

Selenium WebDriver阶段五 - (页面对象化及动态xpath)

WebDriver (阶段五)

课程目的：掌握pagefactory的用法，及动态xpath。TestNG的测试报告美化

培训结果：在一个电子商务的项目中能够熟练使用以上两种方法。

课程相关脚本：PracticeSeven

作者：Terry

QQ:314768474

个人微博：<http://weibo.com/alwaysterry>

版权所有禁止传播。

PageFactory的使用能够将我们的脚本的设计从简单的过程到页面对象化转移：

Selenium提供一个pagefactory，用法如下图：

静态initElements()实现页面对象的初始化，这样避免产生空值的情况下，又提供给我们@FindBy和@FindBys

对于@FindBy类似于driver.findElement(By.id, xpath)等

对于@FindBys类似于driver.findElements(By.id...)等

这里我就不用看@FindBy，而是@FindBys

我看一下以下的一个场景一般我注册一个用户的时候都会要求输入两次密码，

The screenshot shows a registration form with the following elements:

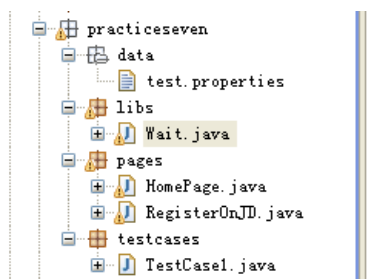
- A label "* 账户名：" followed by a text input field and a button labeled "请输入邮箱/用户名/手机号".
- A red rectangular box highlights two password input fields:
 - * 请设置密码:
 - * 请确认密码:
- A yellow callout box with a lock icon points to the password fields, containing the text: "这里的输入的密码是一样的，所以我们可以放到一个操作中".
- A checkbox labeled "我已阅读并同意 《京东用户注册协议》".
- A red button labeled "立即注册".

4

，为了节省这样的冗余操作，我们可以这么做，我们设计这样一个注册jd的类，将passwd用@FindBys这样我们可以对passwds进行一个方法的操作。

在我们做自动化开发的时候很多时候都会碰到List，希望大家对于这种情况多加思索。

我们的项目结构：



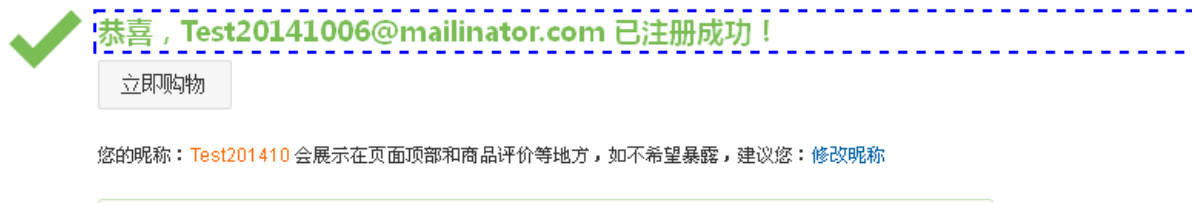
我们看一下testcase是怎么设计的：

```
@Test
public void regAccount(){
    //Wait wait = new Wait(ffwb);
    ffwb.get("http://www.jd.com/");
    RegisterOnJD reg = new RegisterOnJD(ffwb);
    HomePage homepage = new HomePage(ffwb);

    homepage.register();
    reg.setAccountName(data.getValue("accountname"));
    reg.setPassword(data.getValue("password"));
    reg.submit();
}
```

这里我们看到就只有一个setPassword方法。

当我们注册完一个用户后会有一个这样的提示，告知用户你已经成功注册，那么我事先写得xpath是不是还适用呢，这里我们就要怎么设计动态的xpath



我们将那个注册类改一下，加一些东西进去

```

public void setAccountName(String accountName) {
    this.accountname.sendKeys(accountName);
    setWebElement(accountname);
}

```

我们在设置accountname的时候调用了setWebElement

```

public void setPassword(String password) {
    for (WebElement passwd:passwords) {
        passwd.sendKeys(password);
    }
}

```

```

public void submit() {
    submit.click();
}

```

仔细研究一下setWebElement的replace方法

//动态设置webpage上的元素xpath

```

public void setWebElement(String value) {
    String locator = "//div[text()='恭喜, %var% 已注册成功!']";
    locator = locator.replace("%var%", value);
    regfinished = driver.findElement(By.xpath(locator));
}

public WebElement getWebElement() {
    return regfinished;
}

```

```

@Test
public void regAccount() {
    //Wait wait = new Wait(ffwb);
    ffwb.get("http://www.jd.com/");
    RegisterOnJD reg = new RegisterOnJD(ffwb);
    HomePage homepage = new HomePage(ffwb);

    homepage.register();
    reg.setAccountName(data.getValue("accountname"));
    reg.setPassword(data.getValue("password"));
    reg.submit();
    Assert.assertEquals(reg.getWebElement().isDisplayed(), true);
}

```

- 1) beforeclass内容放到libs中
- 2) jd购买流程

TestNG的测试报告美化

首先下载testng-xslt

<https://docs.google.com/file/d/0B3OQaZlrVrdJZUtSMWJseGVzVGs/edit?pli=1>

将zip文件解压后,找到如下几个文件:

testng-xslt-1.1.1\lib\saxon-8.7.jar

testng-xslt-1.1.1\src\main\resources\testng-results.xsl

然后到github上更新fireflyautomation项目拿到最新代码

放入自己的项目,这里我就放入项目中的tool,libs中,可根据自己的项目自行放置。

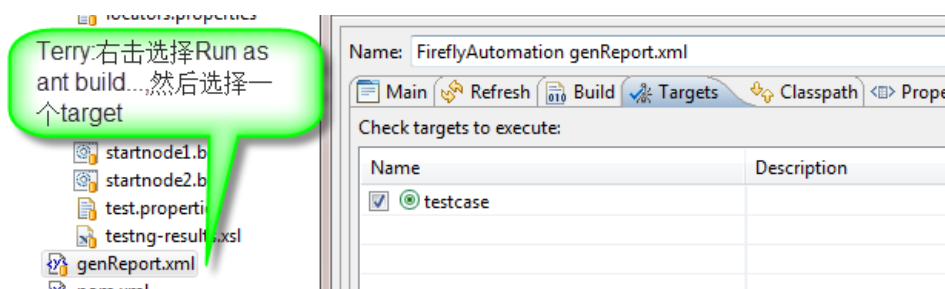
```

<xslt in="${basedir}/test-output/testng-results.xml" style="${basedir}/tool/testng-results.xsl" out="${basedir}/output/index.html"
  <param name="testNgXslt.outputDir" expression="${basedir}/output/" />
  <param name="testNgXslt.sortTestCasesLinks" expression="true" />
  <param name="testNgXslt.testDetailsFilter" expression="FAIL,SKIP,PASS" />
</xslt>

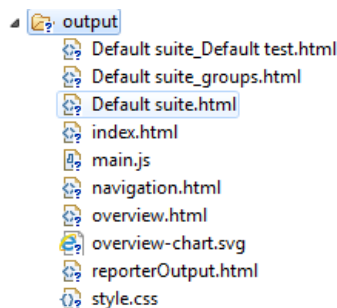
```

Terry:ant文件中添加这样的句子。

我们先运行一下practiceten_1的10个简单的测试用例，run as by testng后会在项目中产生一个test-output目录。



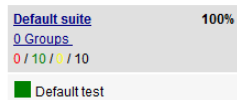
在上图被运行后，会产生一个output的目录我们根据ant脚本可以看到有个index.html文件生成：



以任意你本地的浏览器打开得到下面的截图

TestNG Results

[Results overview](#)
[Reporter output](#)



Test suites overview

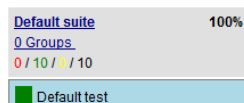
SVG Pie Charts are not supported. Terry: :) Please install a SVG viewer for your browser.

Default suite	0	10	0	10	100%
Default test	0	10	0	10	100%

Generated with TestNG XSLT

TestNG Results

[Results overview](#)
[Reporter output](#)



☐ Group by class

☐ All

☒ Failed ☒ Passed ☒ Skipped ☐ Config

Name	Started	Duration	Exception
test1()	15:30:36	15 ms	
Name: test1 Signature: test1([pri:0, instance:, practicen_1.TestngXlst@9d04dc] Start time: 15:30:36 End time: 15:30:36 Duration: 15 ms			
test10()	15:30:36	1 ms	
test2()	15:30:36	1 ms	
test3()	15:30:36	0 ms	
test4()	15:30:36	0 ms	
test5()	15:30:36	0 ms	
test7()	15:30:36	1 ms	
test6()	15:30:36	0 ms	
test8()	15:30:36	1 ms	
test9()	15:30:36	0 ms	

Terry:可以过滤你想要的测试结果


TestNG Results

[Results overview](#)
[Reporter output](#)

[Default suite](#) 90%

[0 Groups](#)

1 / 9 / 0 / 10

 Default test

Test case Default test

☐ Group by class

☐ All

☒ Failed ☒ Passed ☒ Skipped ☒ Config

Name

test9()

Name: test9

Signature: test9()[pri:0, instance:practiceten_1.TestngXlst@1b347747]

Start time: 16:49:29

End time: 16:49:29

Duration: 3 ms

test1()

test10()

test2()

test3()

test4()

test5()

test6()

test7()

test8()

Selenium对象高亮处理

```
public void highlightElement(WebDriver driver, WebElement element) {  
    JavascriptExecutor js = (JavascriptExecutor) driver;  
    js.executeScript("element = arguments[0];" +  
        "original_style = element.getAttribute('style');" +  
        "element.setAttribute('style', original_style + \";" +  
        "background: yellow; border: 2px solid red;\"");  
    "setTimeout(function(){element.setAttribute('style', original_style);}, 1000);", element);  
}
```