

WebDriver 自动化实战



我们即将学到啥

- Webdriver
- TestNG
- Maven
- Tomcat
- Jenkins
- Page Object
- Robot Framework
- Cucumber
- Log4j
- POI

几点要求

- 上课认真听，不急着实践
- 上课不迟到
- 每次课后都把课上的内容自己完成

Web UI 自动化基础入门 day1

- 课程背景介绍：为什么我们要学自动化
- 初识Selenium
- Java 开发环境的搭建
- Selenium 2环境配置
- TestNG 框架介绍
- TestNG 部分常用注解介绍
- TestNG 部分常用校验介绍
- Q&A



课程背景介绍

场景：

面试官：接触过敏捷模式吗？

我：接触过。。。。

面试官：敏捷测试中如何进行快速测试？

我：组长分配任务。。。

面试官：你们不用自动化测试吗？

我：或许。。有可能。。没用过？ Balabala

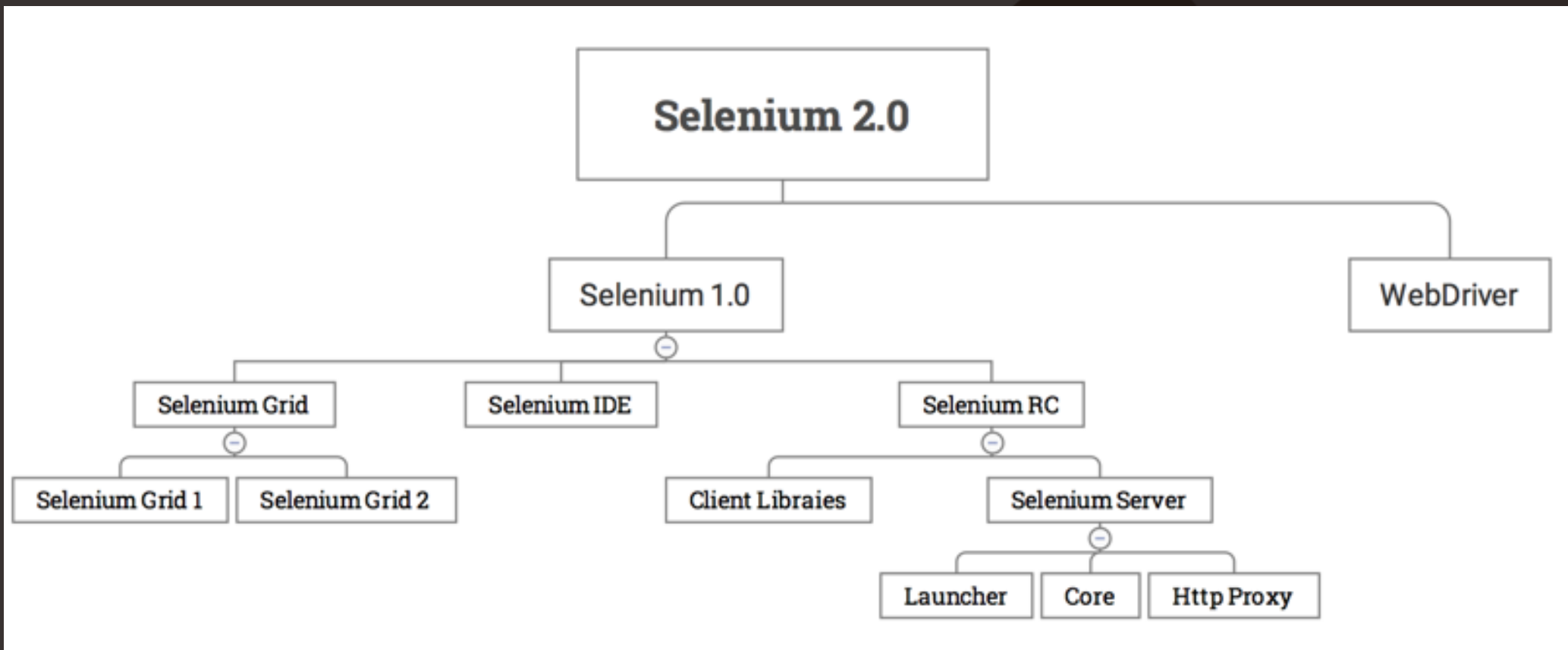
自动化测试最近几年在测试人员中很火，出去面试时，要是说不会自动化测试，都能感觉到面试官的鄙夷之情溢于言表，甚至在公司做自动化测试的同事都感觉比做手工测试的有技术含量一些，正因为有这样的情怀，所以导致自动化测试越来越火，但是火的背后，却伴随着很多的盲目，盲目的追求自动化，盲目的夸大自动化的作用，甚至投入很多的人力物力去做自动化，但往往最后的效果并不太理想，失败的案例太多太多，于是，我们是时候停下来去冷静的思考一下，同时提出：做高质量的自动化测试。

厦门云层天咨软件技术有限公司



Xiamen cloudits Consulting Software Technology Co. Ltd.

初识Selenium



Selenium 3.0 主要变更

1. 移除Selenium RC
2. Firefox 和 Safari 推出了自己的driver（geckodriver,Safaridriver）
3. Selenium 3 支持IE 9+， Selenium 2支持IE 7到11
4. 全面拥抱Java 8

Java 环境搭建

1. 下载JDK

2. 安装JDK

3. 配置Java 环境

1. 设置JAVA_HOME 环境变量：安装路径
2. 设置Path值： %JAVA_HOME%\bin;%JAVA_HOME%\jre\bin;
3. 设置CLASSPATH: .;%JAVA_HOME%\lib;%JAVA_HOME%\lib\tools.jar

4. 校验是否安装完成

1. 控制台输入: java -version

5. 安装IDEA集成开发环境

Selenium 环境搭建

Maven 介绍

简单来说，Maven是一个项目管理工具，我们可以通过Maven来对项目进行构建打包等。Maven来管理项目所需的jar包。

通过IDEA新建Maven工程：

1. IDEA新建Maven工程
2. Pom.xml文件引入Selenium所需包

TestNG 简介

1. 简介

是一个强大的测试框架，设计灵感来源于junit，但优于junit，它提供了很强大的注解，便于我们对case的各种操作。

2. TestNG 给我提供了啥：

- 提供强大的注释，方便测试人员的使用。
- 支持数据驱动测试（DDT）
- 支持并行测试
- 可以灵活配置测试，强大的执行模式
- 可生成多种测试报告
- 等等



TestNG 引入

项目引入:

Pom.xml 添加如下:

```
<dependency>  
  <groupId>org.testng</groupId>  
  <artifactId>testng</artifactId>  
  <version>6.9.10</version>  
</dependency>
```

TestNG 部分注解介绍

@BeforeTest:

注解的方法将被运行之前的任何测试方法属于内部类的 `<test>` 标签的运行。

@BeforeMethod:

注解的方法将每个测试方法之前运行。

@Test:

标记一个类或方法作为测试的一部分。

@AfterMethod:

被注释的方法将被运行后，每个测试方法。

@AfterTest:

注解的方法将被运行后，所有的测试方法，属于内部类的 `<test>` 标签的运行。

TestNG 部分常用校验方式介绍

1. 相等:

`Assert.assertEquals();`

2. 不等:

`Assert.assertNotEquals();`

3. 不为空:

`Assert.assertNotNull();`

4. 为空

`Assert.assertNull();`



为企业培养定制化
人员！



就业培训

为学校构建适应企
业的课程体系！



高校实训

千里之外也能获得
专业的培训指导！



在线教育

联系方式:

地 址: 厦门市思明区前埔西路146号 (云层天咨培训中心)

联 系 人: 陈霁、方文庆

联系电话: 0592-8265163/18602195793

邮 箱: wqfang@cloudits.info

请多多关照! 谢谢!

Web UI 自动化基础入门 day2

- 启动Firefox 浏览器
- 启动Chrome 浏览器
- 启动IE 浏览器
- 启动Edge 浏览器
- 关闭浏览器
- 常用浏览器操作



启动Firefox 浏览器

- **WebDriver** 在启动浏览器时，都是启动一个干净的没有任何插件及cookies信息的浏览器
- 启动firefox浏览器:

```
@Test
public void openFireFoxTest() {
    // 启动firefox浏览器
    WebDriver driver = new FirefoxDriver();
}
```

- 启动不在默认安装路径的firefox浏览器:

```
@Test
public void openFireFoxTest() {
    // 指定firefox 安装路径
    System.setProperty("webdriver.firefox.bin", "C:\\Program Files (x86)\\Mozilla Firefox\\firefox.exe");
    // 启动firefox浏览器
    WebDriver driver = new FirefoxDriver();
}
```



启动Chrome浏览器

1. 需要chromedriver.exe

下载地址: <https://sites.google.com/a/chromium.org/chromedriver/downloads>

2. 工程根目录下新建drivers文件夹，并把chromedriver.exe放入

3. 启动Chrome 浏览器

```
@Test
public void openChromeTest() {
//    指定chromedriver.exe路径
    System.setProperty("webdriver.chrome.driver", ".\\drivers\\chromedriver.exe");
//    启动Chrome浏览器
    WebDriver driver = new ChromeDriver();
}
```

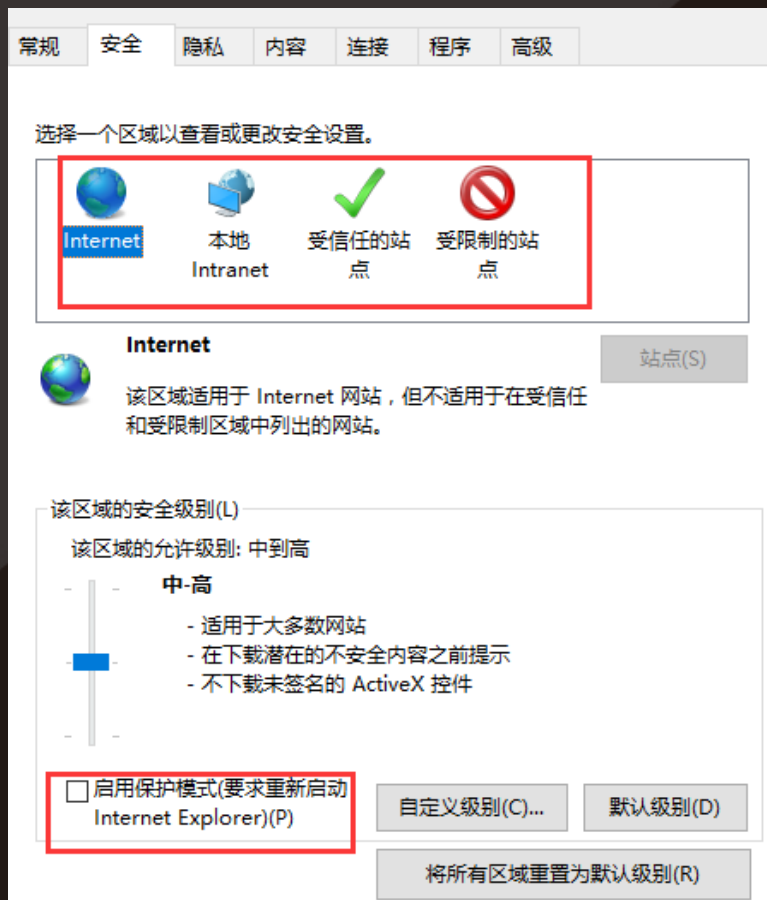
启动IE浏览器

1. 需要IEDriverServer.exe (<http://selenium-release.storage.googleapis.com/index.html>)
IE的exe文件分64位与32位，请根据自己的机器选择相应的exe文件
2. 启动代码

```
@Test
public void openIETest() {
    // 指定IEDriverServer.exe 路径
    System.setProperty("webdriver.ie.driver", ".\\drivers\\IEDriverServer.exe");
    // new 一个InternetExplorerDriver对象, 启动IE浏览器
    WebDriver driver = new InternetExplorerDriver();
}
```

启动IE浏览器

1. `.selenium.remote.SessionNotFoundException: Unexpected error launching Internet Explorer. Protected Mode settings are not the same for all zones. Enable Protected Mode must be set to the same value (enabled or disabled) for all zones. (WARNING: The server did not provide any stacktrace information)`



启动Edge浏览器

1. 控制台输入“ver”查看版本系统版本:

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.10586]
(c) 2015 Microsoft Corporation。保留所有权利。
C:\Users\Administrator.PC-20150712SUJN>ver
Microsoft Windows [版本 10.0.10586]
C:\Users\Administrator.PC-20150712SUJN>a
```

2. 根据系统版本下载对应的driver:

<https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/>

3. 启动浏览器:

```
@Test
public void openEdgeTest() {
    // 指定MicrosoftWebDriver路径
    System.setProperty("webdriver.edge.driver", ".\\drivers\\MicrosoftWebDriver.exe");
    // 启动Edge浏览器
    WebDriver driver = new EdgeDriver();
}
```

关闭浏览器

1. 关闭当前窗口

```
driver.close();
```

2. 关闭所有窗口并退出

```
driver.quit();
```

浏览器常用操作

1. 打开网页:

// 打开某个网页, http不可少

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

// 打开某个网页, 等同于上面的get方法

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

2. 浏览器后退

```
driver.navigate().back();
```

3. 浏览器前进

```
driver.navigate().forward();
```

浏览器常用操作

1. 浏览器刷新

```
driver.navigate().refresh();
```

2. 浏览器最大化

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

3. 设置浏览器大小

```
Dimension dimension = new Dimension(900, 800);  
driver.manage().window().setSize(dimension);
```

4. 获取当前页面URL

```
driver.getCurrentUrl();
```

5. 获取当前页面Title

```
driver.getTitle();
```



为企业培养定制化
人员！



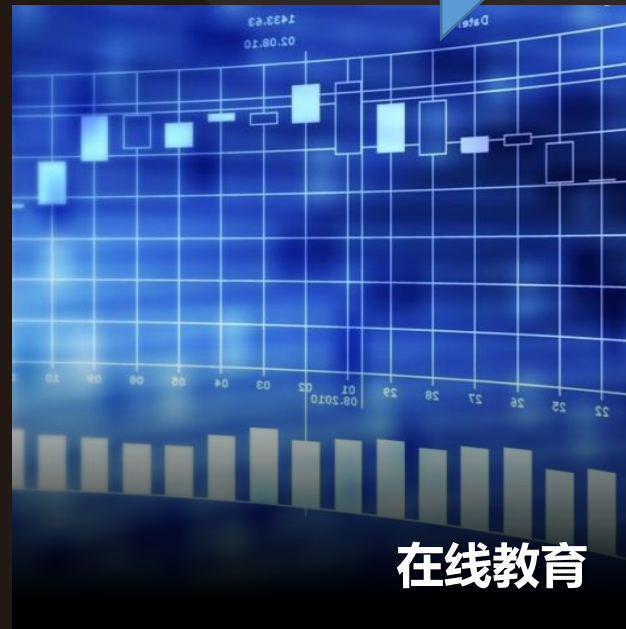
就业培训

为学校构建适应企
业的课程体系！



高校实训

千里之外也能获得
专业的培训指导！



在线教育

联系方式:

地 址: 厦门市思明区前埔西路146号 (云层天咨培训中心)

联 系 人: 陈霁、方文庆

联系电话: 0592-8265163/18602195793

邮 箱: wqfang@cloudits.info

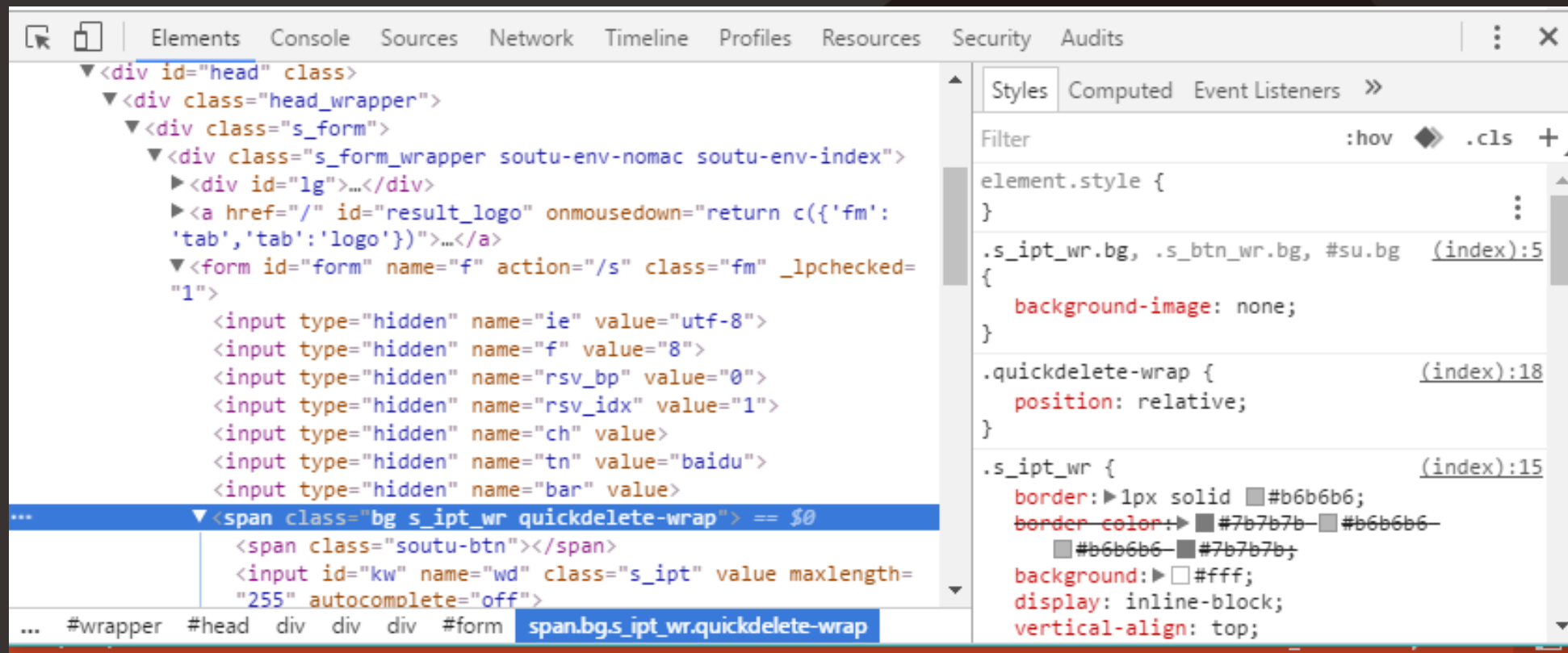
请多多关照! 谢谢!

Web UI 自动化基础入门 Day 3

- **Chrome** 开发者工具介绍
- **Firebug** 插件介绍
- **FirePath** 插件介绍
- **HTML** 介绍
- **XPath** 介绍
- **findElement\findElements** 方法
- 八种定位方式

Chrome 开发者工具介绍

1. 打开Chrome 浏览器
2. 按下F12 打开开发者工具

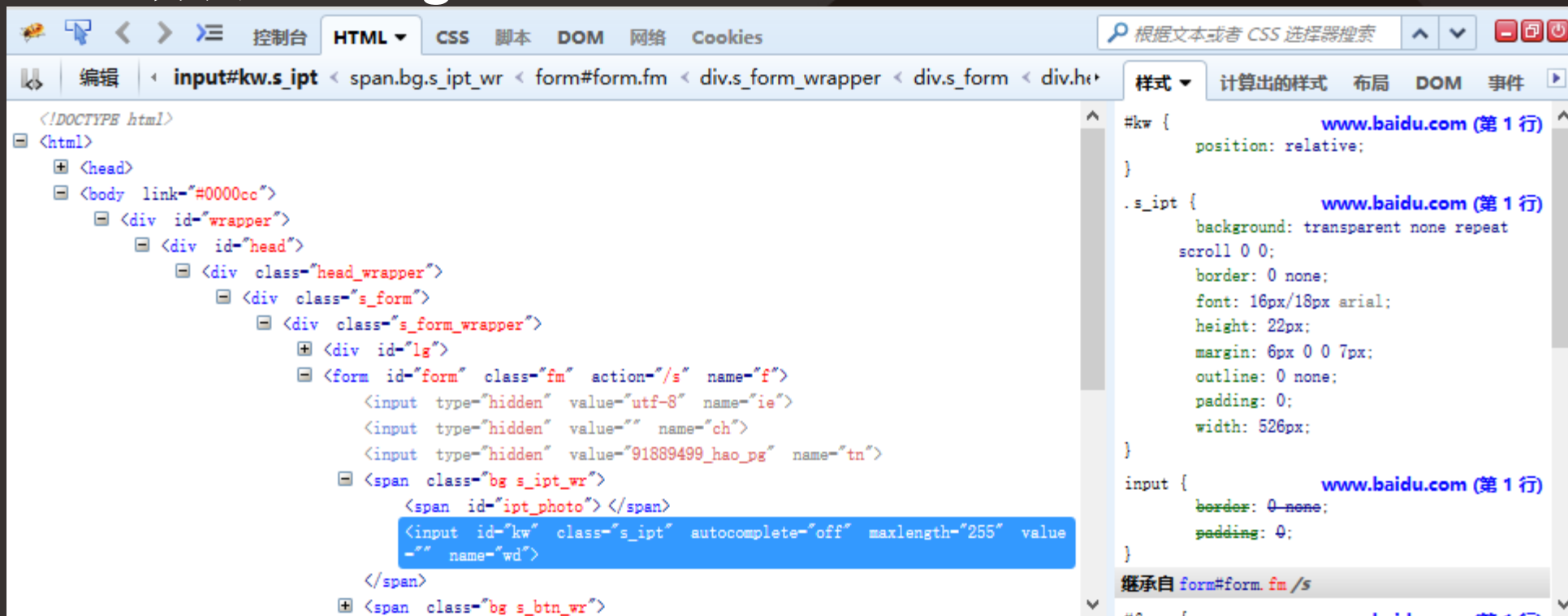


Firebug 插件介绍

1. 安装插件:

打开火狐=>附加组件=>获取附加组件=>搜索“firebug”=>安装并重启

2. F12 打开Firebug

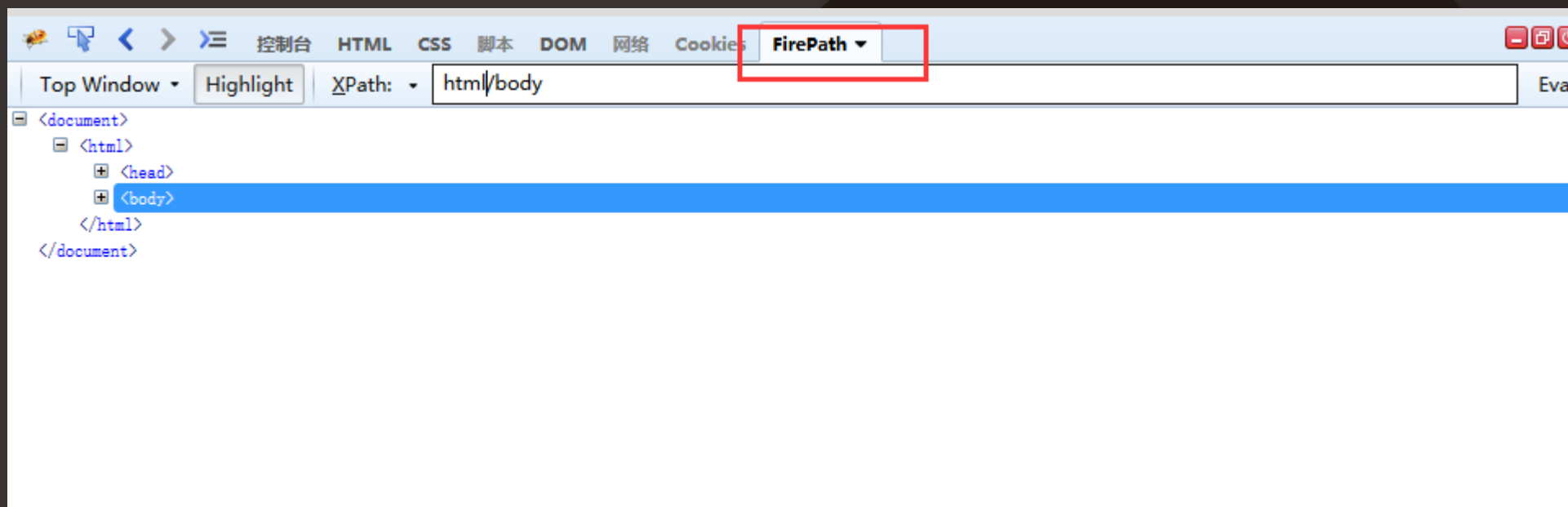


FirePath 插件介绍

1. 安装插件:

打开火狐=>附加组件=>获取附加组件=>搜索“firepath”=>安装并重启

2. F12 打开Firebug, 可以看到多了firepath选项



HTML介绍

1. 什么是HTML

HTML(Hyper Text Markup Language):指的是超文本标记语言，他不是一种编程语言，而是一种标记语言，HTML包括一套标记标签，使用标签来描述网页。

2. HTML 基本结构

```
<html>
```

```
  <head>
```

```
    .....
```

```
  </head>
```

```
  <body>
```

```
    .....
```

```
  </body>
```

```
</html>
```



HTML介绍

<html></html>:

为文档的根元素，所有的描述都在这内部

<head></head>:

为文档头信息，头信息的元素不会浏览器中显示

可以包含<title></title>、<script></script>、<style></style>标签

<body></body>:

为文档正文，其信息会在浏览器中显示。

可以包含文本标签，链接，图像标签，表格标签，框架标签等等

XPath 介绍

1. 什么是XML

XML (eXtensible Markup Language) 指可扩展标记语言，与**HTML**类似，但是他为了传输和存蓄数据而非显示数据。

2. 什么是XPath

XPath 是在**XML**文档中查找信息的一种语言，**XPath** 用来在 **XML** 文档中对元素和属性进行遍历。

虽然**XPath** 是用来查找**XML**节点，但同样可以用来查找**HTML**文档中的节点，因为**HTML**和**XML**结构类似。

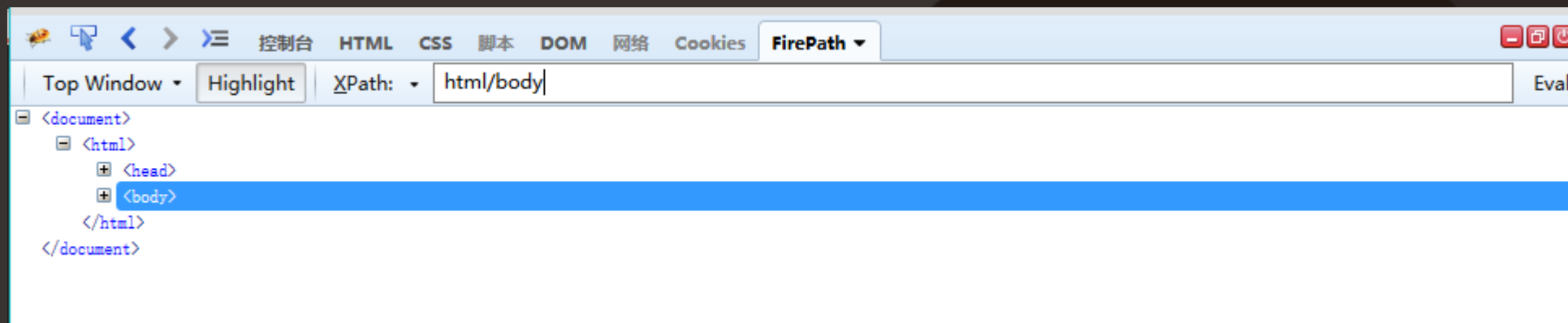
XPath 介绍

1. Xpath语法

表达式	描述
/	从根节点选取。
//	从匹配选择的当前节点选择文档中的节点，而不考虑它们的位置。
.	选取当前节点。
..	选取当前节点的父节点。
@	选取属性。
[]	属性的判断表达式

XPATH介绍

1. 打开示例：index.html  index.html
2. 在FirePath中输入：/html/body



3. 如上图所示，已经选择根目录html下的body

XPATH介绍

在index.html上尝试如下XPATH

- `/html/div`，没有节点可以被选择，因为/如果用在中间，表示绝对路径，是上一个节点的子结点，而html的子节点是head与body
- `/html//div`，表示选择根目录下的所有的子孙后代节点中的div节点，//表示相对路径
- `//div`，表示选择所有的div节点，可以想想/html//div与//div为什么结果是一样的！
- `//div/div`，表示选择所有的div节点的子节点中含有div的节点
- `//div/div/.`，表示选择//div/div节点的当前层的节点，与//div/div的结果相同
- `//div/div/..`，表示选择//div/div节点的上一层节点，也就是选择一个div节点，该div节点的子节点有div节点。有点绕口，但细细理解，会恍然大悟的
- `//div/div/*`，表示选择//div/div的所有子节点，//div/div会有两个匹配出来的节点，但为什么//div/div/*只有一个了呢？这是因为第二个//div/div下面没有子节点了，所以只匹配出来了一个
- `//div[@id='input']`，表示选择一个id为'input'的div节点
- `//div[@id='input']/input`，表示选择一个id为'input'的div节点的input子节点



XPATH介绍

在index.html上尝试如下XPATH

- `//table//input[@id='user']`，表示选择table的子孙后代中id为user的input节点
- `//input[@name='identity' and @class='Volvo']`，有的节点，只用一个属性无法定位出来，必须要用到多个属性进行组合定位，用连接符and。这个XPATH表示选择一个name为identity并且class为Volvo的input节点
- `//input[@name='identity' or @class='Volvo']`，这个多属性组合用的是or的连接符，这个XPATH表示选择一个name为identity，或者class为Volvo的节点，所以，这个XPATH匹配出来了4个节点
- `//input[@name='identity' or @class='Volvo'][1]` 见图13，我们刚知道了，`//input[@name='identity' or @class='Volvo']`匹配出4个，我们只需要第一个，怎么办？加index即可：`//input[@name='identity' or @class='Volvo'][1]`，请注意，xpath的index是以1开头的，并不是0，请切记！
- 取最后一个，`//input[@name='identity' or @class='Volvo'][last()]`



在index.html尝试如下XPATH

- 需要特别注意的一个地方:

`//table//tr//input`，这个匹配出来的，有14个节点，但是如果我们需要取到第一个，怎么办？

有可能会用到：`//table//tr//input[1]`，但是我们来看看结果，匹配出来的节点居然是8个，而不是1个，这是因为`//table//tr//input[1]`是指先匹配出`//table`下面的所有的tr子孙后代节点，并且再此基础上，再匹配出tr节点的所有的子孙后代中的input结点的第一个，由于tr众多，所以匹配出的结果肯定不是一个，但如何能匹配出1个？也就是说我们需要把众多的tr给固定出一个，这时候再看：

`//table//tr[1]//input[1]`，这时候就只有一个匹配出来的节点，所以，请大家仔细揣摩这里面的区别，细细体会



XPATH介绍

在index.html上尝试如下XPATH

- XPATH的几个常用函数

1. `contains ()`: `//div[contains(@id,'in')]` ,表示选择id中包含有'in'的div节点
2. `text()`: 由于一个节点的文本值不属于属性, 比如
“`baidu`” ,所以, 用`text()`函数来匹配节点: `//a[text()='baidu']`
3. `last()`: 前面已介绍
4. `starts-with()`: `//div[starts-with(@id,'in')]` , 表示选择以'in'开头的id属性的div节点
5. `not()`函数, 表示否定, `//input[@name='identity' and not(contains(@class,'a'))]` , 表示匹配出name为identity并且class的值中不包含a的input节点。 `not()`函数通常与返回值为true or false的函数组合起来用, 比如`contains()`,`starts-with()`等, 但有一种特殊情况请注意一下: 我们要匹配出input节点含有id属性的, 写法如下: `//input[@id]`, 如果我们要匹配出input节点不含用id属性的, 则为: `//input[not(@id)]`



XPATH介绍

- XPATH的轴
 - 轴的概念:

<code>ancestor</code>	选取当前节点的所有先辈（父、祖父等）。
<code>ancestor-or-self</code>	选取当前节点的所有先辈（父、祖父等）以及当前节点本身。
<code>attribute</code>	选取当前节点的所有属性。
<code>child</code>	选取当前节点的所有子元素。
<code>descendant</code>	选取当前节点的所有后代元素（子、孙等）。
<code>descendant-or-self</code>	选取当前节点的所有后代元素（子、孙等）以及当前节点本身。
<code>following</code>	选取文档中当前节点的结束标签之后的所有节点。
<code>following-sibling</code>	选取当前节点之后的所有同级节点。
<code>namespace</code>	选取当前节点的所有命名空间节点。
<code>parent</code>	选取当前节点的父节点。
<code>preceding</code>	选取文档中当前节点的开始标签之前的所有节点。
<code>preceding-sibling</code>	选取当前节点之前的所有同级节点。
<code>self</code>	选取当前节点。

XPATH介绍

XPATH的轴的使用法:

- `//div[@id='radio']//label[text()='Saab']/preceding-sibling::input[1]`, 选择label的text为Saab的节点之前的同级节点中为input节点的第一个, 有点绕口, 看实例更易明白, 从中可以看出, 我们可以根据一个radiobox的label来匹配出这个radiobox
- 用XPATH轴时应该注意的几个问题:
 - 调用轴时, 最好用'/'
 - 轴后面要加上符号"::"
 - "::<"后面可以接节点名称, 也可以接"*"
- 轴的另外一种写法
 - `//input[following-sibling::label[1][text()='Saab']]`, 这个的作用与 `//div[@id='radio']//label[text()='Saab']/preceding-sibling::input[1]`的作用是一样的!



FirePath 获取元素XPath

1. 打开FirePath => 点击下图1位置 => 选取你要获取xpath的元素

控件名称	控件操作
Input按钮	 2
Link链接	登陆界面



Top Window ▾ Highlight XPath: `//*[@id='user']` 3 Eval

```
<document>
├── <html>
│   ├── <head>
│   └── <body>
│       └── <div>
│           ├── <div class="text" style="font-size:36;text-align:center">
│           │   └── <br/>
│           └── <table cellpadding="10" border="1">
│               ├── <thead>
│               └── <tbody>
│                   └── <tr>
│                       ├── <td>Input按钮</td>
│                       └── <td class="widgetStyle">
│                           └── <div id="input">
│                               └── <input id="user" type="text"/>
│                           </div>
│                       </td>
│                   </tr>
│               </tbody>
│           </table>
│       </div>
│   </body>
└── </html>
</document>
```

1 matching node

元素定位的重要性

- 自动化测试的根本：
准确的操作测试对象（元素）

查找元素



操作元素

findElement 和 findElements 方法

1. findElement()

返回一个WebElement元素

2. findElements()

返回一个List，多个WebElement元素

八种定位方式

- **By.id(id):** 通过ID 属性查找
- **By.name(name):** 通过name属性查找
- **By.className(className) :** 通过classname属性查找
- **By.linkText(链接文本):** 通过链接文本
- **By.partialLinkText(部分链接文本):** 通过部分链接文本
- **By.cssSelector(Css路径):** 通过CSS路径
- **By.tagName(name):** 通过tagname查找
- **By.xpath(XPath路径):** 通过XPath查找



八种定位方式

By.id(属性id值):

1. ID查找最有效，也是最快的，如果一个dom节点有ID，那用它是最好的定位方式，而且ID基本上唯一。

2. HTML 源码:

```
<a onclick="return false;" id="lb" name="tj_login"
href="https://passport.baidu.com/v2/?login&tpl=mn&u=http%3A%2F%2Fwww.baidu
.com%2F">登录</a>
```

代码例子:

```
WebElement element = driver.findElement(By.id("lb"));
```



八种定位方式

By.name(属性name值):

HTML 源码:

```
<a class="reg" name="tj_reg" target="_blank"  
href="https://passport.baidu.com/v2/?reg&regType=1&tpl=mn&u=http%3A%2F%2Fwww.baidu.com%2F">注册</a>
```

代码例子:

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

厦门云层天咨软件技术有限公司



Xiamen cloudits Consulting Software Technology Co. Ltd.

八种定位方式

By.className(属性class值):

HTML 源码:

```
<a class="reg" name="tj_reg" target="_blank"  
href="https://passport.baidu.com/v2/?reg&regType=1&tpl=mn&u=http%3A%2F%2Fwww.baidu.com%2F">注册</a>
```

代码例子:

```
WebElement element = driver.findElement(By.className( "reg"));
```

厦门云层天咨软件技术有限公司



Xiamen cloudits Consulting Software Technology Co. Ltd.

八种定位方式

By. linkText (Link文本):

HTML 源码:

```
<a name="tj_setting" href="http://www.baidu.com/gaoji/preferences.html">搜索设置</a>
```

代码例子:

```
WebElement element = driver.findElement(By.linkText( "搜索设置" ));
```

八种定位方式

By. partialLinkText(Link文本):

HTML 源码:

```
<a name="tj_setting" href="http://www.baidu.com/gaoji/preferences.html">搜索设置</a>
```

代码例子:

```
WebElement element = driver.findElement(By. partialLinkText( "搜索" ));
```

八种定位方式

By. tagName(dom节点名):

HTML 源码:

```
<a name="tj_setting" href="http://www.baidu.com/gaoji/preferences.html">搜索设置</a>
```

代码例子:

```
WebElement element= driver.findElement(By. tagName( "a" ));
```

八种定位方式

By. xpath(xpath路径):

通过firepath 获取到页面元素的xpath路径:



代码例子:

```
WebElement element= driver.findElement(By. xpath( "//*[@id='kw']" ));
```

八种定位方式

By. cssSelector(Css路径):

跟xpath类似，通过firepath 获取到页面元素的css路径：



代码例子:

```
WebElement element= driver.findElement(By. cssSelector("#kw"));
```

补充:

1. 使用**findElement()**方法查找元素，元素必须是唯一
2. **findElements()**同样支持这八种定位方式，只是获取的是多个元素，返回List

```
List<WebElement> elements = driver.findElements(By.id("test"));
```
3. 如果定位方式错误，元素找不到将抛出**NoSuchElementException**错误，如下图:

```
org.openqa.selenium.NoSuchElementException: no such element: Unable to locate element: {"method":"id","selector":"test"}
```

为企业培养定制化
人员！



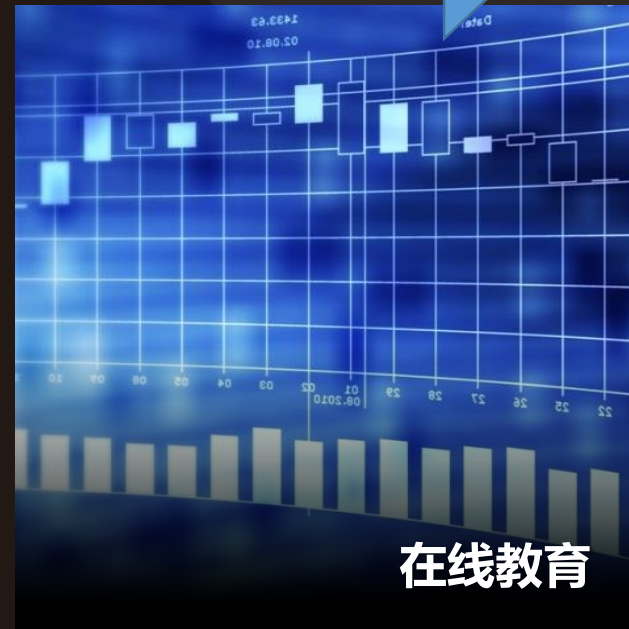
就业培训

为学校构建适应企
业的课程体系！



高校实训

千里之外也能获得
专业的培训指导！



在线教育

联系方式:

地 址: 厦门市思明区前埔西路146号 (云层天咨培训中心)

联 系 人: 陈霁、方文庆

联系电话: 0592-8265163/18602195793

邮 箱: wqfang@cloudits.info

请多多关照! 谢谢!

上次课问题总结

- **Selenium** 对应包无法下载
- 某个浏览器无法启动
- 乱码
- **Assert**
- **Case**编写问题

Web UI 自动化基础入门 Day 4

- 点击: `click()`
- 文本框输入文本: `sendKeys()`
- 清空文本框: `clear()`
- 获取文本: `getText()`
- 获取title: `getTitle()`
- 获取TagName: `getTagName()`
- 获取属性值: `getAttribute()`
- 判断元素是否展示: `isDisplayed()`
- 判断选择框是否选取: `isSelected()`
- 判断输入框是否激活: `isEnabled()`
- 截图

元素定位的重要性

- 自动化测试的根本：
准确的操作测试对象（元素）

查找元素



操作元素

点击：click()

测试用例：

1. 打开百度首页
2. 点击百度首页的“糯米”链接。

代码例子：

```
@Test
public void clickTest() {
//    打开百度首页，http 不能遗漏
    driver.get("http://www.baidu.com");
//    定位 糯米 链接文本
    WebElement element = driver.findElement(By.xpath(".*[@id='u1']/a[1]"));
//    点击该该文本
    element.click();
}
```

文本框输入文本: sendkeys()

测试用例:

1. 打开百度页面
2. 在搜索文本框输入selenium
3. 点击百度一下按钮

代码例子:

```
@Test
public void sendKeysTest(){
//    打开百度首页, http 不能遗漏
    driver.get("http://www.baidu.com");
//    搜索框输入 selenium
    driver.findElement(By.id("kw")).sendKeys("selenium");
//    点击百度一下按钮
    driver.findElement(By.id("su")).click();
}
```



清空文本框：clear()

测试用例：

1. 打开百度首页
2. 在搜索文本框输入selenium
3. 清空搜索输入框

代码例子：

```
@Test
public void clearTest() throws InterruptedException {
    // 打开百度首页，http 不能遗漏
    driver.get("http://www.baidu.com");
    // 定位 搜索输入框
    WebElement element = driver.findElement(By.id("kw"));
    // 输入框输入 selenium
    element.sendKeys("selenium");
    // 为了看到效果，清空前等待3S钟
    Thread.sleep(3000);
    // 清空输入框文本
    element.clear();
}
```



获取文本: getText()

测试用例:

1. 打开百度首页
2. 获取右上角所有的文本并输出

代码例子:

```
@Test
public void getTextTest() {
    // 打开百度首页, http 不能遗漏
    driver.get("http://www.baidu.com");
    // 定位百度首页右上角的文本
    List<WebElement> listText = driver.findElements(By.xpath(".*//*[id='u1']/a"));
    for (int i = 0; i < listText.size(); i++) {
        // 获取文本
        String text = listText.get(i).getText();
        System.out.println(text);
    }
}
```

获取tagName: getTagName()

测试用例:

1. 打开百度首页
2. 获取搜索框的TagName

代码例子:

```
@Test
public void getTagNameTest() throws InterruptedException {
    // 打开百度首页
    driver.get("http://www.baidu.com");
    // 定位 搜索框
    WebElement element = driver.findElement(By.id("kw"));
    // 获取tagName 并输出
    String tag = element.getTagName();
    System.out.println(tag);
}
```


获取属性值: `getAttribute()`

测试用例:

1. 打开百度首页
2. 搜索框输入“webdriver”
3. 获取搜索框的 `value` 属性值

代码例子:

```
@Test
public void getAttributeTest() throws InterruptedException {
    // 打开百度首页
    driver.get("http://www.baidu.com");
    // 定位 搜索框
    WebElement element = driver.findElement(By.id("kw"));
    element.sendKeys(...keysToSend: "webdriver");
    // 获取tagName 并输出
    String att = element.getAttribute(name: "value");
    System.out.println(att);
}
```



获取title: getTitle()

测试用例:

1. 打开百度首页
2. 获取当前页面的title
3. 校验title值等于“百度一下，你就知道”

代码例子:

```
@Test
public void getTitleTest() {
    // 打开百度首页, http 不能遗漏
    driver.get("http://www.baidu.com");
    // 获取百度首页的title
    String title = driver.getTitle();
    // 校验title值等于“百度一下，你就知道”
    Assert.assertEquals(title, "百度一下，你就知道");
}
```

判断元素是否展示: isDisplayed()

测试用例:

1. 打开百度首页
2. 校验百度一下按钮已经展示

代码例子:

```
@Test
public void isDisplayedTest() {
    // 打开百度首页, http 不能遗漏
    driver.get("http://www.baidu.com");
    // 获取 百度一下 按钮是否显示
    Boolean bl = driver.findElement(By.id("su")).isDisplayed();
    // 校验结果
    Assert.assertTrue(bl, "校验百度一下按钮是否显示");
}
```

判断选择框是否选取: isSelected()

测试用例:

1. 打开 “UI自动化测试” 主页
2. 校验 “Volvo” 单选框已经选中

代码例子:

```
@Test
public void isSelectedTest() {
    // 打开UI自动化测试主页, http 不能遗漏
    driver.get("http://192.168.1.109:8081/");
    // 定位并获取“Volvo”单选框是否选中状态
    Boolean bl = driver.findElement(By.xpath(".*[@id='radio']/input[1]")).isSelected();
    // 校验结果
    Assert.assertTrue(bl, "校验单选框已选中");
}
```

判断元素是否激活: isEnabled()

测试用例:

1. 打开“UI自动化测试”主页
2. 校验 submit 文本框为不可输入状态

代码例子:

```
@Test
public void isEnabledTest() {
//    打开UI自动化测试主页, http 不能遗漏
    driver.get("http://192.168.1.109:8081/");
//    定位并获取文本框是否可输入状态
    Boolean bl = driver.findElement(By.name("buttonhtml")).isEnabled();
//    校验结果
    Assert.assertFalse(bl, "校验输入框为不可输入状态");
}
```

截图

测试用例:

1. 打开百度主页
2. 截图

代码例子:

```
@Test
public void screenShotFile() {
    driver.get("http://www.baidu.com");
    // 下面代码是得到截图并保存在D盘下
    File screenShotFile = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);
    try {
        FileUtils.copyFile(screenShotFile, new File("D:/test.png"));
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

为企业培养定制化
人员！



就业培训

为学校构建适应企
业的课程体系！



高校实训

千里之外也能获得
专业的培训指导！



在线教育

联系方式:

地 址: 厦门市思明区前埔西路146号 (云层天咨培训中心)

联 系 人: 陈霁、方文庆

联系电话: 0592-8265163/18602195793

邮 箱: wqfang@cloudits.info

请多多关照! 谢谢!

Web UI 自动化基础入门 Day 5

- **Alert、Confirm、prompt** 的处理
- **iFrame** 的处理
- 下拉框的选取
- 多窗口的处理
- 元素等待的处理

Alert的处理

测试用例:

1. 打开“UI自动化测试”主页
2. 点击Alert按钮
3. 在alert警告框点击确定按钮

代码例子:

```
@Test
public void alertTest() throws InterruptedException {
    // 打开ui测试主页
    driver.get("http://192.168.1.108:8081/");
    // 点击Alert按钮
    driver.findElement(By.xpath(".*[@id='alert']/input")).click();
    // 为了看效果等待3S
    Thread.sleep(3000);
    // 选取 alert弹窗
    Alert alert = driver.switchTo().alert();
    // 点击警告框的确定按钮
    alert.accept();
}
```



Confirm的处理

测试用例:

1. 打开“UI自动化测试”主页
2. 点击Confirm按钮
3. 在Confirm警告框点击确定\取消按钮

代码例子:

```
@Test
public void confirmTest() throws InterruptedException {
//    打开ui测试主页
    driver.get("http://192.168.1.108:8081/");
//    点击confirm按钮
    driver.findElement(By.xpath("//*[id='confirm']/input")).click();
//    为了看效果等待3S
    Thread.sleep(3000);
//    选取 alert弹窗
    Alert alert = driver.switchTo().alert();
//    处理confirm的确定按钮
    alert.accept();
//💡 处理confirm的取消按钮
//    alert.dismiss();
}
```



Prompt的处理

测试用例:

1. 打开“UI自动化测试”主页
2. 点击Prompt按钮
3. 在Prompt 弹窗中，输入“这个是Prompt”
4. 点击确定\取消按钮

代码例子:

```
@Test
public void promptTest() throws InterruptedException {
    // 打开ui测试主页
    driver.get("http://192.168.1.108:8081/");
    // 点击confirm按钮
    driver.findElement(By.xpath("//*[id='prompt']/input")).click();
    // 选取警告弹窗
    Alert prompt = driver.switchTo().alert();
    // 输出弹窗内容
    prompt.sendKeys("输入值。。。。");
    // 为了看效果等待2s
    Thread.sleep(2000);
    // 处理确定按钮
    prompt.accept();
    // 处理取消按钮
    prompt.dismiss();
}
```



iFrame的处理

思路:

1. 定位iFrame
2. driver控制权交给iFrame
3. 操作iFrame里面的元素
4. driver控制权交回原页面

代码例子:

```
@Test
public void iframeTest(){
    // 打开ui测试主页
    driver.get("http://192.168.1.108:8081/");
    // 用id或者name定位iframe, 并控制权交给iframe,
    driver.switchTo().frame("aa");
    // 操作iframe里面的元素
    driver.findElement(By.id("user")).sendKeys("我在iframe里面的元素输入了。。");
    // 控制权交回顶部的frame
    driver.switchTo().defaultContent();
}
```

下拉框的处理

三种处理方式:

1. `selectByIndex()` 根据索引来选取, 从0开始
2. `selectByValue()` 根据属性value的属性值来选取
3. `selectByVisibleText()` 根据标签之间的Text值, 也就是页面显示的

代码例子:

```
@Test
public void selectTest() throws InterruptedException {
    // 打开ui测试主页
    driver.get("http://192.168.1.108:8081/");
    // 定位下拉框
    WebElement element = driver.findElement(By.name("select"));
    // new 一个Select对象
    Select select = new Select(element);
    // 通过索引选取
    select.selectByIndex(2);
    // 为了看到页面选取效果等待5S
    Thread.sleep(5000);
    // 通过属性value值选取
    select.selectByValue("volvo");
    // 为了看到页面选取效果等待5S
    Thread.sleep(5000);
    // 通过text选取
    select.selectByVisibleText("Audi");
}
```



多窗口的处理

代码例子:

```
@Test
public void winTest() throws InterruptedException {
    // 打开ui测试主页
    driver.get("http://192.168.1.108:8081/");
    // 定位元素并点击
    driver.findElement(By.xpath(".*[@id='open']/a")).click();
    // 获取当前页面句柄
    String handle = driver.getWindowHandle();
    // 获取所有页面的句柄，并循环判断不是当前的句柄，就做选取switchTo()
    for (String handles : driver.getWindowHandles()) {
        if (handles.equals(handle))
            continue;
        driver.switchTo().window(handles);
    }
    // 为了看到效果暂停3S
    Thread.sleep(3000);
    // 再新打开的也输入
    driver.findElement(By.id("user")).sendKeys("test1");
    // 为了看到效果暂停3S
    Thread.sleep(3000);
    // 关闭当前页面
    driver.close();
    // 选取原来的句柄，回到原来页面
    driver.switchTo().window(handle);
    Thread.sleep(3000);
    // 在原来的页面输入
    driver.findElement(By.id("user")).sendKeys("test2");
}
```



元素等待的处理

等待方式1:

线程等待: `Thread.sleep(xxxx)`

等待方式2:

全局等待: `driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS)`

等待方式3:

显示等待:

`new WebDriverWait(driver, 30).until(ExpectedConditions.presenceOfElementLocated(By))`

为企业培养定制化
人员！



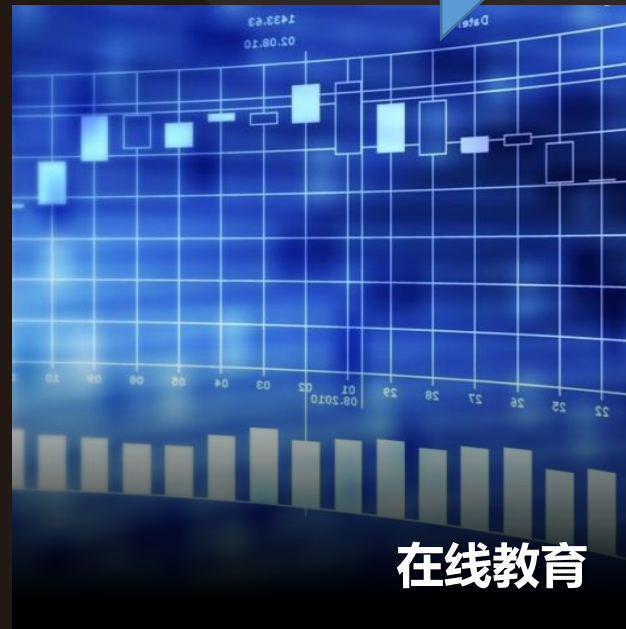
就业培训

为学校构建适应企
业的课程体系！



高校实训

千里之外也能获得
专业的培训指导！



在线教育

联系方式:

地 址: 厦门市思明区前埔西路146号 (云层天咨培训中心)

联 系 人: 陈霁、方文庆

联系电话: 0592-8265163/18602195793

邮 箱: wqfang@cloudits.info

请多多关照! 谢谢!

Web UI 自动化基础入门 Day 6

- **Actions**类
- **Java**的**Robot**类
- 上传文件
- 下载文件

Actions类

背景:

平时我们在做自动化过程中可能需要模拟鼠标、键盘等的一些行为，例如鼠标单击，双击，右键，而且很多web应用可能存在快捷组合键等等。那么可以用WebDriver中提供了Actions类来处理这类需求，更复杂的键盘鼠标处理我们可以通过Java提供的Robot类解决。

Actions类

在元素上鼠标右击和双击

```
@Test
public void testContextClick() {
    // 打开百度页面
    driver.get("http://www.baidu.com");
    // 定位百度一下按钮
    WebElement element = driver.findElement(By.id("su"));
    // new Actions 对象
    Actions builder = new Actions(driver);
    // 在百度一下按钮上右键
    builder.contextClick(element).perform();
    // 双击百度一下按钮
    builder.doubleClick(element).perform();
}
```

Actions类

鼠标移动到某个元素上

```
@Test
public void testMoveToElement() {
    // 打开百度页面
    driver.get("http://www.baidu.com");
    // 定位更多产品
    WebElement element = driver.findElement(By.name("tj_briicon"));
    // 实例化Actions
    Actions builder = new Actions(driver);
    // 鼠标移动到更多产品上
    builder.moveToElement(element).perform();
}
```

Actions类

把元素拖动到 (x,y)

```
@Test
public void testDragAndDropBy() {
//    打开测试页面
    driver.get("file:///C:/selenium_html/dragAndDrop.html");
//    定位元素
    WebElement drag = driver.findElement(By.xpath("/html/body/div[1]"));
//    创建Action 实例
    Actions builder = new Actions(driver);
//    执行拖动到x500 y500的位置
    builder.dragAndDropBy(drag, 500, 500).perform();
}
```

Actions类

把元素拖动到另一个元素上

```
@Test
public void testClickAndHold() {
    driver.get("file:///C:/selenium_html/dragAndDrop.html");
    WebElement oneElement = driver.findElement(By.id("drag"));
    WebElement otherElement = driver.findElement(By.xpath("/html/body/h1"));
    Actions builder = new Actions(driver);
    builder.clickAndHold(oneElement)
            .moveToElement(otherElement)
            .release(otherElement)
            .perform();
}
```


Actions类

下拉框多选

```
@Test
public void test05() {
    driver.get("file:///C:/selenium_html/index.html");
    Actions builder = new Actions(driver);
    WebElement select = driver.findElement(By.id("selectWithMultipleEqualsMultiple"));
    List<WebElement> options = select.findElements(
        By.tagName("option"));
    Action multipleSelect = builder.keyDown(Keys.SHIFT)
        .click(options.get(0))
        .click(options.get(2))
        .build();
    multipleSelect.perform();
}
```

Robot类

1. 按住某个按键

keyPress()

2. 松开某个按键

keyRelease()

```
@Test
    public void robotDemo() throws AWTException {
        Robot robot = new Robot();
        // 按住Ctrl按键
        robot.keyPress(KeyEvent.VK_CONTROL);
        // 按住S按键
        robot.keyPress(KeyEvent.VK_S);
        // 得到S的ASCII值
        int keyS = (int)new Character('S');
        // 松开s键
        robot.keyRelease(keyS);
        // 松开Ctrl键
        robot.keyRelease(KeyEvent.VK_CONTROL);
    }
```



上传文件的处理

思路:

1. 定位上传控件
2. 使用sendkeys()方法, 并传入文件路径

代码例子:

```
@Test
public void uploadTest() throws InterruptedException {
    // 打开UI测试主页
    driver.get("http://192.168.1.108:8081/");
    // 定位页面元素, 并传入要上传的文件路径和名
    driver.findElement(By.id("load")).sendKeys("C:\\Downloads\\export.html");
}
```

作业

1. 完成今天所有的**DEMO**例子
2. 找个下载页面，并完成下载功能的自动化实现

为企业培养定制化
人员！



就业培训

为学校构建适应企
业的课程体系！



高校实训

千里之外也能获得
专业的培训指导！



在线教育

联系方式:

地 址: 厦门市思明区前埔西路146号 (云层天咨培训中心)

联 系 人: 陈霁、方文庆

联系电话: 0592-8265163/18602195793

邮 箱: wqfang@cloudits.info

请多多关照! 谢谢!

补充

➤完成下载操作

➤执行JS 脚本

➤使用PhantomJS

火狐下载文件

例子:

更多参数: http://kb.mozillazine.org/Firefox_%3a_FAQs_%3a_About%3aconfig_Entries

```
@Test
public void testFirefoxDown() throws InterruptedException, AWTException {
    System.setProperty("webdriver.firefox.bin", "C:\\\\Users\\vidorh\\AppData\\Local\\Mozilla Firefox\\firefox.exe");
    FirefoxProfile firefoxProfile = new FirefoxProfile();
    // 设置火狐的默认下载文件夹, 0表示桌面 1表示我的下载 2表示自定义文件夹
    firefoxProfile.setPreference("browser.download.folderList", "2");
    // 设置自定义文件夹位置
    firefoxProfile.setPreference("browser.download.dir", "D:\\soft");
    // 打开一个预先配置的火狐
    WebDriver driver = new FirefoxDriver(firefoxProfile);
    driver.get("http://ri.baidu.com/soft/detail/13478.html?ald");
    driver.findElement(By.xpath(".*[@id='softAbs']/a[2]")).click();
    Thread.sleep(3000);
    Robot robot = new Robot();
    robot.keyPress(KeyEvent.VK_TAB);
    Thread.sleep(1000);
    robot.keyPress(KeyEvent.VK_TAB);
    Thread.sleep(1000);
    robot.keyPress(KeyEvent.VK_ENTER);
    Thread.sleep(10000);
    driver.quit();
}
```



FirefoxDownDemo.java



FirefoxDownDemo.java

执行JS

例子:

```
WebDriver driver;  
@Test  
public void testJSE() throws InterruptedException {  
    System.setProperty("webdriver.chrome.driver", ".\\drivers\\chromedriver.exe");  
    driver = new ChromeDriver();  
    driver.get("http://www.baidu.com");  
    // 把driver转成JavascriptExecutor类型  
    JavascriptExecutor js = (JavascriptExecutor) driver;  
    // 执行js  
    js.executeScript("document.getElementById(\"kw\").setAttribute(\"value\", \"执行js\")");  
    Thread.sleep(5000);  
    driver.quit();  
}
```

使用PhantomJS

1. 下载PhantomJS.exe

下载地址: <http://phantomjs.org/download.html>

2. 配置Pom.xml文件

```
<dependency>  
  <groupId>com.codeborne</groupId>  
  <artifactId>phantomjsdriver</artifactId>  
  <version>1.3.0</version>  
</dependency>
```

使用PhantomJS

代码例子:

```
@Test
public void testPH() {
//    设置phantomjs driver路径
    System.setProperty("phantomjs.binary.path", ".\\drivers\\phantomjs.exe");
//    打开PhantomJS 浏览器
    PhantomJSDriver driver = new PhantomJSDriver();
    driver.get("http://www.baidu.com");
    String title = driver.getTitle();
    System.out.println(title);
    driver.quit();
}
```

Day 7 实战

1. 完成163邮箱的注册
2. 完成正确账号的登录
3. 完成错误账号的登录
4. 完成带附件的发送邮件


RegisterTest.java


SendEmail.java


LoginTest.java


LoginDemo.java

Web UI 自动化基础入门 Day 7

➤ **Selenium IDE**简介与入门

➤ **Selenium Grid**简介与入门

厦门云层天咨软件技术有限公司



Xiamen cloudits Consulting Software Technology Co. Ltd.

Selenium IDE 简介

Selenium-IDE (Selenium Integrated Development Environment :

它是一个开发Selenium测试用例的集成环境。同时它也是一个使用简单的Firefox的扩展，通过该工具我们可以快速的生成Case。

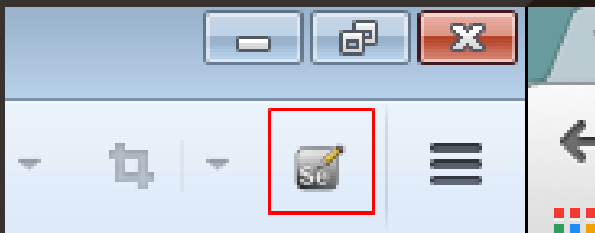
厦门云层天咨软件技术有限公司



Xiamen cloudits Consulting Software Technology Co. Ltd.

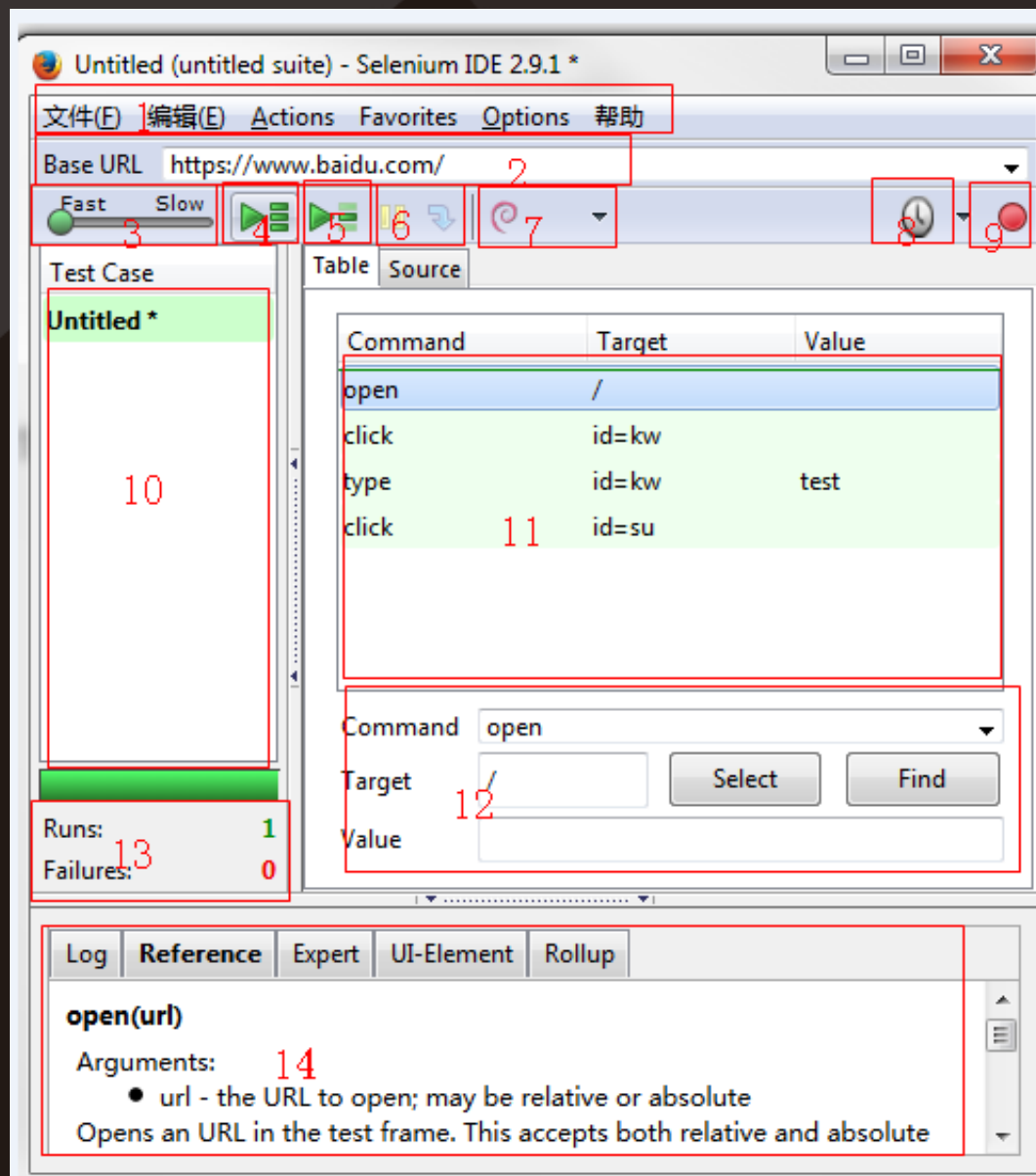
Selenium IDE 安装

1. 打开Firefox
2. 打开: <https://addons.mozilla.org/en-US/firefox/addon/selenium-ide/>
3. 点击页面的【Add to Firefox】按钮
4. 点击【Install】，并重启Firefox
5. Firefox 右上角多了如下标示，便安装完成



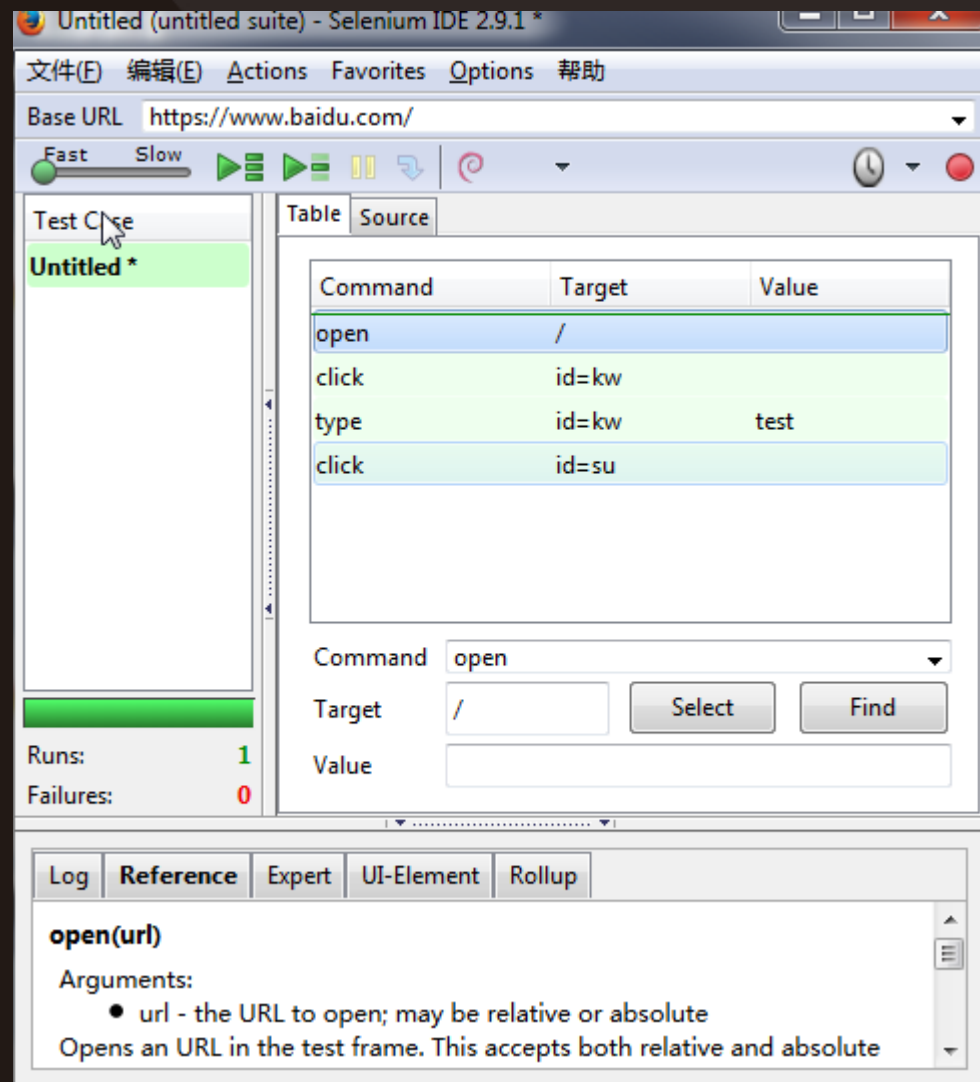
Selenium IDE 面板介绍

1. 菜单栏
2. 被测网站地址
3. 控制运行速度
4. 运行所有测试用例
5. 运行当前测试用例
6. 暂停/恢复
7. 单步运行
8. 设定定时执行脚本
9. 录制/非录制
10. 测试用例集合
11. 测试步骤
12. 当前选择测试步骤对应参数
13. 脚本运行结果
14. 日志等



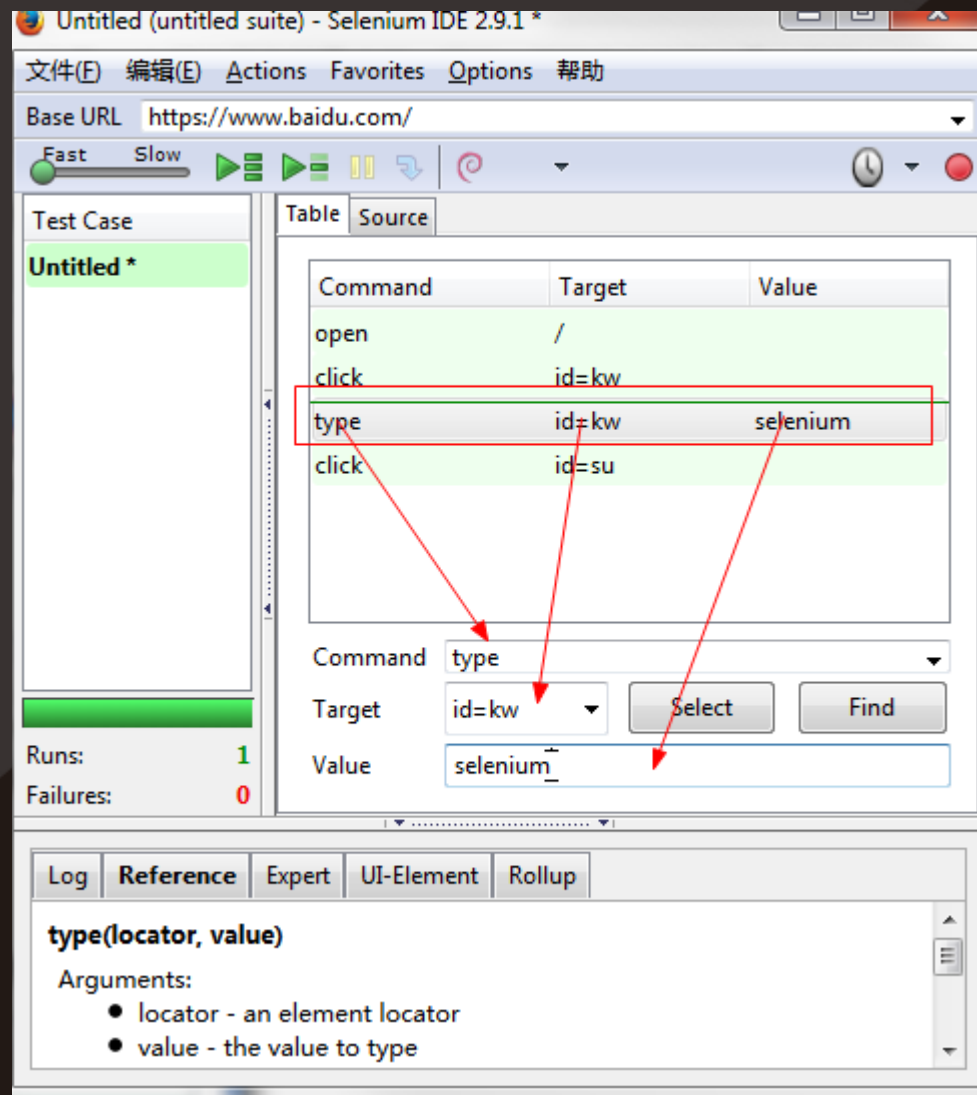
Selenium IDE 录制百度查询

1. 打开IDE
2. 输入<http://www.baidu.com>,并确保录制状态
3. 操作Firefox浏览器，做查询



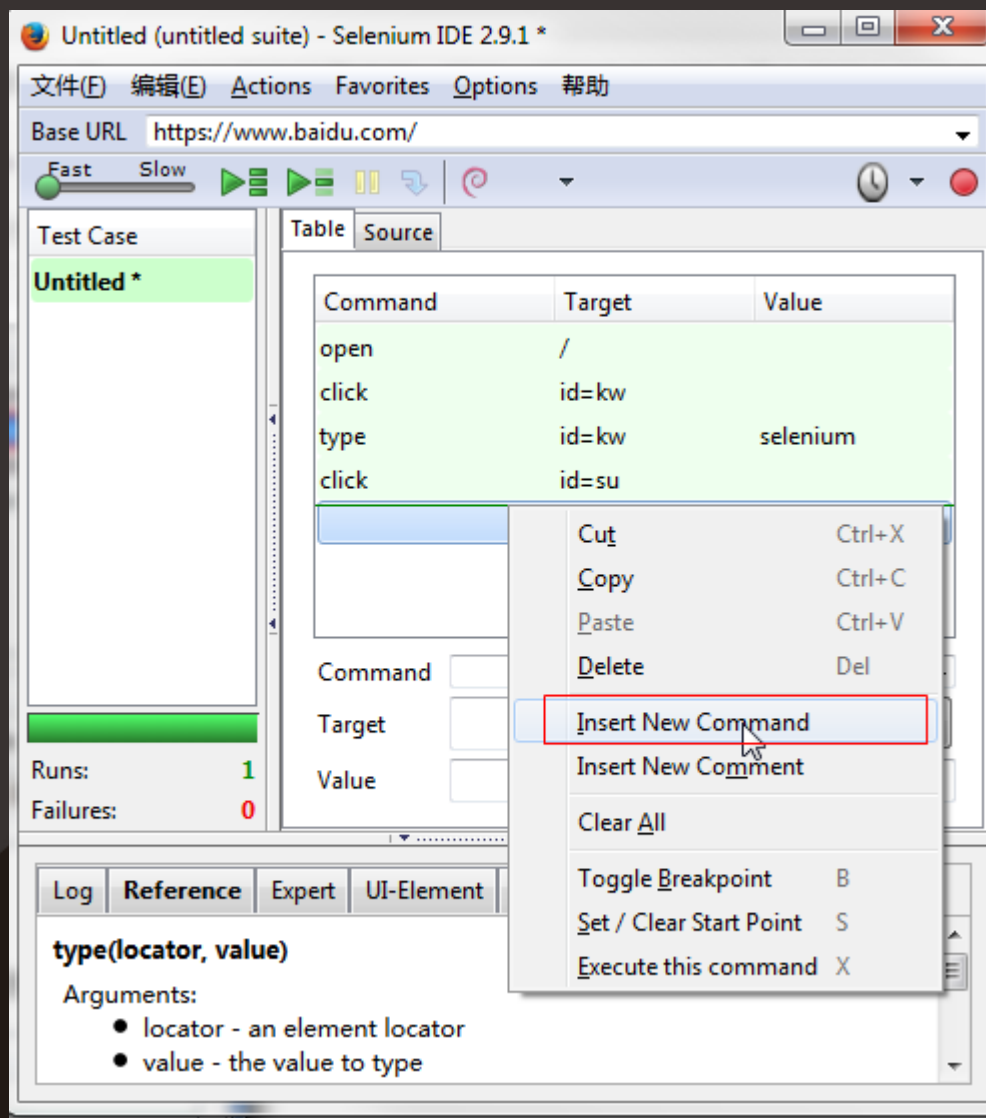
Selenium IDE 常用操作

编辑指令



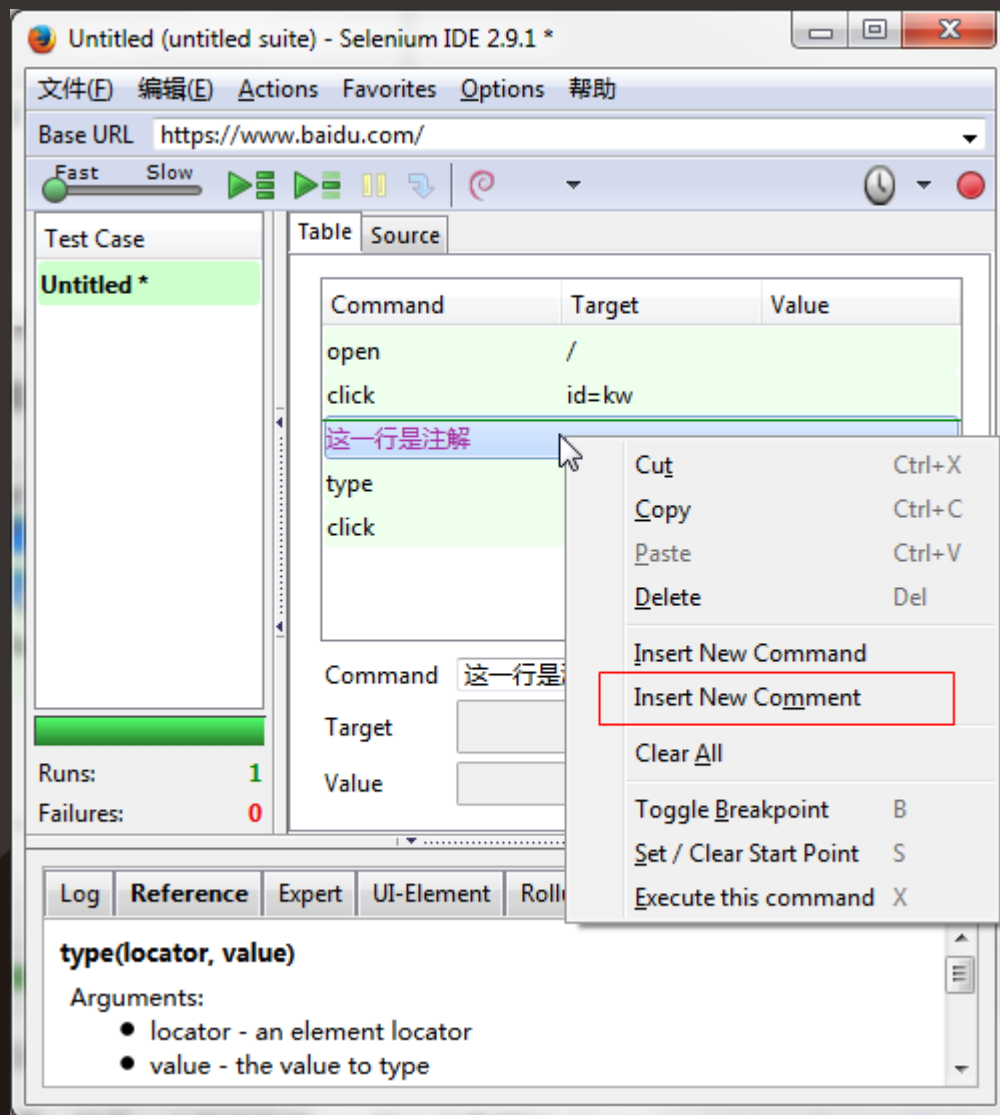
Selenium IDE 常用操作

插入指令



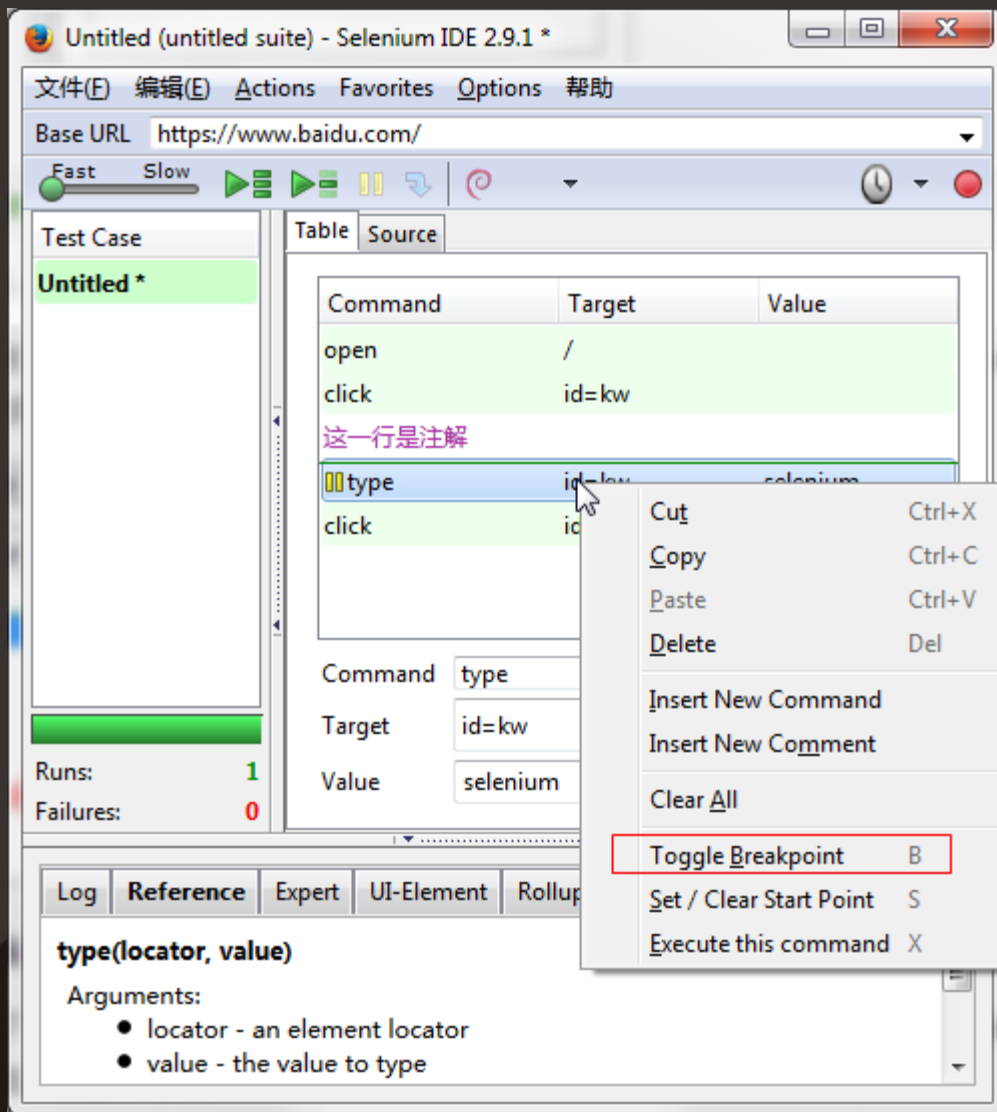
Selenium IDE 常用操作

插入注解



Selenium IDE 常用操作

断点



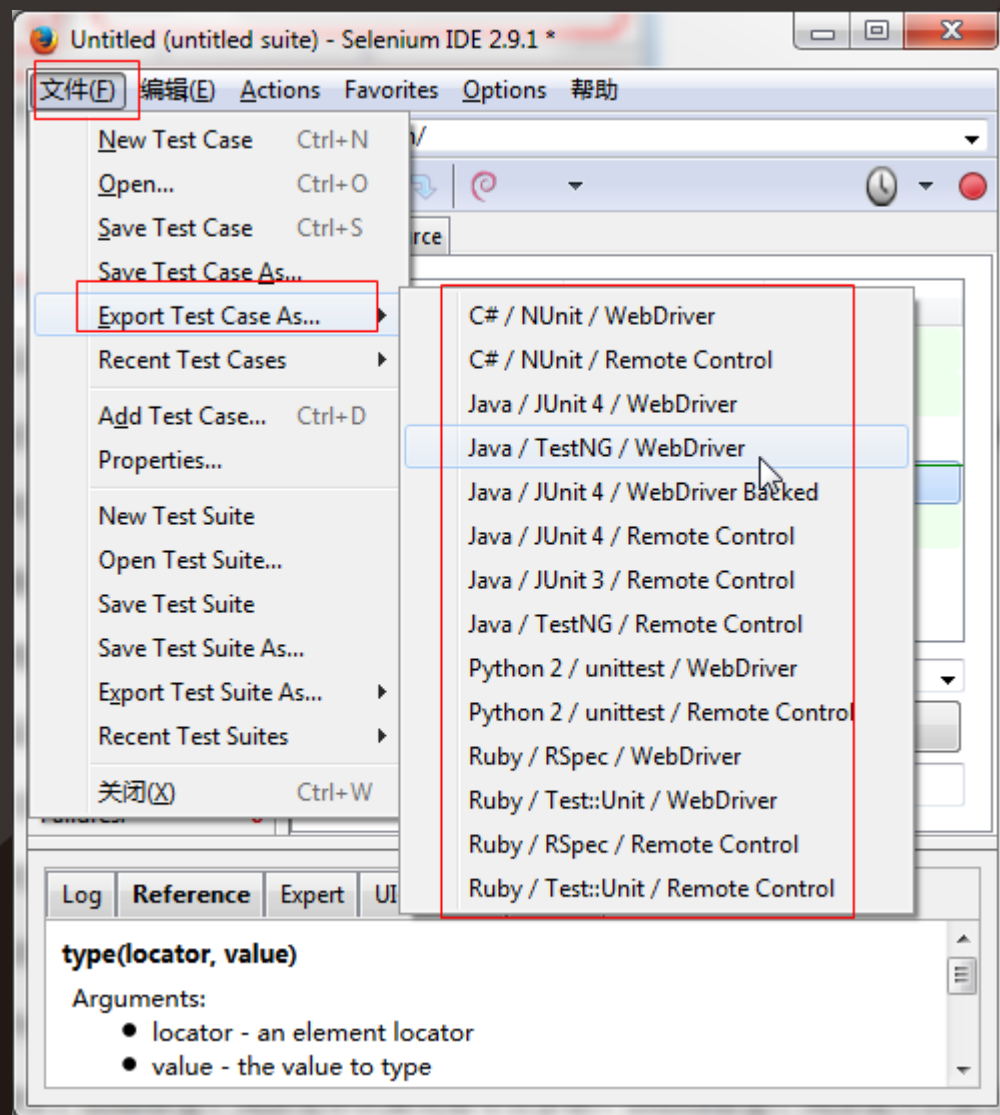
Selenium IDE 常用操作

添加校验等



Selenium IDE 常用操作

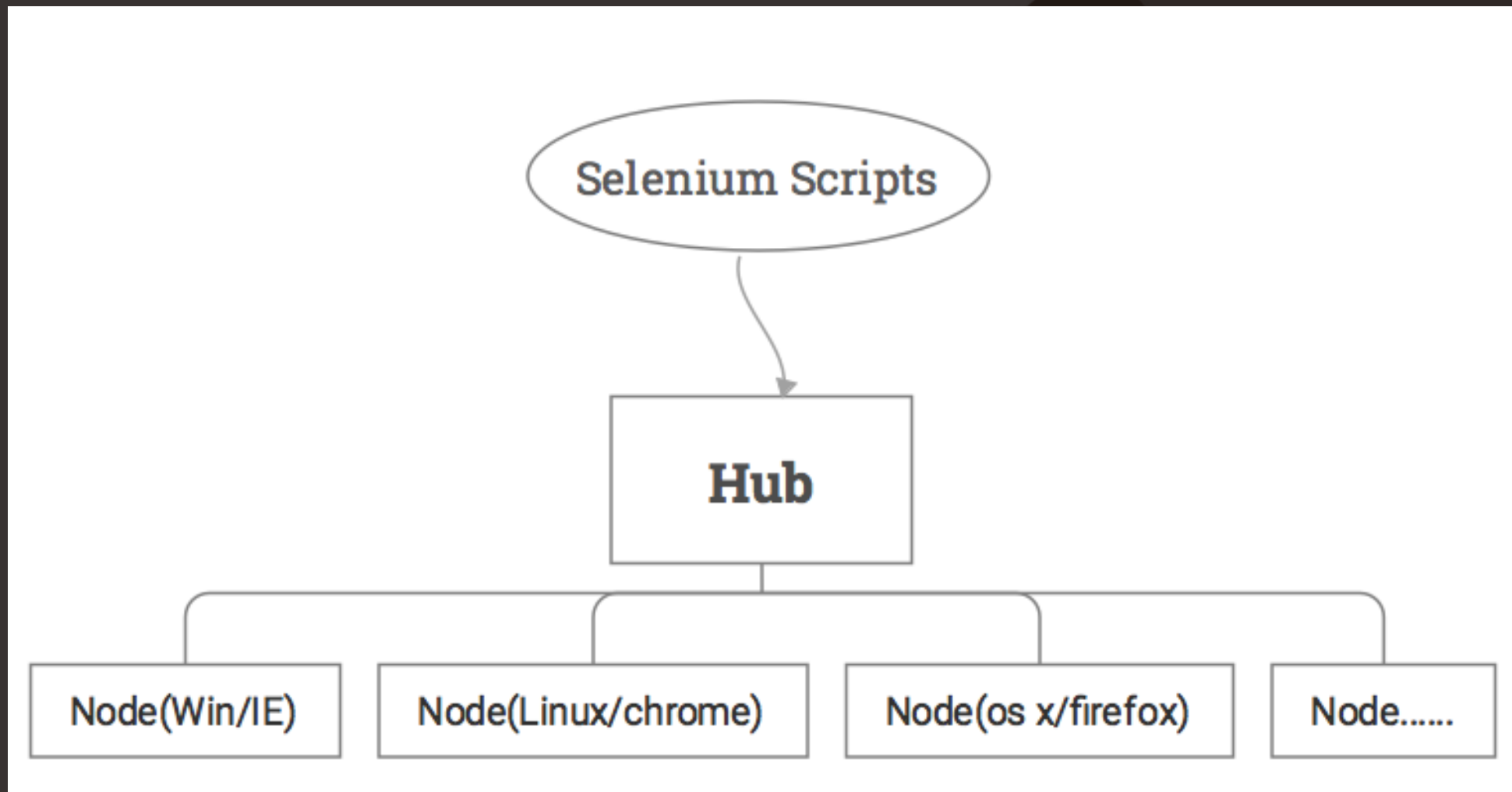
导出脚本



Selenium Grid 简介

- Selenium Grid 是一个可以方便的让你脚本运行在不同的平台以及不同的浏览器上的一个框架。
- Selenium Grid 分1和2两个版本，其中Selenium Grid 2的发布还晚于Selenium 2.0，也就是说Selenium Grid 2 并不是和Selenium 2.0 一起发布的，但是Selenium Grid 2基本上支持Selenium 2.0的所有功能。

Selenium Grid 结构图



何时需要selenium Grid

- 1.需要在不同的系统和浏览器运行测试
- 2.减少测试执行时间

如何使用

- 下载:

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

- 启动Hub:
- `java -jar selenium-server-standalone-2.53.0.jar -role hub`
- 浏览器打开: `http://localhost:4444/grid/console`

Hub常见参数

- **throwOnCapabilityNotPresent : true** 默认为 true，表示当前hub只有在有node存在时，才会接受测试请求。为false则反之；
- **capabilityMatcher : org.openqa.grid.internal.utils.DefaultCapabilityMatcher**这是一个实现了CapabilityMatcher接口的类，默认指向**org.openqa.grid.internal.utils.DefaultCapabilityMatcher**该类用于实现grid在分布测试任务到对应node时所使用的匹配规则，如果想要更精确的测试分配规则，那么就注册一个自己定义的匹配类；
- **prioritizer : null** 这是一个实现了Prioritizer接口的类。设置grid执行test任务的优先逻辑；默认为null，那个脚本先到那个先执行；
- **port : 4444** 这个是hub 默认的端口号；
- **newSessionWaitTimeout : -1** 默认-1，即没有超时；指定一个新的测试session等待执行的间隔时间。即一个代理节点上前后2个测试中间的延时时间，单位为毫秒；
- **browserTimeout : 0** 浏览器无响应的超时时间，默认为0表示没有超时时间

修改Hub配置

1.直接通过命令修改：

```
java -jar selenium-server-standalone-2.53.0.jar -role hub -port 4445
```

2.通过Json文件修改：

```
java -jar selenium-server-standalone-2.53.0.jar -role hub -hubConfig hub.json
```

json格式：

```
{ "host": null, "port": 4444, "newSessionWaitTimeout": -1, "servlets" : [], "prioritizer": null,  
  "capabilityMatcher": "org.openqa.grid.internal.utils.DefaultCapabilityMatcher",  
  "throwOnCapabilityNotPresent": true, "nodePolling": 5000, "cleanUpCycle": 5000, "timeout": 300000,  
  "browserTimeout": 0, "maxSession": 10 }
```

node 节点环境要求

- 1. node 节点必须要有 java 环境
- 2. node 节点跟hub 节点机器间可以互相 ping 通。
- 3. node 节点负责执行Selenium 脚本，所以必须有Selenium 环境（脚本语言对应的环境如 java, 各个浏览器及其对应的driver）

注册node节点

- 启动node并注册到hub:

```
java -jar selenium-server-standalone-2.53.0.jar -role node -hub  
http://192.168.1.110:4444/grid/register
```

修改node节点配置

- 通过命令修改:

```
java -jar selenium-server-standalone-2.53.0.jar -role rc -port 6666
```

- 通过json修改:

```
{ "capabilities": [ { "browserName": "chrome", "maxInstances": 5, "platform":  
  "WINDOWS", "version": "51" } ], "configuration": { "proxy":  
  "org.openqa.grid.selenium.proxy.DefaultRemoteProxy", "maxSession": 5, "port": 5555,  
  "register": true, "registerCycle": 5000, "hub": "http://192.168.84.209:4444" }}
```


Selenium Grid的测试用例

```
@Test
public void testGrid() throws MalformedURLException, InterruptedException {
    DesiredCapabilities chromeDC = DesiredCapabilities.chrome();
    WebDriver driver = new RemoteWebDriver(new URL("http://192.168.153.131:5555/wd/hub"), chromeDC);
    driver.get("http://www.baidu.com");
    Thread.sleep(5000);
    driver.quit();
}
```

需求

- 一套脚本在不同的系统和浏览器执行
- 执行速度尽可能快

代码实现

```
@DataProvider(name = "data1")
public Object[][] dataT() {
    return new Object[][]{
        {"http://192.168.1.107:5555", "chrome"},
        {"http://192.168.153.131:5555", "firefox"},
    };
}

@Test(dataProvider = "data1")
public void testGrid2(String url, String browser) throws MalformedURLException, InterruptedException {
    DesiredCapabilities desiredCapabilities;
    if (browser == "chrome"){
        desiredCapabilities = DesiredCapabilities.chrome();
    }else if(browser == "firefox"){
        desiredCapabilities = DesiredCapabilities.firefox();
    }else {
        desiredCapabilities = DesiredCapabilities.internetExplorer();
    }
    String uri = url + "/wd/hub";
    WebDriver driver = new RemoteWebDriver(new URL(uri),desiredCapabilities);
    driver.get("http://www.baidu.com");
    Thread.sleep(3000);
    driver.quit();
}
```



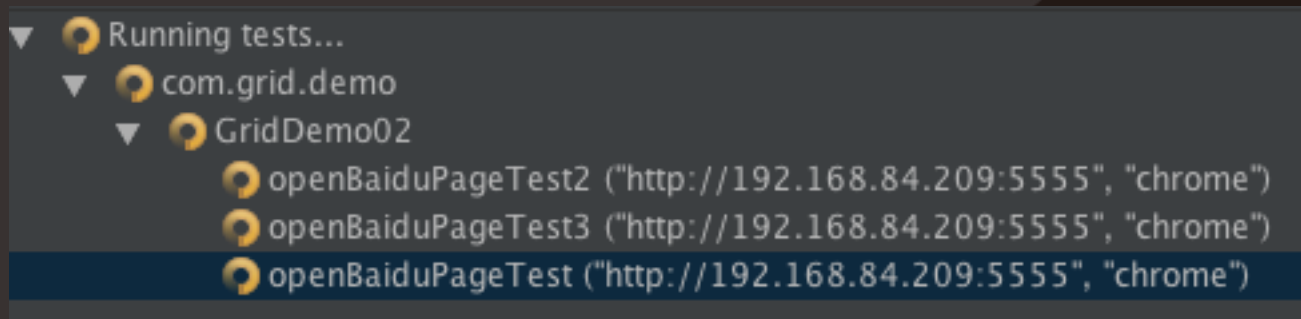
TestNG 并发配置

- 新建个testng.xml 文件，添加如下内容：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Default Suite" parallel="methods" thread-count="3">
  <test name="Selenium_Grid_Demo">
    <classes>
      <class name="com.grid.demo.GridDemo02"/>
    </classes>
  </test>
</suite>
```

运行结果

- 右键运行testng.xml，可以看到3个方法同时执行：



为企业培养定制化
人员！



就业培训

为学校构建适应企
业的课程体系！



高校实训

千里之外也能获得
专业的培训指导！



在线教育

联系方式:

地 址: 厦门市思明区前埔西路146号 (云层天咨培训中心)

联 系 人: 陈霁、方文庆

联系电话: 0592-8265163/18602195793

邮 箱: wqfang@cloudits.info

请多多关照! 谢谢!

Web UI 自动化基础入门 Day 7

➤数据驱动

➤PO模式

Testng 数据驱动

1. @DataProvider注解

```
@DataProvider(name = "data1")
public Object[][] data() {
    return new Object[][]{{"1", "a"}, {"2", "b"}, {"3", "c"}};
}

@Test(dataProvider = "data1")
public void dataProviderTest(String i, String j) {
    System.out.println("i:" + i + "和j:" + j);
}
```

PO（Page-object）理念

◆Page-object思想介绍

◆代码重构及演示



Page-object思想介绍

◆为什么要使用page-object

- ◆集中管理元素对象
- ◆集中管理一个page内的公共方法
- ◆后期维护方便

Page-object思想介绍

◆ 集中管理元素对象

◆ 实现方法:

```
public static By keyTextField = By.id("kw");  
  
public static By baiduButtonb = By.id("su");
```

◆ 调用方法:

◆ `WebElement element = driver.findElement(TestPage.keyTextField);`

Page-object思想介绍

◆ Page类的实现

◆ 说明:

◆ 在具体的脚本中的使用方法:

```
@Test
public void queryTest() {
    driver = SeleniumDriver.initialDriver("chrome");
    driver.get("http://www.baidu.com");
    HomeAction.query("selenium");
    QueryResultAction.assertFirstResult("Selenium");
    SeleniumDriver.stopDriver();
}
```

Page-object思想介绍

PO引入前

- 存在大量冗余代码
- 业务流程不清晰
- 后期维护成本大

PO 引入后

- 减少代码冗余
- 业务和实现分开
- 降低维护成本

<http://blog.csdn.net/mey>



为企业培养定制化
人员！



就业培训

为学校构建适应企
业的课程体系！



高校实训

千里之外也能获得
专业的培训指导！



在线教育

联系方式:

地 址: 厦门市思明区前埔西路146号 (云层天咨培训中心)

联 系 人: 陈霁、方文庆

联系电话: 0592-8265163/18602195793

邮 箱: wqfang@cloudits.info

请多多关照! 谢谢!

Git 和 Github 的简单使用

- 1. 到<https://github.com/>注册账号
- 2. github上新建project
- 3. 安装git客户端: <https://www.git-scm.com/>
- 4. 设置IDEA
- 5. git全局设置: `$ git config --global user.name 'name'`
`$ git config --global user.email 'email'`
- 6. 把本地项目添加成git项目: `$git init`
- 7. 上传代码到github: `$git add *` `$git commit -m "first commit"`
`$git remote add origin XXX.git` `$git push -u origin master`

Web UI 自动化基础入门

- Tomcat 简介
- Tomcat 下载与安装
- Tomcat 配置登录账号
- Jenkins 简介
- Jenkins 下载与安装
- Jenkins 常用系统配置
- Jenkins 插件安装
- Jenkins 构建Maven项目

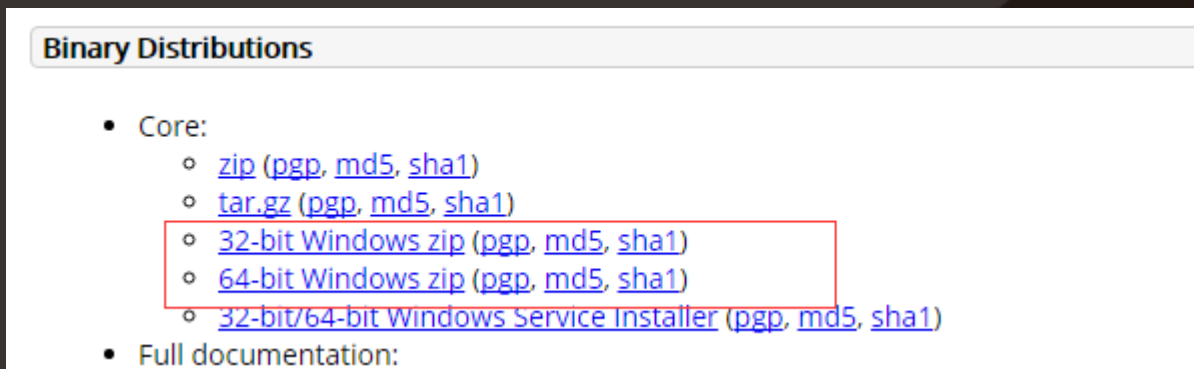
Tomcat 简介

Tomcat 服务器是一个免费的开源的、轻量的**Web**应用服务器。在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试**JSP**程序的首选。对于一个初学者来说，可以这样认为，当在一台机器上配置好**Apache**服务器，可利用它响应对**HTML**页面的访问请求。实际上**Tomcat**部分是**Apache**服务器的扩展，但它是独立运行的，所以当你运行**tomcat**时，它实际上作为一个与**Apache**独立的进程单独运行的。

Tomcat的安装

Tomcat 安装

1. 下载Tomcat 包: <http://tomcat.apache.org/>



2. 解压
3. 启动
双击 bin\startup.bat 文件
4. 浏览器访问: <http://localhost:8080/>

Tomcat 配置登录

1. 打开conf/tomcat-users.xml 文件
2. 添加如下代码:

```
<role rolename="manager-gui"/>  
<user username="admin" password="admin" roles="manager-gui"/>
```

Jenkins的安装

Jenkins简介:

Jenkins是一个开源软件项目，旨在提供一个开放易用的软件平台，使软件的持续集成变成可能。

Jenkins 下载地址: <https://jenkins.io/>

Jenkins启动:

1. 下载tomcat(<http://tomcat.apache.org/>)
2. 把jenkins.war 包放到tomcat的webapps下
3. tomcat 启动
4. 打开浏览器输入<http://localhost:8080/jenkins/>

pom.xml 添加插件

Maven pom.xml 添加插件：
该插件为编译插件：

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>3.3</version>
  <inherited>true</inherited>
  <configuration>
    <source>1.8</source>
    <target>1.8</target>
    <encoding>UTF-8</encoding>
  </configuration>
</plugin>
```

pom.xml 添加插件

Maven pom.xml 添加插件：
单元测试（执行）插件：

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-surefire-plugin</artifactId>
  <version>2.17</version>
  <configuration>
    <suiteXmlFiles>
      <suiteXmlFile>./testng.xml</suiteXmlFile>
    </suiteXmlFiles>
  </configuration>
</plugin>
</plugins>
```

更多详情查看：<http://maven.apache.org/surefire/maven-surefire-plugin/examples/testng.html>

Jenkins新建Maven Job

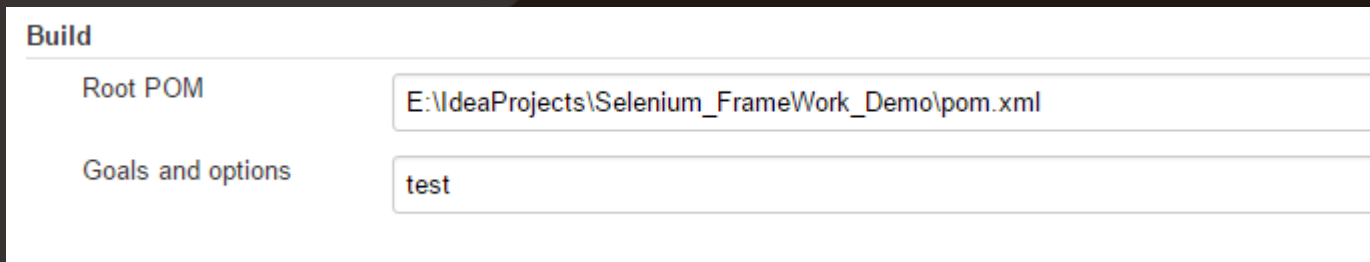
- 1. 点击“新建”，输入Item名称，选择“构建一个maven项目”，点击确定



The image shows the 'New Item' dialog box in Jenkins. At the top, there is a text input field for 'Item 名称'. Below it, there are five radio button options:

- ☐ **构建一个自由风格的软件项目**
这是Jenkins的主要功能。Jenkins将会结合任何SCM和任何构建系统来构建你的项目。甚至可以构建软件以外的系统。
- ☐ **构建一个maven项目**
构建一个maven项目。Jenkins利用你的POM文件。这样可以大大减轻构建配置。
- ☐ **External Job**
这个类型的任务允许你记录执行在外部Jenkins的任务。任务甚至运行在远程机器上。这可以让Jenkins作为你所有自动构建的代理。[详细内容](#)
- ☐ **构建一个多配置项目**
适用于多配置项目，例如多环境测试，平台指定构建，等等。
- ☐ **复制已有的Item**
要复制的任务名称

- 2. 在Build 栏目中输入你pom.xml文件的详细路劲



The image shows the 'Build' configuration section of a Jenkins job. It has a title 'Build' and two input fields:

- Root POM**:
- Goals and options**:

- 3. 点击保存，完成配置
- 4. 进入刚完成的job点击“立即构建”

Jenkins SVN的配置

1. SVN的配置

Source Code Management

☐ CVS

☐ None

☒ Subversion

Modules

Repository URL

https://wwwin-svn-sjwclouds.com/eng/

Unable to access https://wwwin-svn-sjwclouds.com/eng/ : svn: PROPFIND /eng failed (show details)
(Maybe you need to enter credential?)

Local module directory (optional)

Use 'svn update' as much as possible

Use 'svn update' whenever possible, making the build faster. But this causes the artifacts from the previous build to re when a new build starts.

2. 设置登录方式:

Subversion Authentication

Enter the authentication information needed to connect to the Subversion repository. This information will be stored in Jenkins.

Repository URL

svn的地址

☒ Username/password authentication

User name

账户名

Password

☐ SSH public key authentication (svn+ssh)

☐ HTTPS client certificate

OK

Jenkins 设置不同触发方式

☒ Build periodically

日程表

H 8 ***

Would last have run at 2016年7月9日 星期六 上午08时03分57秒 CST; would next run at 2016年7月10日 星期日 上午08时03分57秒 CST.

☒ Poll SCM

日程表

H/5 *****

Would last have run at 2016年7月9日 星期六 下午05时45分04秒 CST; would next run at 2016年7月9日 星期六 下午05时50分04秒 CST.

厦门云层天咨软件技术有限公司



Xiamen cloudits Consulting Software Technology Co. Ltd.

Jenkins 邮件设置


1. 进入系统设置，设置邮件通知：

邮件通知

SMTP服务器	smtp.163.com
用户默认邮件后缀	
<input checked="" type="checkbox"/> 使用SMTP认证	
用户名	[REDACTED]@163.com
密码
使用SSL协议	<input type="checkbox"/>
SMTP端口	
Reply-To Address	
字符集	UTF-8
<input type="checkbox"/> 通过发送测试邮件测试配置	

1. Jenkins Location中设置系统管理员邮件地址

Jenkins Location

Jenkins URL	http://localhost:8080/jenkins/  Please set a valid host name, instead of localhost
系统管理员邮件地址	<div>meyoungtester@163.com</div>

Jenkins 插件安装

1. 进入系统设置→管理插件

可更新 可选插件 已安装 高级		
安装	名称	版本
.NET Development		
<input type="checkbox"/>	CCM Plug-in This plug-in generates reports on cyclomatic complexity for .NET code.	3.1
<input type="checkbox"/>	Change Assembly Version This plug-in change the \[AssemblyVersion\] and \[AssemblyFileVersion\] from AssemblyInfo.cs sources.	1.5.1
<input type="checkbox"/>	FxCop Runner plugin FxCopCmd.exe execute plugin.	1.1
<input type="checkbox"/>	MSBuild Plugin This plugin allows you to use MSBuild to build .NET projects.	1.25
<input type="checkbox"/>	MSTest plugin This plugin converts MSTest TRX test reports into JUnit XML reports so it can be integrated with Jenkin's JUnit features. This plugin converts the .coveragexml files found in the project workspace to the EMMA format. You can use MSTestRunner plugin or VsTestRunner plugin to run the test and use this plugin to process the results.	0.19
<input type="checkbox"/>	MSTestRunner plugin This plugin allow you to execute test using MSTest command line tool.	1.2.0
<input type="checkbox"/>	NAnt Plugin This plugin allows for the execution of a NAnt build as a build step.	1.4.3
<input type="checkbox"/>	NCover plugin Archive and publish .NET code coverage HTML reports from NCover .	0.3
<input type="checkbox"/>	PowerShell plugin Integrates with Windows PowerShell	1.3
Violation Comments to Bitbucket Server Plugin		



联系方式:

地 址: 厦门市思明区前埔西路146号 (云层天咨培训中心)

联 系 人: 陈霁、方文庆

联系电话: 0592-8265163/18602195793

邮 箱: wqfang@cloudits.info

请多多关照! 谢谢!

为企业培养定制化
人员！



就业培训

为学校构建适应企
业的课程体系！



高校实训

千里之外也能获得
专业的培训指导！



在线教育

Web UI 自动化基础入门

- BDD 介绍
- Cucumber 介绍
- Cucumber 环境搭建
- Cucumber 实例演示
- 关键字驱动介绍



BDD 介绍

TDD:

TDD (Test-Driven Development) 测试驱动开发，是敏捷开发中的一项核心实践和技术，也是一种设计方法论。TDD的原理是在开发功能代码之前，先编写单元测试用例代码，测试代码确定需要编写什么产品代码。

BDD:

BDD (Behavior Driven Development) 行为驱动开发，建立在TDD基础之上，也是一种敏捷软件开发的技术。它鼓励软件项目中的开发者、QA和非技术人员或商业参与者之间的协作通过一种通用的语言来描述和讨论我们开发的系统。

ATDD:

ATDD (Acceptance Test Driven Development) 验收测试驱动开发

厦门云层天咨软件技术有限公司



Xiamen cloudits Consulting Software Technology Co. Ltd.

Cucumber 介绍

Cucumber:

Cucumber 是一个提供能让我们都理解的普通语言，通过普通语言来描述测试用例，并支持行为驱动开发的测试工具。**Cucumber**支持大多数编程语言如Ruby、Java和Python等。

官方地址: <https://cucumber.io/>

Cucumber 的特性

1.可执行性 (Executable)： 您可以像执行代码 (Java、Ruby...) 一样运行这些规范，来验证、验收目标应用。

2.规范性 (Specification)： 从非技术人员的视角触发，相比验证本身，他们更加关心系统功能的清晰描述：系统在什么场景下能够做什么样的事情。

如何使用 cucumber

大致步骤:

1. 编写feature文件
2. 生成steps
3. 运行测试用例

Cucumber Java版环境搭建

Pom文件引入相关包如下：

```
<dependency>
  <groupId>info.cukes</groupId>
  <artifactId>cucumber-java</artifactId>
  <version>1.2.5</version>
</dependency>
```

如果想通过Testng来运行，还需要添加如下：

```
<dependency>
  <groupId>info.cukes</groupId>
  <artifactId>cucumber-testng</artifactId>
  <version>1.2.5</version>
</dependency>
```

Cucumber 支持的语言输出内容

| **feature** | "功能" |

| **background** | "背景" |

| **scenario** | "场景", "剧本" |

| **scenario outline** | "场景大纲", "剧本大纲" |

| **examples** | "例子" |

| **given** | "*" , "假如", "假设", "假定" |

| **when** | "*" , "当" |

| **then** | "*" , "那么" |

| **and** | "*" , "而且", "并且", "同时" |

| **but** | "*" , "但是" |

| **given (code)** | "假如", "假设", "假定" |

| **when (code)** | "当" |

| **then (code)** | "那么" |

| **and (code)** | "而且", "并且", "同时" |

| **but (code)** | "但是" |

Cucumber 实例演示

1. 新建firstfeature.feature文件

```
#language: zh-CN
```

功能： 用户登录

为了测试登录功能是否正常

@Automtion

场景： 正确的用户账号登陆

假如我已经打开"<http://mail.163.com/>"网站

当我用账号"[XXXX](#)",密码"[XXXX](#)"作为登录账号

当我点击登录按钮

那么我登录成功

@Automtion

场景大纲：： 正确的用户账号登陆

假如我已经打开"<http://mail.163.com/>"网站

当我用账号"<[username](#)>",密码"<[pwd](#)>"作为登录账号

当我点击登录按钮

那么我登录成功

例子：

username	pwd
aaa	111
bbb	222
ccc	333



Cucumber 实例演示

2. 实现Steps

```
@假如("^我已经打开\"(.*)\"网站$")
public void 我已经打开_网站(String url) {
    OpenBrowser.openBrowser(url);
}

@当("^我用账号\"(.*)\",密码\"(.*)\"作为登录账号$")
public void loginSteps(String userName, String pwd) {
    Login.login(userName, pwd);
}

@当("^我点击登录按钮$")
public void clickLoginButton() {
    driver.findElement(LoginPage.dologinButton).click();
}

@那么("^我登录成功$")
public void loginSuccess() {
    Login.success();
}
```



Cucumber 实例演示

3. Webdriver 部分自动化代码实现

```
public class Login extends StepsDemo {  
    public static void login(String userName, String pwd) {  
        driver.switchTo().frame("x-URS-iframe");  
        driver.findElement(LoginPage.userNameTextBox).sendKeys(userName);  
        driver.findElement(LoginPage.pwdTextBox).sendKeys(pwd);  
    }  
}
```

```
public class OpenBrowser extends StepsDemo {  
    public static void openBrowser(String url) {  
        System.setProperty("webdriver.chrome.driver", ".\\drivers\\chromedriver");  
        driver = new ChromeDriver();  
        driver.get(url);  
    }  
}
```

Cucumber 实例演示

通过Testng 来运行

```
import cucumber.api.CucumberOptions;
import cucumber.api.testng.AbstractTestNGCucumberTests;

@CucumberOptions(
    features = "src/test/resources/first_feature.feature",
    format = {"pretty",
        "html:target/site/cucumber-pretty",
        "rerun:target/rerun.txt",
        "json:target/cucumber1.json"},
    tags = {"~@unimplemented"},
    glue = {"demo.cucumber.steps"})
public class RunDemo extends AbstractTestNGCucumberTests {
}
```

Cucumber 在Jenkins上运行

1. pom.xml 添加插件

2. Jenkins 安装插件: [Cucumber-JVM reports](#)

3. Job 配置: maven构建方式, 测试报告生成

关键字驱动

Robot Framework

Robot Framework是一款python编写的功能框架。具备良好的可扩展性，支持关键字驱动，可以同时测试多种类型的客户端或者接口，可以进行分布式测试执行。

baidu_search				
Settings >>				
1	Open Browser	http://www.baidu.com	chrome	
2	Input Text	id=kw1	robot framework学习	
3	Click Button	id=su1		
4	sleep	2		
5	close Browser			
6				

RobotFramework 验证

AssertTest

Settings >>

1	open browser	http://www.baidu.com	chrome	
2	\${title}	gettext		
3	shouldcontain	\${title}	百度一下, 你就知道	
4	sleep	5		
5	closebrowser			
6				
7				
8				
9				
10				

RobotFramework 变量声明

TestCaseValue				
Settings >>				
1	<code>\${value1}</code>	SetVariable	rft	
2	log	<code>\${value1}</code>		
3				

RobotFramework if 语句

ifTest					
Settings >>					
1	<code>\${a}</code>	setvariable	1		
2	<code>\${b}</code>	setvariable	0		
3	run keyword if	<code>\${a}>\${b}</code>	log	<code>\${a}</code>	
4	...	ELSE IF	<code>\${b}==0</code>	log	b等于0
5	...	ELSE	log	b不等于0	
6					

RobotFramework 循环语句

forTest				
Settings >>				
1	:FOR	<code>\${I}</code>	in range	10
2		log	<code>\${I}</code>	
3				

为企业培养定制化
人员！



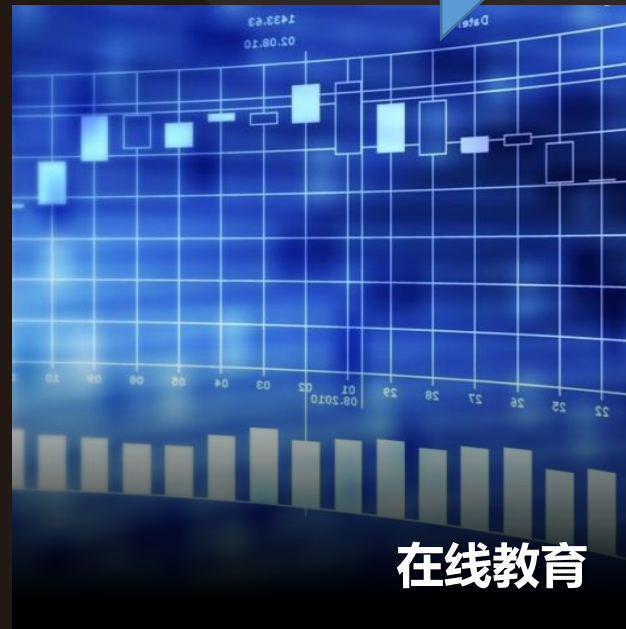
就业培训

为学校构建适应企
业的课程体系！



高校实训

千里之外也能获得
专业的培训指导！



在线教育

联系方式:

地 址: 厦门市思明区前埔西路146号 (云层天咨培训中心)

联 系 人: 陈霁、方文庆

联系电话: 0592-8265163/18602195793

邮 箱: wqfang@cloudits.info

请多多关照! 谢谢!