Selenium WebDriver阶段八 - (分布并行测试Grid，json实施node节点)

WebDriver
(阶段八)

课程目的：学习使用Grid实现测试任务分配；
培训结果：了解json,会使用Grid实现本地多浏览器执行脚本或远程执行脚本
课程相关脚本：PracticeTen
作者：Terry

QQ:314768474

个人微博：http://weibo.com/alwaysterry

Windows下文件名或目录的简写方法

经常碰到Windows中的路径变成这样的形式：

```
D:\IBM\WCDE_E~1
```

一直以为这种路径是随机的，今天才发现原来这类路径也是有效的，它其实是有缩写规则的，美其名曰"DOS 8.3命名规则"，详情可见微软官方文档。

这种命名规则简单说来是用8个字符缩写来代替文件（或目录）全名，对于目录，可以写头六个字母（略去空白），另加波浪号和1；如果首字母不足六个字母，略去空格，用了第二个词的字母，凑成六个，再按规则继续处理。

例如：

```
Documents and Settings
```

可表示为

```
DOCUME~1
```

如果多个文件前6字符一样，则按dir中的输出顺序累计下去。

假设下面是你的C盘根目录中的文件夹：

```
Program Files
Program Files (x86)
```

则三个目录分别表示为：

```
PROGRA~1
PROGRA~2
```

```
Runtime.getRuntime().exec("cmd /c  start "+startnode2+" "+seleniumserverstandalone+" "+node2json+" "+"C:\\PROGRA~2\\MOZILL~1\\firefox.exe")
```

1)Json

- JSON 指的是 JavaScript 对象表示法（**J**ava**S**cript **O**bject **N**otation）
- JSON 是轻量级的文本数据交换格式
- JSON 独立于语言 *
- JSON 具有自我描述性，更易理解

我们看看一下，下面的例子：

```
{
"employees": [
{ "firstName":"Bill" , "lastName":"Gates" },
{ "firstName":"George" , "lastName":"Bush" },
{ "firstName":"Thomas" , "lastName":"Carter" }
]
}
```

这里的employees是一个对象，他有3组属性每组呢，包含两个属性：firstName,lastName;这两个属性都有对应的值。
我们写json的目的是简化我们在传递给selenium-server-standalone.jar的一些参数配置：

2)Grid的实现方式是通过selenium-server-standalone来实现的，首先你需要通过下面的地址把最新的selenium-server-standalone下载下来。

以下是selenium-server-standalone的下载地址：

https://code.google.com/p/selenium/downloads/list

grid实现的第一步需要开启hub server，然后各个远程node都需要连上这个hub server。

①你找一台机器把该机器作为hub server，然后运行一下命令：

java -jar selenium-server-standalone.jar -role hub -hubConfig %2

这里有个%2需要我们手动去传入一个json文件，这个json文件就是一个hub server的配置信息，注意这里的端口号。注意这里的role是hub。

```
{
  "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": 5
}
```

②我们还需要找一台node,作为远程任务分配机器，然后运行一下命令：

java -jar selenium-server-standalone.jar -role node -Dwebdriver.chrome.driver=chromedriver.exe -nodeConfig %2

这里有个%2需要我们手动去传入一个json文件，这个json文件就是一个node的配置信息。

注意这里的端口号。
注意这里的role是node。
注意这里采用的协议，两种selenium,webdriver

```
{
  "capabilities":
      [
        {
          "browserName": "firefox",
          "maxInstances": 5,
          "seleniumProtocol": "WebDriver"
        },
        {
          "browserName": "chrome",
          "maxInstances": 5,
          "seleniumProtocol": "WebDriver"
        },
        {
          "browserName": "internet explorer",
          "maxInstances": s,
          "seleniumProtocol": "WebDriver"
        }
      ],
  "configuration":
  {
    "proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy",
    "maxSession": 5,
    "port": 5555,
    "host": ip,
    "register": true,
    "registerCycle": 5000,
    "hubPort": 4444,
    "hubHost": ip
  }
}
```

capabilities对象中设置你可能会执行的browsers，这样，只要你修改-Dwbedriver就可以了。再看一下这里的configureation中的hubPort就是hubserver上开启的4444端口号。

这里需要注意的是如果开启中心或node出现端口号被占用那么，要么改端口号，要么通过下面的命令（这里是windows）

a,netstat -ano 找到端口号被占用的pid
b,tasklist 找到对应的pid的程序
c,通过任务控制面板去kill掉

中心与node的连接需要将防火墙关闭。

③以上的json和启动hub,node我们可以写到一个bat文件中，这样的话，我们就可以批量处理了。

3)在理解了上面的知识后，下面我们就要去实施了。

①我们要学习使用类RemoteWebDriver，它的构造函数如下：
public RemoteWebDriver(URL remoteAddress, Capabilities desiredCapabilities)

这样我们就可以对**driver**进行初始化了。这里的**url**是你**node**节点的地址一般就是**http://ipaddress:port/wd/hub**

②我们在一个类中定义个@BeforeSuite中先对hub进行开启操作操作：

```java
public class MutipleTasks{

    private String seleniumserverstandalone  = null;
    private String hubjson                   = null;
    private String projectpath               = null;
    private String starthub                  = null;



    @BeforeSuite()
    public void beforesuite(){

        projectpath = System.getProperty("user.dir");
        seleniumserverstandalone = projectpath+"\\tool\\selenium-server-standalone-2.jar";
        hubjson    =  projectpath+"\\src\\practiceten\\config\\hub.json";
        starthub   = projectpath+"\\tool\\starthub.bat";

        try {
            Runtime.getRuntime().exec("cmd /c  start "+starthub+" "+seleniumserverstandalone+" "+hubjson);
            Thread.sleep(30000);

        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }


    }



}
```

这里自动开启 hubserver

③我们将所有的remote driver设置放到一个BrowserType中这样在需要哪个deriver的时候就可以拿来用了。

BrowsersType.java | MutipleTasks.java | TestCase1.java | onepcwithtwobrowser.xml

```java
package practiceten;

import java.io.File;

public class BrowsersType {
    private WebDriver driver = null;
    private Wait wait =  null;
    private String projectpath = System.getProperty("user.dir");
    private DesiredCapabilities caps;



    public WebDriver setFirefox(String nodeurl){
        caps = DesiredCapabilities.firefox();
        File firebug = new File(projectpath+"/tool/firebug-1.12.1-fx.xpi");
        File firepath = new File(projectpath+"/tool/firepath-0.9.7-fx.xpi");
        FirefoxProfile firefoxprofile =  new FirefoxProfile();
        try {
            firefoxprofile.addExtension(firebug);
            firefoxprofile.addExtension(firepath);
            firefoxprofile.setPreference("webdriver.accept.untrusted.certs", "true");
            firefoxprofile.setPreference("extensions.firebug.currentVersion", "1.12.1");
            caps.setCapability(FirefoxDriver.PROFILE, firefoxprofile);
            driver = new RemoteWebDriver(new URL(nodeurl), caps);
            wait = new Wait(driver);
            driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
            driver.manage().window().maximize();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return driver;
    }

}
```

④接下来我们要写一个测试用例需要这个用例在一个是ff，一个是chrome上运行,这里们假设需要开启两个node

```java
public class TestCase1 extends MutipleTasks{

    private String projectpath = System.getProperty("user.dir");
    private  WebDriver driver;
    private Wait wait;

    @Parameters({"node","browser"})
    @BeforeClass
    public void beforeClass(@Optional("")String node,@Optional("") String browser){

        BrowsersType bt = new BrowsersType();

        if(browser.equals("ff"))
            driver = bt.setFirefox(node);
        else if(browser.equals("chrome"))
            driver = bt.setChrome(node);
        else
            driver = bt.setIE(node);

        wait =  new Wait(driver);
    }

    @Test
    public void selectItemFromDropDownList(){
        //登录jd官方网站
        driver.get("http://www.jd.com");
        wait.waitForElementPresent("//a[text()='[登录]']");
        driver.findElement(By.xpath("//a[text()='[登录]']")).click();
        wait.waitFor(5000);
    }
```

这里就调用了上面的类去设置不同的browsers

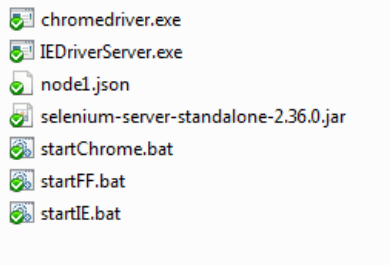
一个简单的case

我们看一下testsuite.xml文件中是怎么设置的：
这里重复设置了一个test，变更一个browser,node的值就可以将测试用例执行两次。分别在不同的node上，这里可以自行替换localhost和port.
注意parameter的位置哦。

```xml
<test name="FireflyAutomation1 examples"  preserve-order="true" >
    <parameter name="node" value="http://localhost:5555/wd/hub"/>
    <parameter name="browser" value="chrome"/>
    <classes>
        <class name="practiceten.testcases.TestCase1" >
            <methods>
              <include name="selectItemFromDropDownList"/>
             </methods>
       </class>
    </classes>
</test>


<test name="FireflyAutomation2 examples"  preserve-order="true" >
    <parameter name="node" value="http://localhost:6666/wd/hub"/>
     <parameter name="browser" value="ff"/>
    <classes>
        <class name="practiceten.testcases.TestCase1" >
            <methods>
              <include name="selectItemFromDropDownList"/>
             </methods>
       </class>
    </classes>
</test>
```

④我们提供了一个这样的工具放在tool中一个是node1,一个是node2根据你的xml文件中设置的你可以运行你的相应的bat：



这里我们用到三种类型的browser依次我们可以写到不同bat中

bat文件类似与这样。

java -jar selenium-server-standalone-xxx.jar -role node -Dwebdriver.？.driver=？ -nodeConfig node1.json

在“②中我们在一个类中定义个@BeforeSuite中先对hub进行开启操作操作”-----这个操作完后，我们就可以使用上面的bat开启执行你的@Test内的东西了