

Selenium WebDriver阶段二 - (数据驱动,properties的数据存储)

WebDriver (阶段二)

课程目的：掌握如何解析properties文件。

培训结果：会对selenium做简单的数据驱动。

课程相关脚本：practicefour

作者：Terry

QQ:314768474

个人微博：<http://weibo.com/alwayserry>

版权所有禁止传播。

1)在测试过程中往往需要对测试脚本进行参数化，所以TestNG中提供了可以参数的annotation进行参数化
关键字@Parameters({"p1","p2","p3"}),@Optional("")

下载FireflyAutomation项目的最新代码：

找到包practicefour:

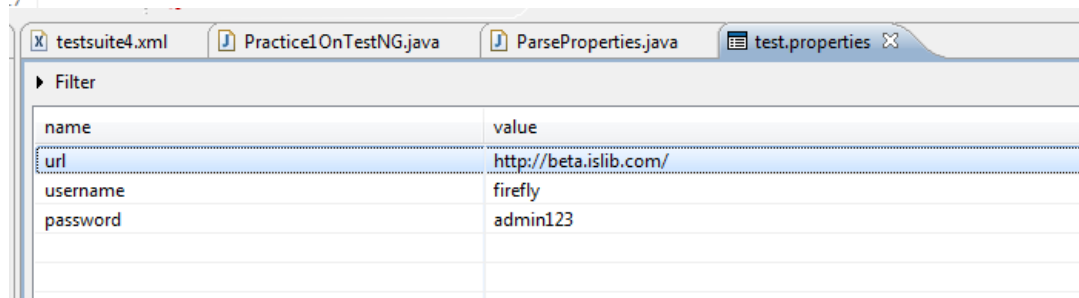
如果将以上的关键字用到实际的脚本：

a)testsuite.xml文件中添加如下，name和value;当然也可以多个parameter,注意name要保持唯一

```
<suite name="FireflyAutomation" >
  <parameter name="TestData" value="\tool\test.properties" />
  <test name="Practice1" preserve-order="true" >
    <classes>
      <class name="practicefour.Practice1OnTestNG" >
        <methods>
          <include name="test"/>
        </methods>
      </class>
    </classes>
  </test>
</suite>
```

2)用例脚本中添加

```
4
5 public class Practice1OnTestNG {
6
7   @Parameters({"TestData"})
8   @Test
9   public void test(@Optional("aaa") String testdata){
10     ParseProperties pp = new ParseProperties(System.getProperty("user.dir")+testdata);
11     System.out.println(pp.getTestData("url"));
12     System.out.println(pp.getTestData("username"));
13     System.out.println(pp.getTestData("password"));
14   }
15
16
17 }
```



Filter	
name	value
url	http://beta.islib.com/
username	firefly
password	admin123

@Parameters({"p1","p2","p3"})

@Test

```
public void searchMerchandise(@Optional("film") String book, String elec,String life){
}
}
```

这里额外讲一下关于引进properties的用法，properties文件主要是key->value的一个存储类型的文件。java中有对properties文件的操作，这里我已经提供一个加载properties的方法，和取值的方法。

这里我们可以联想到什么呢？是不是我的测试数据和xpath都可以放到这个properties呢？

大家可以看一下一下的这个类

```
1 package practicefour;
2
3 import java.io.FileInputStream;
4 import java.io.IOException;
5 import java.io.InputStream;
6 import java.util.Properties;
7
8 public class ParseProperties {
9
10     private Properties pro = new Properties();
11
12     public ParseProperties(String propertiespath){
13         this.loadProperties(propertiespath);
14     }
15
16     private void loadProperties(String propertiespath){
17
18         try {
19             InputStream in = new FileInputStream(propertiespath);
20             pro.load(in);
21         } catch (IOException e) {
22             // TODO Auto-generated catch block 加载properties文件
23             e.printStackTrace();
24         }
25     }
26
27     //to get value of specific keyname
28     public String getTestData(String keyname){
29         return pro.getProperty(keyname).trim();
30     }
31 }
32
33 获取properties文件中的特定value值
```

我们在脚本中已经添加实例化了一个pp,然后我们用testng去运行那个xml文件看看有什么结果:

结果是打印出：url,username,password对应的值。

习题：试着将脚本test完善一下，

- 1) 要求参数化url：126.com，及用户名密码。做登陆操作。
- 2) 要求xpath都写到properties文件中。

2)在测试过程中需要对一些功能点进行验证，所以这个需要一个假定Assert

脚本在运行结束后如果没有抛出异常那么testng将此测试用例即作为通过，而实际我们需要自己的判断来衡量一个测试用例是不是通过。testng中提供了Assert类中很多静态方法我们这里就使用几个。

现在有个需求比如登陆一号店输入一个特定的商品我们验证该商品是不是出现：

我们的思路，既然之前我们用了properties，我们就将xpath和数据都放入properties的不同文件中。

```

private ParseProperties data = new ParseProperties(System.getProperty("user.dir")+"\\tool\\test.properties");
private ParseProperties locator = new ParseProperties(System.getProperty("user.dir")+"\\tool\\locators.properties");

@BeforeClass
public void startFirefox(){
    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 = new FirefoxProfile();
    try {
        firefoxprofile.addExtension(firebug);
        firefoxprofile.addExtension(firepath);
        firefoxprofile.setPreference("webdriver.accept.untrusted.certs", "true");
        firefoxprofile.setPreference("extensions.firebug.currentVersion", "1.12.1");
        ffwb = new FirefoxDriver(firefoxprofile);
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

@Test
public void searchMerchandise(){
    ffwb.get(data.getValue("url"));
    ffwb.findElement(By.xpath(locator.getValue("SearchBox"))).sendKeys(data.getValue("merchandise"));
    ffwb.findElement(By.xpath(locator.getValue("SearchBtn"))).click();
    WebElement we = ffwb.findElement(By.xpath(locator.getValue("Item")));
    Assert.assertEquals(we.isDisplayed(), false);
}
}

```

locator.properties文件

name	value
SearchBox	//input[@name='q']
Item	//a[contains(text(),'Adolescent Schizophrenia')]
SearchBtn	//input[@type='submit']

3)在测试过程中需要等待一个页面或一个元素加载，我们建议不要使用Thread.sleep()来做，这样的话一效率问题，二稳定性问题。

我们可能碰到什么情况了

- 1) 等待一个元素的加载
- 2) 等待一个元素有效

基本这两个是常用的，也可以覆盖90%的等待问题

全局等待超时

`ffwb.manage().timeouts().implicitlyWait(3000, TimeUnit.SECONDS);`//定义全局的等待超时时间。

我们可以将所有等待放到一个这个类我们叫做Wait中这样我们就可以直接使用了：

```

public class Wait {
    private WebDriver driver;

    public Wait(WebDriver driver){
        this.driver = driver;
        PageFactory .initElements(driver, this);
    }

    public void waitForElementPresent(String locator){
        (new WebDriverWait(driver, 10)).until(ExpectedConditions.presenceOfElementLocated(By.xpath(locator)));
    }

    public void waitForElementIsEnable(String locator){
        (new WebDriverWait(driver, 10)).until(ExpectedConditions.elementToBeClickable(By.xpath(locator)));
    }

    public void waitFor(long timeout){
        try {
            Thread.sleep(timeout);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

```

然后我们就可以使用wait了，不用担心元素是否不出现而导致一些不稳定的问题。
 我们可以在上面的例子中

添加这样一个等待，也就是在search出来的商品可能需要一定的时间，这样我们就需要给定一个超时等待：

```

@Test
public void searchMerchandise(){
    Wait wait = new Wait(ffwb);
    ffwb.get(data.getValue("url"));
    ffwb.findElement(By.xpath(locator.getValue("SearchBox"))).sendKeys(data.getValue("merchandise"));
    ffwb.findElement(By.xpath(locator.getValue("SearchBtn"))).click();
    WebElement we = ffwb.findElement(By.xpath(locator.getValue("Item")));
    wait.waitForElementPresent(locator.getValue("Item"));
    Assert.assertEquals(we.isDisplayed(), false);
}

```

1)作业草稿箱判段信显示的个数是否与我们打开的草稿箱的信个数一致，用browsers类,assert,wait ;
 getText();
 findElements;
 List<WebElement>