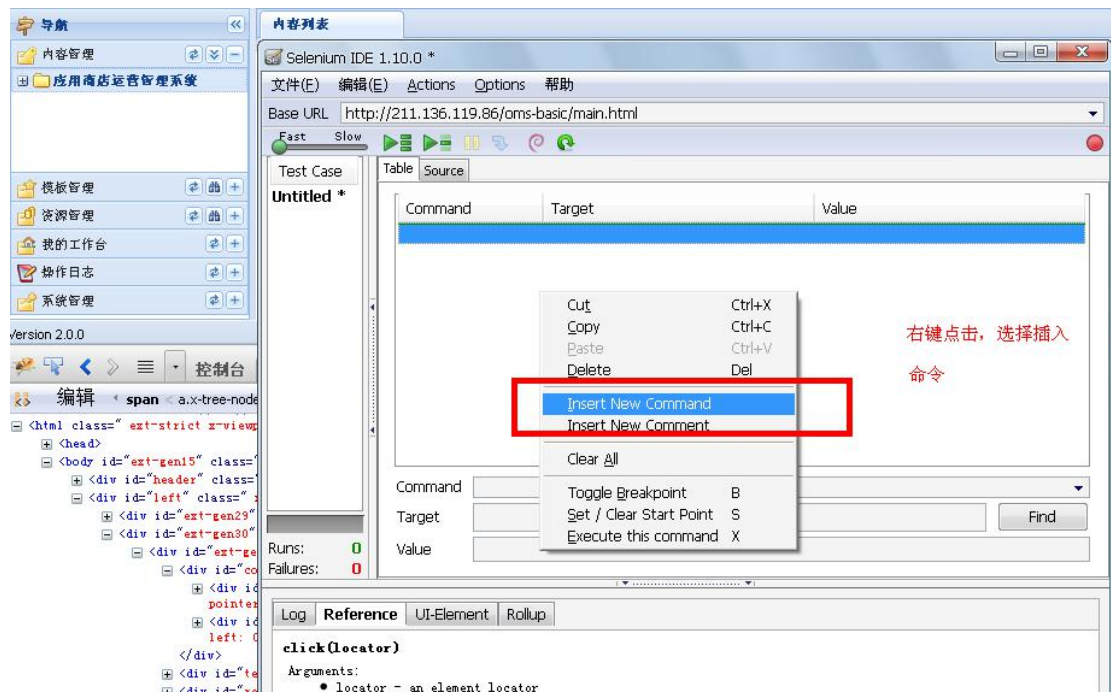
如何检验我们的动作函数及动作参数是否准确。可以用 selenium IDE 进行单步调试。

【注】selenium IDE 也可以进行录制，但回放一般都会由于元素找不到而失败，因此不采用 selenium IDE 录制用例，而采用 xpath 定位元素的办法手写用例。

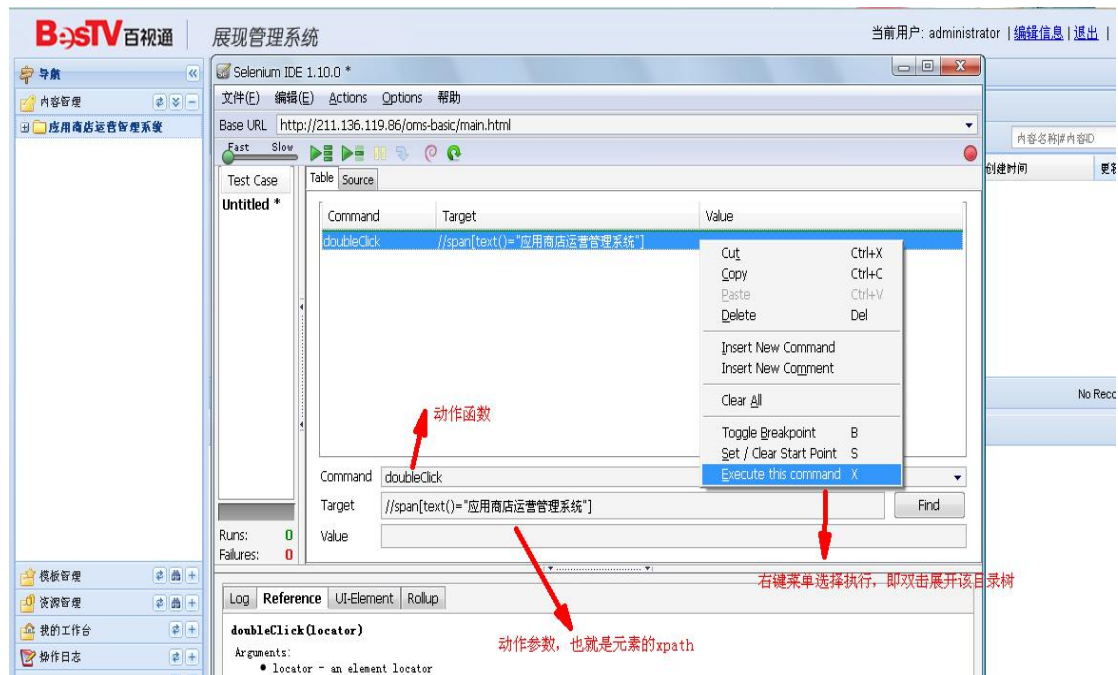
在需要进行调试的页面，点击 firefox 浏览器的 selenium IDE 插件



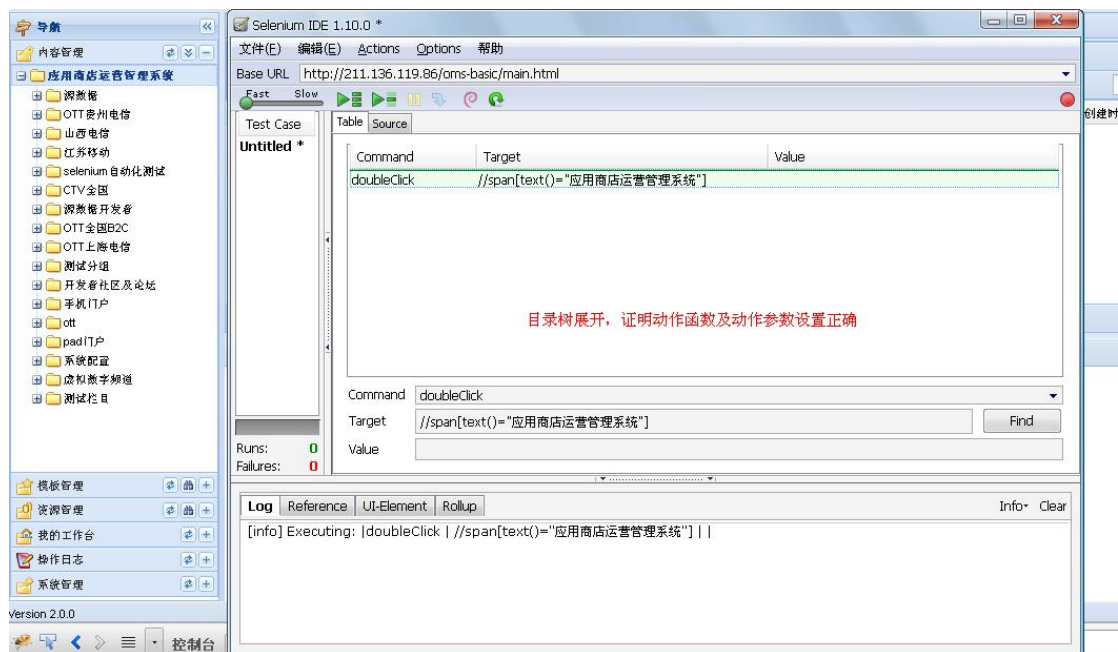
插入一条命令



输入需要调试的动作函数及动作参数



执行命令, 进行调试



3 OMS 自动化测试实例

简单介绍了下元素定位方法后, 就可以开始编写自动测试用例了, 下面以 OMS-basic->内容管理->【新建子栏目】为例, 开始编写并运行自动化测试用例。以百视通测试环境展现管理系统 <http://192.168.1.151:8080/oms-basic/main.html> 作为被测系统, 首先通过修改数据库, 取消登录动态验证码。【将 Oms_sso_bestv 数据库 system_config 表的 authType 字段值改为 1, 重启 tomcat 服务即可】

3.1 测试系统初始化

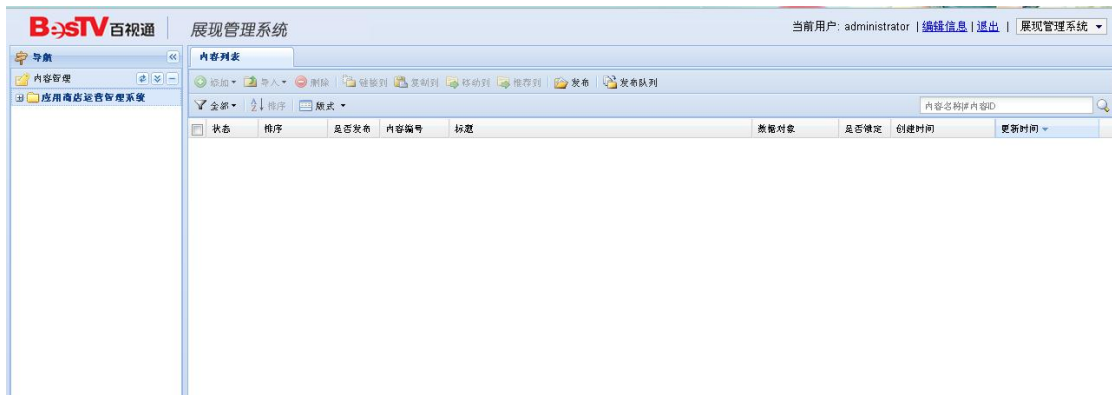


用户登录页面已在主程序中进行了初始化，可参考 OmsTest.java 的 10-16 行

```
public class OmsTest {
10     public int START=1; //用例步骤起始行 (excel表总行数-表头1行)
11     public int LAST=21; //用例步骤结束行 (excel表总行数-表头1行)
12     public String testCasePath="D:/oms.xls"; //用例存放路径
13     public String testUrl="http://192.168.1.151:8080/"; //测试网站的根路径
14     public String testSys="/oms-basic/main.html"; //测试系统的首页
15     public String username="administrator"; //用户名
16     public String password="Oms@2010"; //密码
}
```

正式写用例之前先把测试环境的值填好，写完用例以后再把测试用例存放路径以及需要执行的测试用例的起始行及结束行填进去即可。

【注】正式编写用例将以 administrator 刚登陆进去以后的页面为初始页面，即：



3.2 编写自动化测试用例

打开 excel 自动测试用例模板，将用例步骤细化，在该用例模板中，以下字段为必填项：

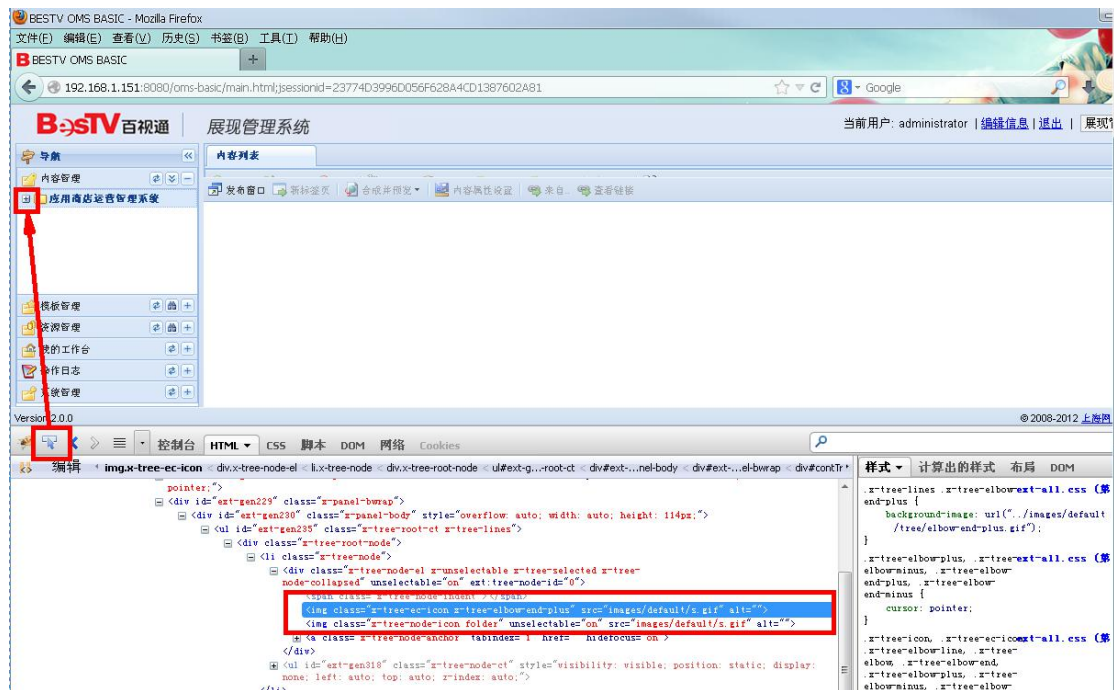
【用例编号】、【动作函数】、【动作参数 1】、【动作参数 2】、【动作参数 3】、【元素检查点】

以在 应用商店运营管理系统->selenium 自动化测试 栏目下“添加子栏目”为例：

- 第 1 步：展开“应用商店运营管理系统”目录树(因为刚登陆进去的时候该根目录树是折叠的)。

展开目录树有两种方法 1:点击前面的+ 号；2：双击该栏目。

如果选择第 1 种方式，则【动作函数】字段填写 click,【动作参数 1】填写“+”的 xpath 路径，采用 FIREBUG 的查看元素功能查看该元素的标签及属性。



可以看到+号的元素标签及属性如下：

```

```

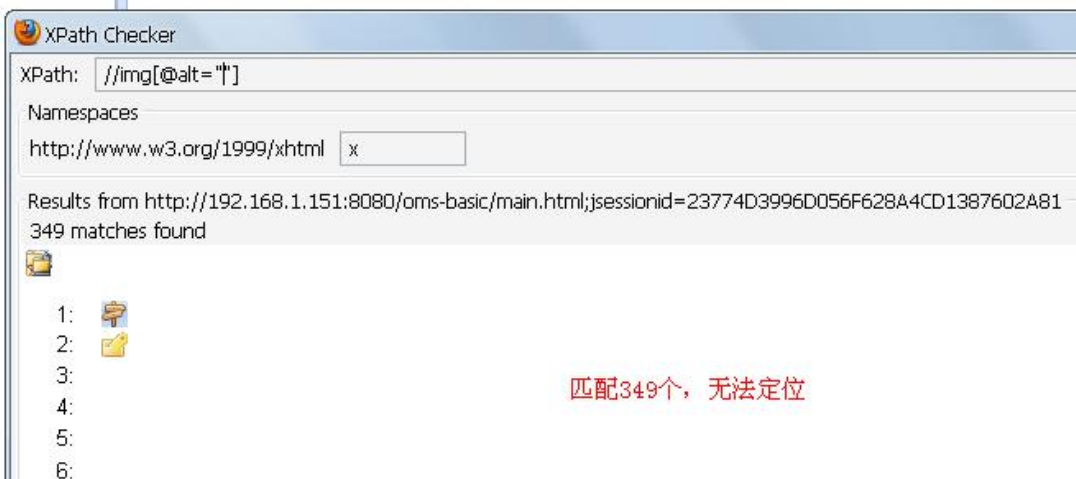
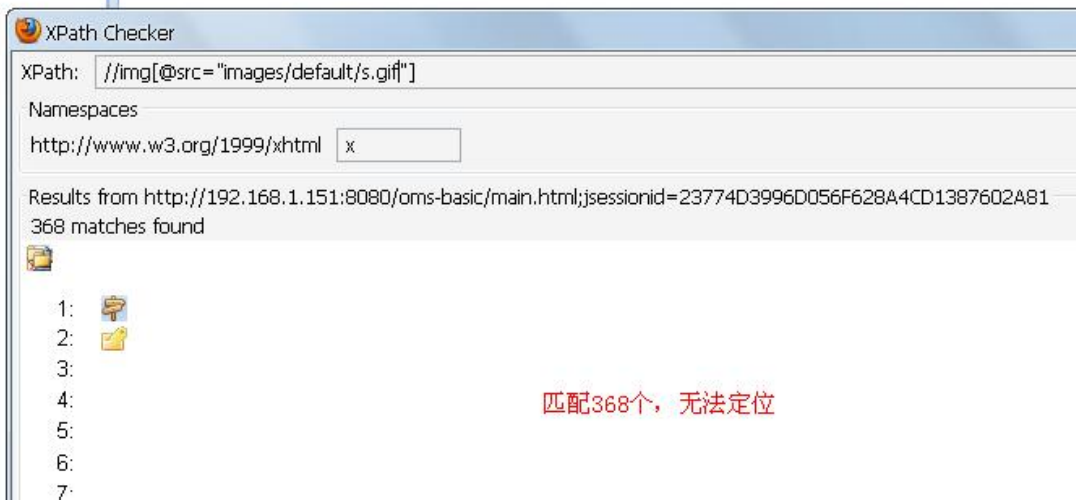
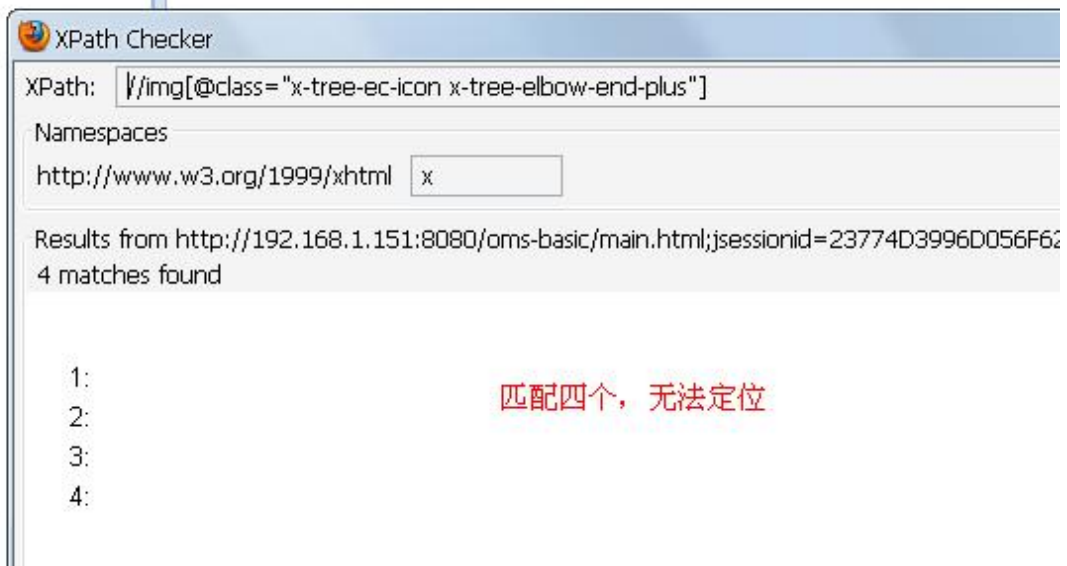
如果按照//元素标签[@属性名="属性值"]的 xpath 语法的话，该元素有三种 xpath 定位写法，分别是：

```
//img[@class=" x-tree-ec-icon x-tree-elbow-end-plus"]
```

```
//img[@src=" images/default/s.gif"]
```

```
//img[@alt=""]
```

分别使用 xpath checker 进行判断：



再看看+号 标签对

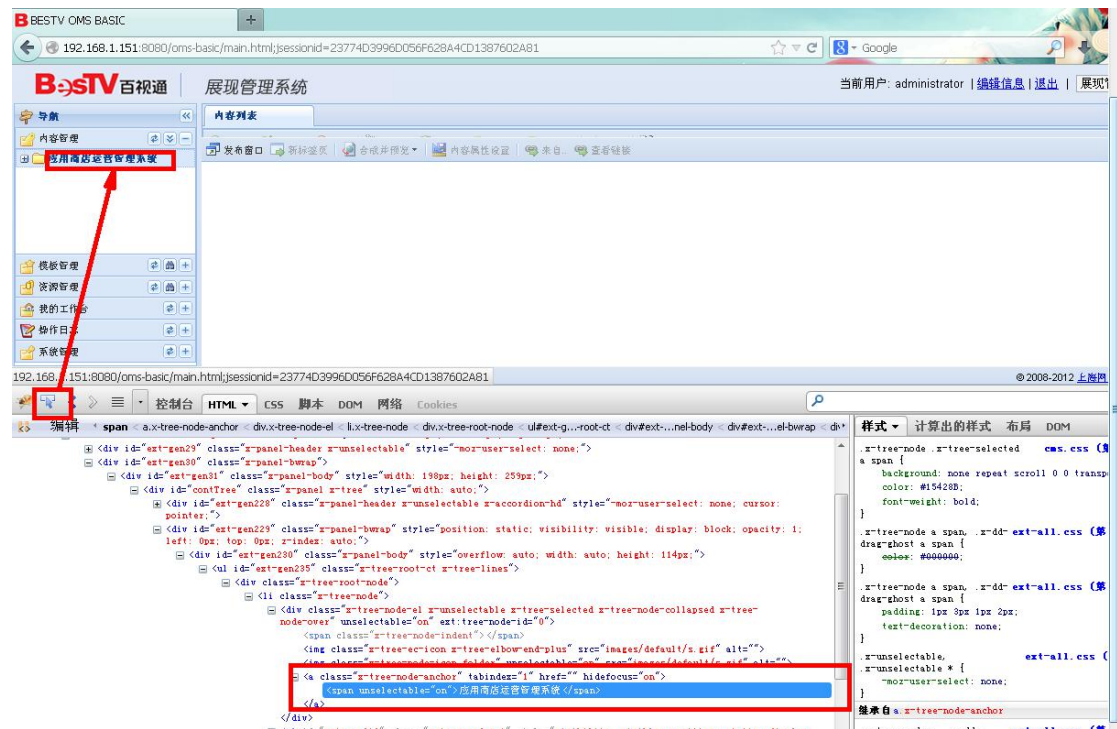
``

之间也没有文本，不能用`//元素标签名[text()='文本']`来进行定位

因此该 + 号的定位就有点难度。我们采用第二种方式展开根目录树，即鼠标双击。

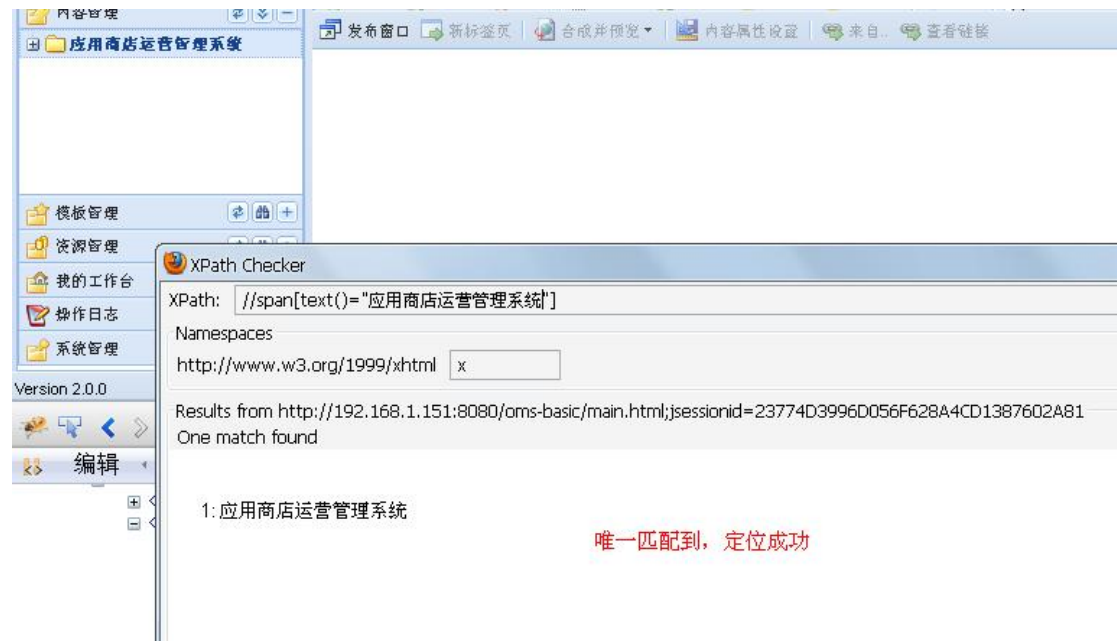
鼠标双击的【动作函数】是 `doubleClick`,我们只要找到“应用商店运营管理系统”的 `xpath` 路径作为【动作参数 1】即可。

使用 FIREBUG 查看该元素的标签属性：



`应用商店运营管理系统`

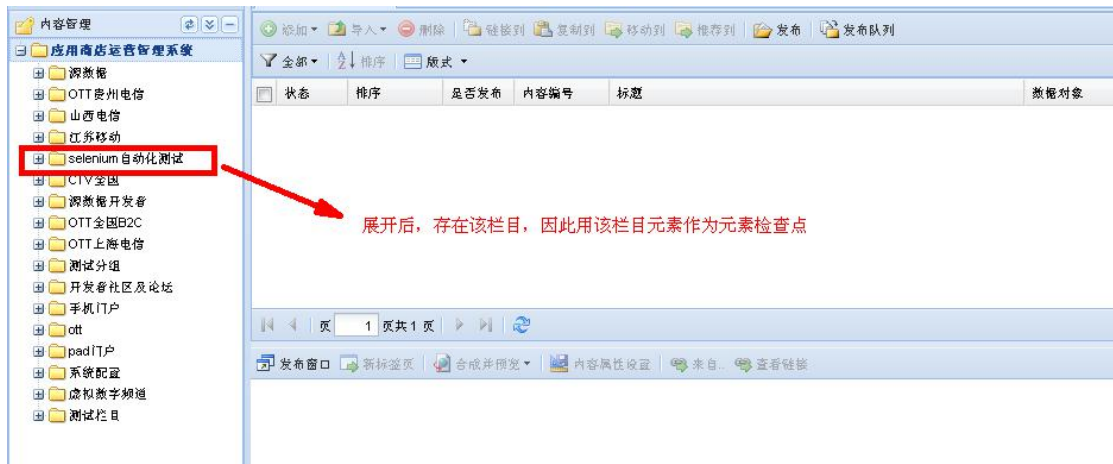
采用 `//元素标签名[text()='文本']` 来尝试定位该元素



因此在自动化用例模板里面第一步可以这样写：

用例编号	用例说明	用例步骤说明	动作函数	动作参数1	动作参数2	动作参数3	元素检查点
1	新建子栏目	展开站点树	<code>doubleClick</code>	<code>//span[text()='应用商店运营管理系统']</code>	无	无	<code>//span[text()='seLEnium自动化测试']</code>

元素检查点中存放的也是 xpath，我们这一步检查的是【selenium 自动化测试】这个栏目，因为展开站点树后，实际是存在这个栏目的。如果执行展开站点树后，没有找到该栏目，则程序执行失败。



【注】：

1 因为 doubleClick 动作只有 1 个参数，因此【动作参数 2】【动作参数 3】填写“无”，以便让程序找到动作入口。如果后面的【动作参数 2】【动作参数 3】不填的话，程序会报空指针异常，这是下一步优化的方向。

2 【实际是否存在】【测试是否通过】【统计结果】这三栏不需要填写，程序执行后会自动统计结果

以此类推展开树以后，需要右键点击“selenium 自动化测试”，点击“添加子栏目”，输入必填信息，点击保存。每一步根据需要设置检查点。(完整的测试用例见附件 oms.xls)

用例编号	用例说明	用例步骤说明	动作函数	动作参数1	动作参数2	动作参数3	元素检查点
1	新建子栏目	展开站点树	doubleClick	//span[text()='应用商店运营管理系统']	无	无	//span[text()='selenium自动化测试']
1		点击selenium自动化测试栏目	click	//span[text()='selenium自动化测试']	无	无	无
1		右键弹出菜单	contextMenu	//span[text()='selenium自动化测试']	无	无	//span[text()='添加子栏目']
1		点击添加子栏目	click	//span[text()='添加子栏目']	无	无	//input[@name='nodeNet.node.name']
1		光标点进栏目名输入框	click	//input[@name='nodeNet.node.name']	无	无	无
1		输入栏目名	type	//input[@name='nodeNet.node.name']	selenium9	无	无
1		光标点进栏目url输入框	click	//input[@name='nodeNet.node.urlName']	无	无	无
1		输入栏目url	type	//input[@name='nodeNet.node.urlName']	9	无	无
1		光标点进栏目序号输入框	click	//input[@name='nodeNet.ranking']	无	无	无
1		输入栏目序号	type	//input[@name='nodeNet.ranking']	9	无	无
1		点击保存按钮	click	//button[text()='保存']	无	无	//button[text()='是']
1		点击是确认保存	click	//button[text()='是']	无	无	无
1		双击selenium自动化测试栏目	doubleClick	//span[text()='selenium自动化测试']	无	无	//span[text()='selenium6']

3.3 自动化测试用例执行

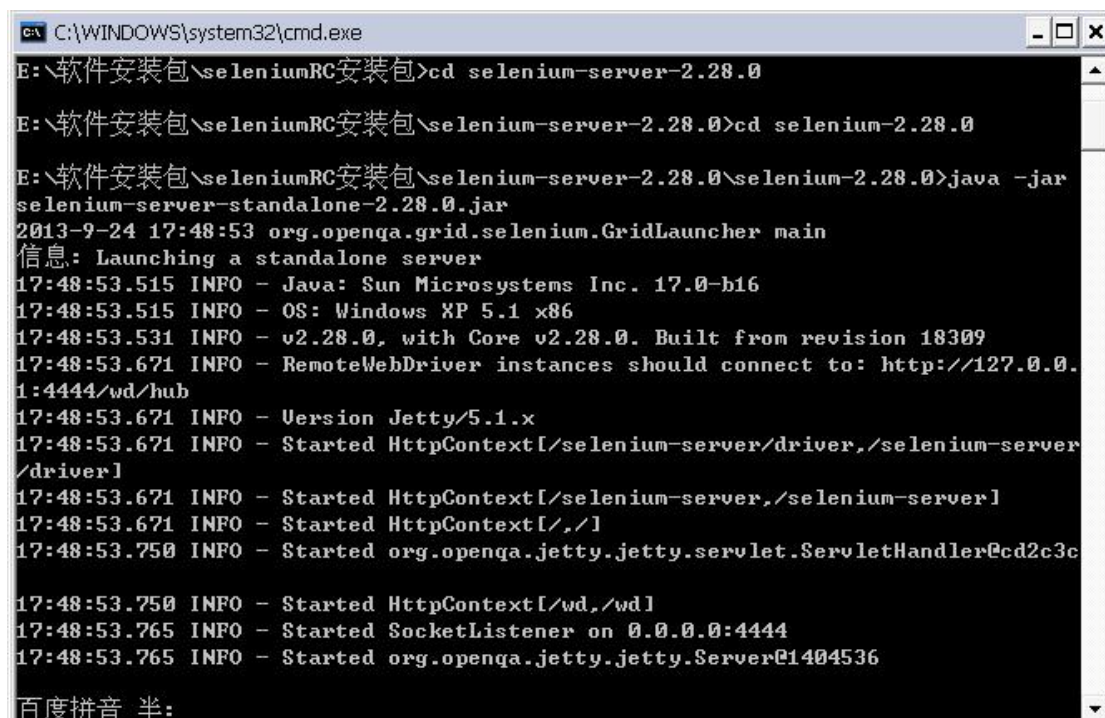
3.3.1 启动 selenium 服务

在桌面新建个 txt 文档，输入以下命令，另存为.bat 格式，并双击。

```
e:
cd 软件安装包
cd seleniumRC 安装包
cd selenium-server-2.28.0
cd selenium-2.28.0
java -jar selenium-server-standalone-2.28.0.jar
```

注意：该批处理命令主要是找到 selenium-server-standalone-2.28.0.jar 的本地存放路径，上面的代码是我机器上的存放路径，大家根据自己的实际存放路径要进行修改。

改好后双击出现如下窗口代表服务成功启动：



```
C:\WINDOWS\system32\cmd.exe
E:\软件安装包\seleniumRC安装包>cd selenium-server-2.28.0
E:\软件安装包\seleniumRC安装包\selenium-server-2.28.0>cd selenium-2.28.0
E:\软件安装包\seleniumRC安装包\selenium-server-2.28.0\selenium-2.28.0>java -jar
selenium-server-standalone-2.28.0.jar
2013-9-24 17:48:53 org.openqa.grid.selenium.GridLauncher main
信息: Launching a standalone server
17:48:53.515 INFO - Java: Sun Microsystems Inc. 17.0-b16
17:48:53.515 INFO - OS: Windows XP 5.1 x86
17:48:53.531 INFO - v2.28.0, with Core v2.28.0. Built from revision 18309
17:48:53.671 INFO - RemoteWebDriver instances should connect to: http://127.0.0.
1:4444/wd/hub
17:48:53.671 INFO - Version Jetty/5.1.x
17:48:53.671 INFO - Started HttpContext[/selenium-server/driver,/selenium-server
/driver]
17:48:53.671 INFO - Started HttpContext[/selenium-server,/selenium-server]
17:48:53.671 INFO - Started HttpContext[/,/]
17:48:53.750 INFO - Started org.openqa.jetty.jetty.servlet.ServletHandler@c1404536
17:48:53.750 INFO - Started HttpContext[/wd,/wd]
17:48:53.765 INFO - Started SocketListener on 0.0.0.0:4444
17:48:53.765 INFO - Started org.openqa.jetty.jetty.Server@c1404536
百度拼音 半:
```

3.3.2 初始化主程序

进入 eclipse ，找到工程里面的 OmsTest.java 主程序文件，根据实际情况进行如下修改

```

1 package automation_test;
2 import java.io.FileNotFoundException;
9 public class OmsTest {
10     public int START=1; //用例步骤起始行 (excel表总行数-1)
11     public int LAST=13; //用例步骤结束行 (excel表总行数-1)
12     public String testCasePath="D:/oms.xls" ; //用例存放路径
13     public String testUrl="http://192.168.1.151:8080/"; //测试网站的根路径
14     public String testSys="/oms-basic/main.html"; //测试系统的首页
15     public String username="administrator"; //用户名
16     public String password="Oms@2010"; //密码
17     ExcelReadAndWrite xls=new ExcelReadAndWrite();
18     private Selenium selenium;
19     //定义一个步骤的操作方法
20     public void step(int stepId) throws FileNotFoundException, IOException, Interrupte

```

【注】：

1 因为我们执行的是第一条用例，因此我们要找出在 oms.xls 中，第一条用例的起始行及结束行。

	A	D	C
1	用例编号	用例说明	用例步骤说明
2	1	新建子栏目	展开站点树
3	1		点击selenium自动化测试栏
4	1		右键弹出菜单
5	1		点击添加子栏目
6	1		光标点进栏目名输入框
7	1		输入栏目名
8	1		光标点进栏目url输入框
9	1		输入栏目url
10	1		光标点进栏目序号输入框
11	1		输入栏目序号
12	1		点击保存按钮
13	1		点击是确认保存
14	1		双击selenium自动化测试栏

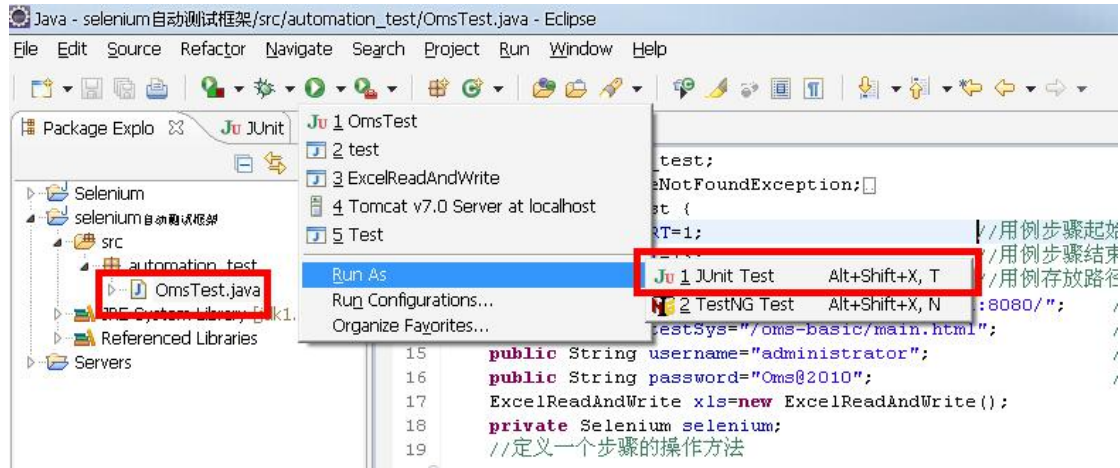
因为 excel 表格是从 0 开始计数的，因此，第一条用例的行数范围是 1-13.

因此 START=1 LAST=13

2 用例存放路径要根据自己的实际情况填写，我是放在 d:根目录下，因此目录是 d:/oms.xls , (注意分隔符斜线的方向不要弄反了)

3.3.3 执行用例

在 OmsTest.java 页面，选择 以 junit Test 方式运行。



执行过程中，selenium 会调用 firefox，并出现执行界面，执行过后会自动将执行结果写入测试用例相关单元格

E	F	G	H	I	J	K
动作参数1	动作参数2	动作参数3	元素检查点	实际是否存在	测试是否通过	结果统计
//span[text()='应用商店运营管理系统']	无	无	//span[text()='selenium自动化测试']			
//span[text()='selenium自动化测试']	无	无	无			
//span[text()='selenium自动化测试']	无	无	//span[text()='添加子栏目']			
//span[text()='添加子栏目']	无	无	//input[@name='nodeNet.node.name']			
//input[@name='nodeNet.node.name']	无	无	无			
//input[@name='nodeNet.node.name']	selenium1	无	无			
//input[@name='nodeNet.node.urlName']	无	无	无			
//input[@name='nodeNet.node.urlName']	1	无	无			用例执行前
//input[@name='nodeNet.ranking']	无	无	无			
//input[@name='nodeNet.ranking']	1	无	无			
//button[text()='保存']	无	无	//button[text()='是']			
//button[text()='是']	无	无	无			
//span[text()='selenium自动化测试']	无	无	//span[text()='selenium1']			

动作参数1	动作参数2	动作参数3	元素检查点	实际是否存在	测试是否通过	结果统计
//span[text()='应用商店运营管理系统']	无	无	//span[text()='selenium自动化测试']	是	是	所选用例执行成功!
//span[text()='selenium自动化测试']	无	无	无	未验证	未验证	
//span[text()='selenium自动化测试']	无	无	//span[text()='添加子栏目']	是	是	
//span[text()='添加子栏目']	无	无	//input[@name='nodeNet.node.name']	是	是	
//input[@name='nodeNet.node.name']	无	无	无	未验证	未验证	
//input[@name='nodeNet.node.name']	selenium1	无	无	未验证	未验证	
//input[@name='nodeNet.node.urlName']	无	无	无	未验证	未验证	
//input[@name='nodeNet.node.urlName']	1	无	无	未验证	未验证	用例执行后
//input[@name='nodeNet.ranking']	无	无	无	未验证	未验证	
//input[@name='nodeNet.ranking']	1	无	无	未验证	未验证	
//button[text()='保存']	无	无	//button[text()='是']	是	是	
//button[text()='是']	无	无	无	未验证	未验证	
//span[text()='selenium自动化测试']	无	无	//span[text()='selenium1']	是	是	

并且通过查看 oms 系统，确实在 应用商店运营管理系统->selenium 自动化测试 栏目下新创建了栏目 selenium1，且 URL 名称、序号与自动化测试用例中设置的值一致



光标点进栏目url输入框	click	//input[@name="nodeNet.node.urlName"]	无	无	无
输入栏目url	type	//input[@name="nodeNet.node.urlName"]	1	无	无
光标点进栏目序号输入框	click	//input[@name="nodeNet.ranking"]	无	无	无
输入栏目序号	type	//input[@name="nodeNet.ranking"]	1	无	无
点击保存按钮	click	//button[text()='保存']	无	无	//button[t

4 重点优化方向

目前该框架是在单机上执行，部署较为繁琐，而且测试执行初始化的时候需要修改底层 JAVA 源代码，维护不太方便，因此下一步打算做成 B/S 架构，部署到服务器上，通过页面导入 excel 测试用例，执行完以后通过预先指定的目录，再将测试结果自动导出到本地。将自动化测试的初始化配置过程实行可视化操作，方便测试人员。

当然，目前的框架也存在其它不尽合理的地方，欢迎各位同事提出合理意见，大家一起讨论完善，为公司的自动化测试开启新的一页作出自己的努力。