Selenium WebDriver阶段一 - (脚本流程及各browsers启动配置profile,证书问题等)

WebDriver+TestNG
(阶段一)

课程目的：学习testng的流程控制管理，设置ff,ie,chrome的profile及装入webdriver去启动。
培训结果：熟悉这3大主流的browsers的启动方式。
课程相关脚本：PracticeOne,PracticeTwo
作者：Terry

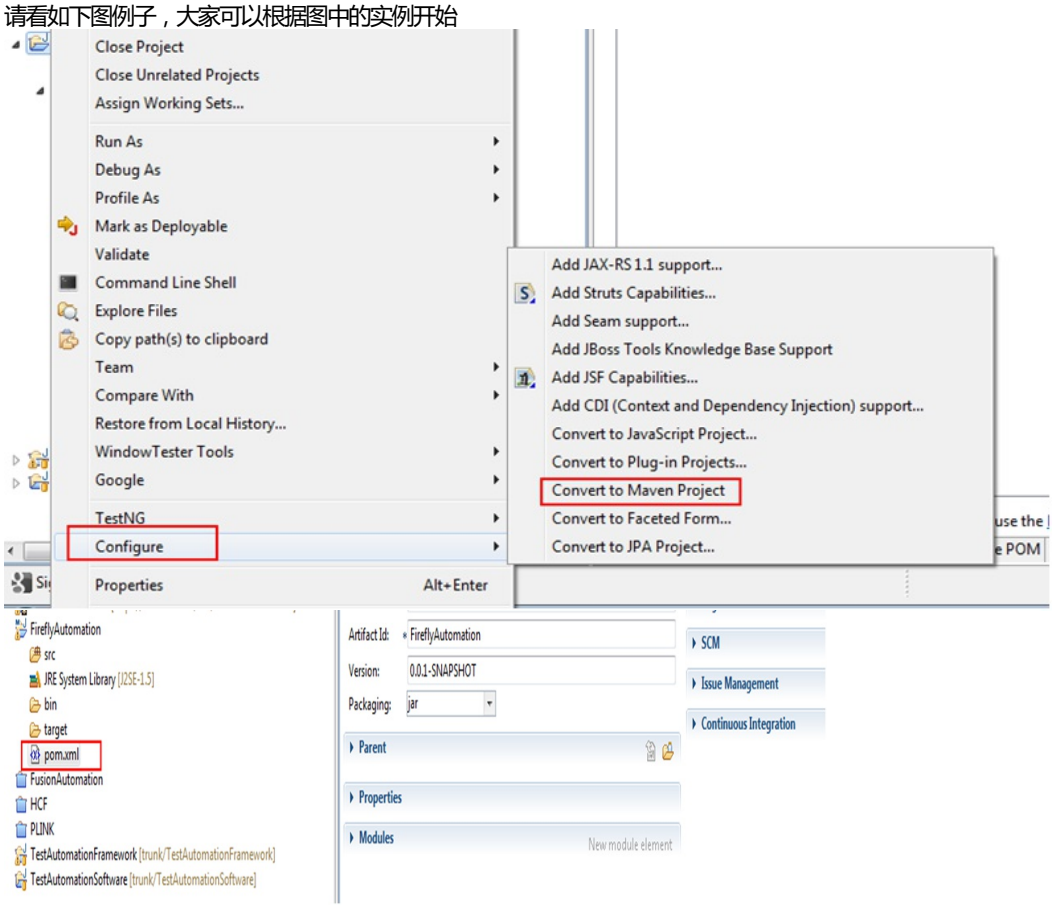
QQ:314768474

个人微博：http://weibo.com/alwaysterry

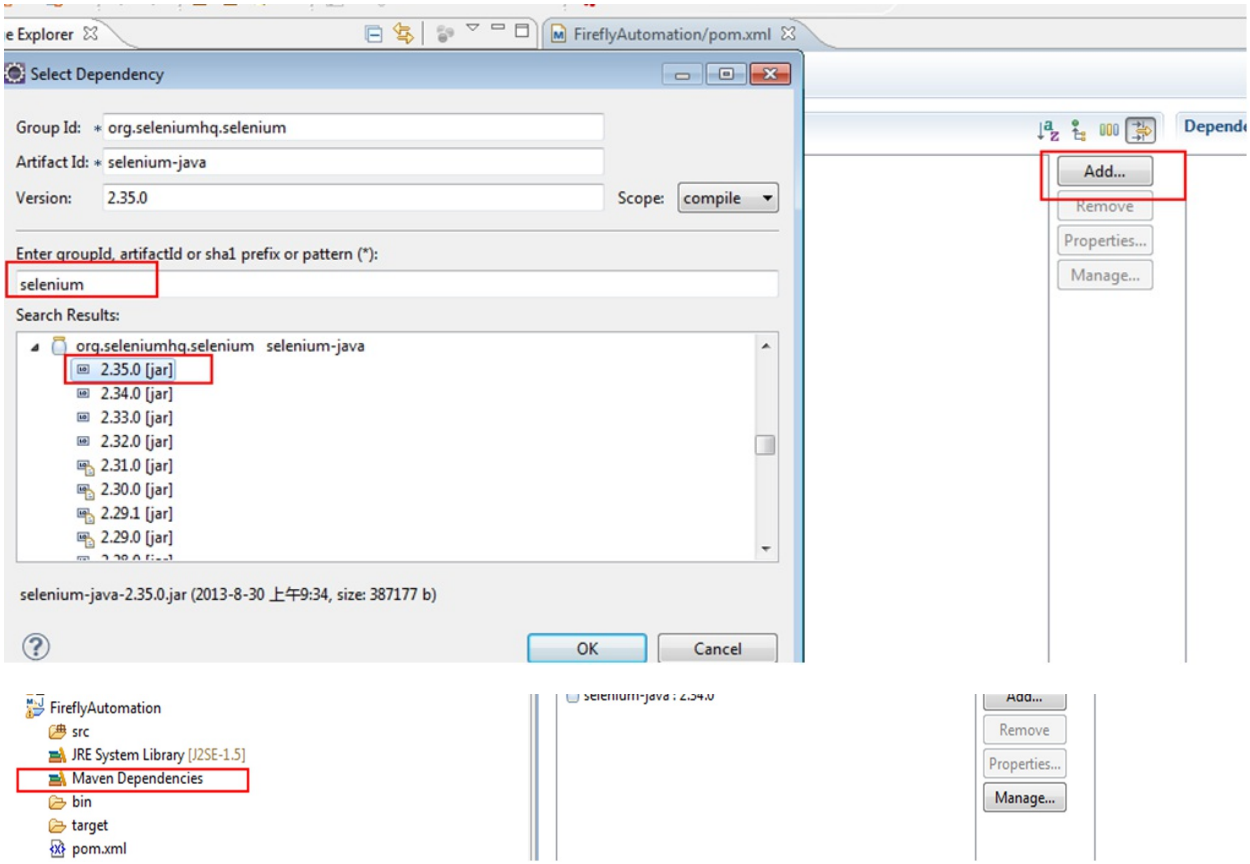
一）怎么样搭建建一个简单的Maven项目：

1)新建一个普通的java 项目
2)将java项目转换为Maven项目

请看如下图例子，大家可以根据图中的实例开始



二）Maven项目中添加jar包：



试着动手做：添加一个log4j的jar包到你项目中去。

三）Testng简单描述：

以下列出的特性是需要牢记的。

Annotations.
注解化编程。

Run your tests in arbitrarily big thread pools with various policies available (all methods in their own thread, one thread per test class, etc...).
测试用例可以以多线程的方式以某种特有的方案运行。

Flexible test configuration.
灵活配置执行用例。

Support for data-driven testing (with @DataProvider).
数据驱动。

Support for parameters.
可参数化。

Powerful execution model (no more TestSuite).
模块化。

Supported by a variety of tools and plug-ins (Eclipse, IDEA, Maven, etc...).
适用性。

四）Testng一个简单例子：

请下载实例代码FireflyAutomation项目

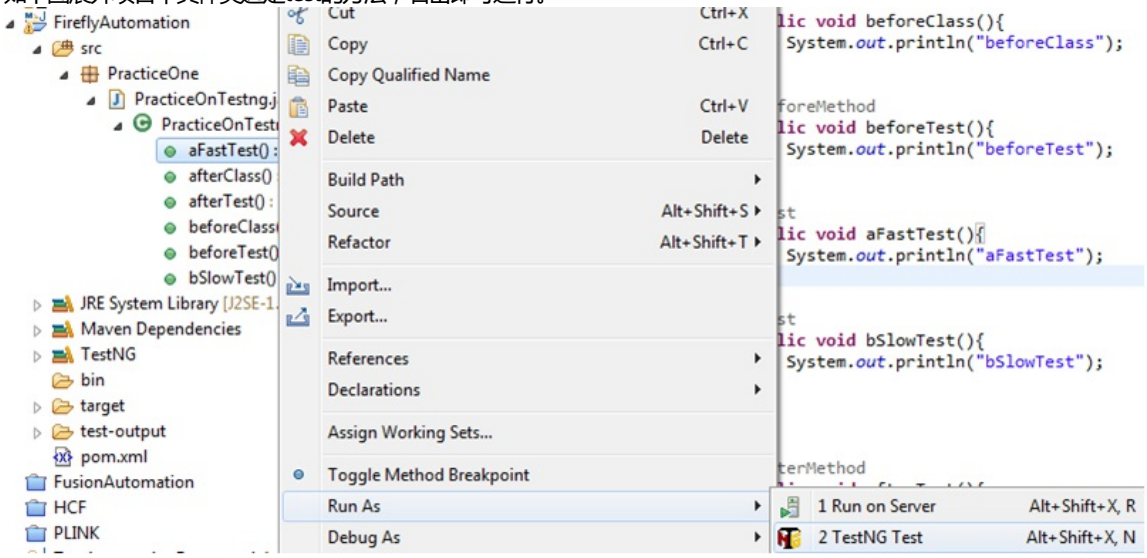
@Test即为需要运行的测试用例。运行顺序不定。
@BeforeClass/AfterClass即为该类文件需要预先或后置运行的条件，通常用来初始化一些变量或用来设置后置条件如用户退出等。

@BeforeClass与AfterClass只被运行一次。
@BeforeMethod和AfterMethod是每个test执行前和后都需要执行的预先条件和后置条件。

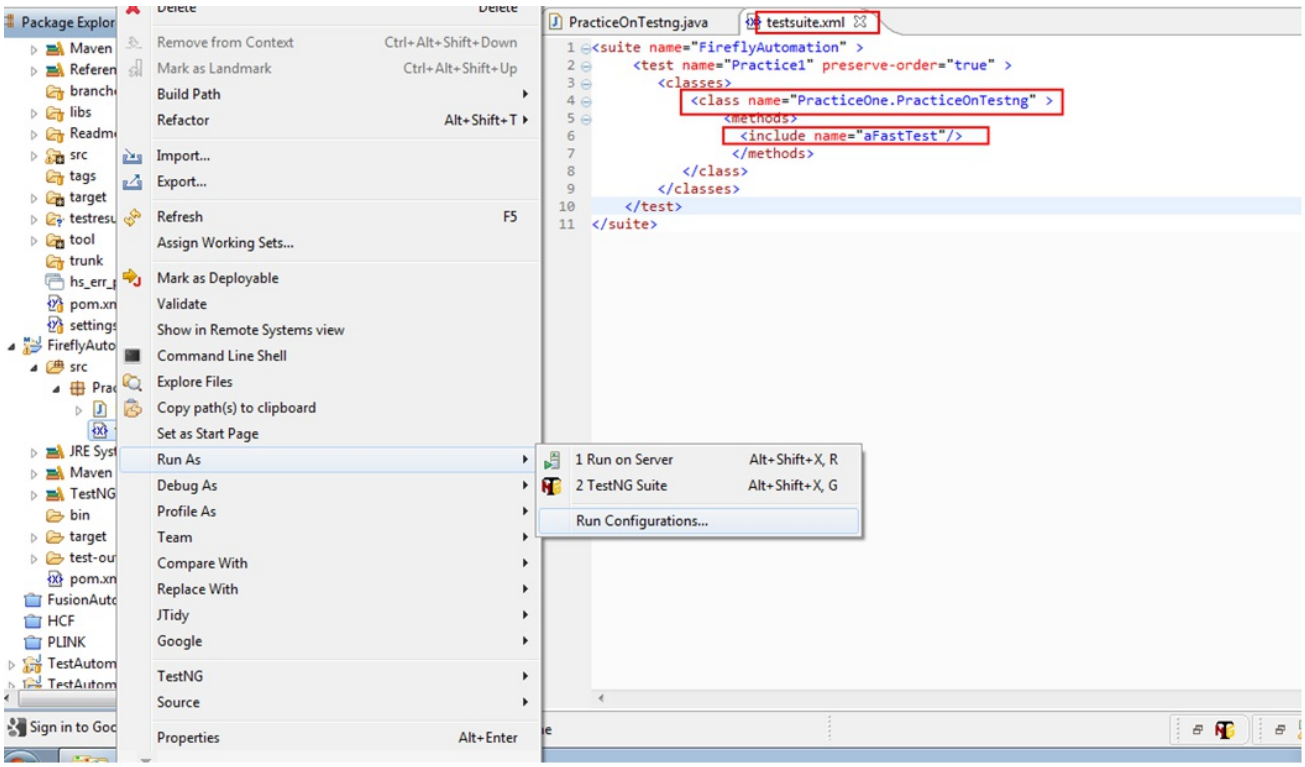
```java
package PracticeOne;

import org.testng.annotations.*;

public class PracticeOnTestng {
    @BeforeClass
    public void beforeClass(){
        System.out.println("beforeClass");
    }

    @BeforeTest
    public void beforeTest(){
        System.out.println("beforeTest");
    }

    @Test
    public void aFastTest(){
        System.out.println("aFastTest");
    }

    @Test
    public void bSlowTest(){
        System.out.println("bSlowTest");
    }


    @AfterTest
    public void afterTest(){
        System.out.println("afterTest");
    }

    @AfterClass
    public void afterClass(){
        System.out.println("afterClass");
    }
```

五）Testng运行方式CLass右击方式和xml方式：

如下图展开项目中具体类选定test的方法，右击即可运行。

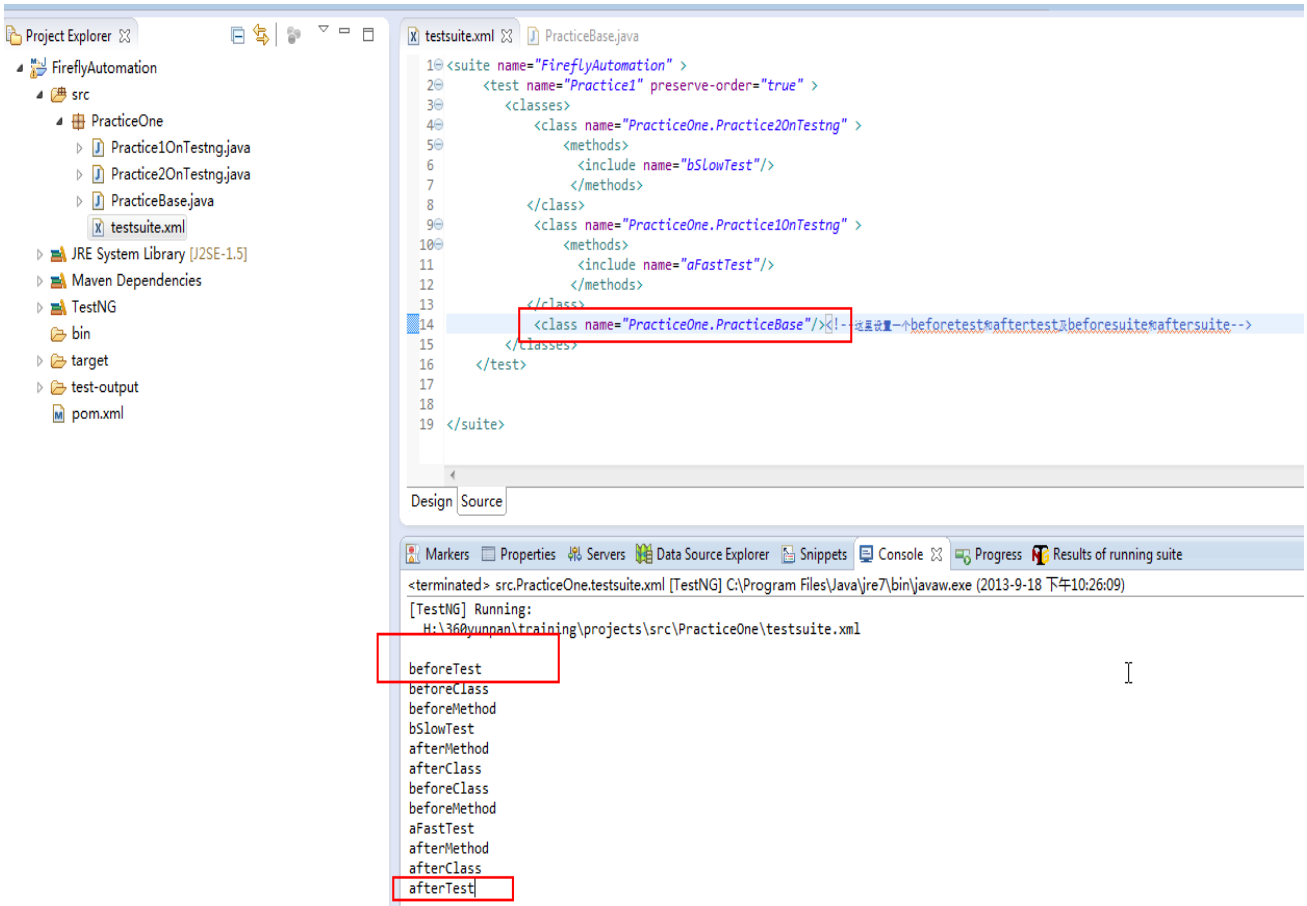


在该项目中新建一个tesuite.xml文件，按图中的要求即可



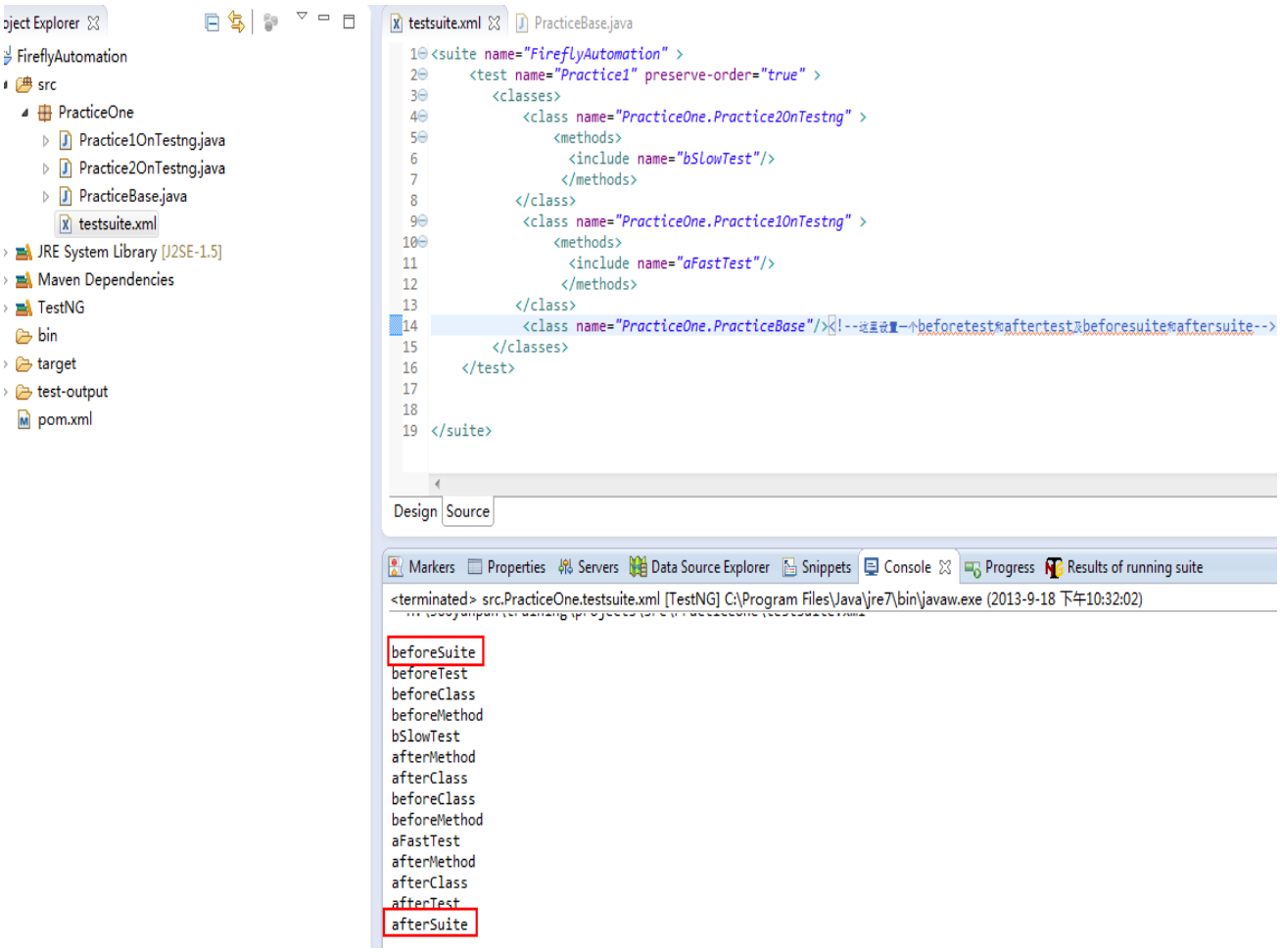
6）Testng的beforetest和aftertest

@Before/AfterTest和@Before/AfterTest：
xml文件中tag<Test>中包含的classes之前和之后需要运行的条件



@Before/AfterSuite和@Before/AfterSuite:
xml文件中tag<suite>中包含的classes之前和之后需要运行的条件

如果在PracticeBase添加beforesuite和aftersuite后看看运行结果：



总结：
testng的执行顺序

BeforeSuite>BeforeTest>BeforeClass>BeforeMethod>Test>AfterMethod>AfterClass>AfterTest>AfterSuite
@Test执行顺序不定。

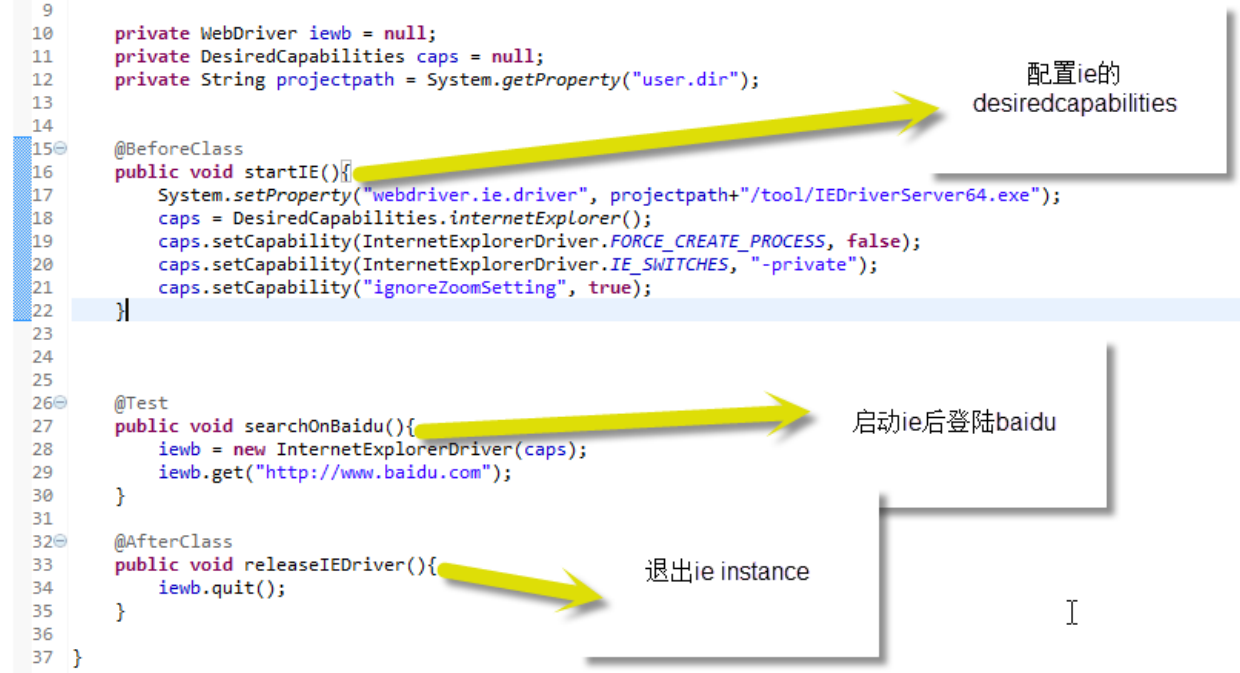
7）Webdriver 启动火狐，IE,和chrome

相关文档：https://code.google.com/p/selenium/wiki/{FirefoxDriver,ChromeDriver,InternetExplorerDriver}

启动ie需要有iedriver,下载地址：
https://code.google.com/p/selenium/downloads/list

LaunchIE.java

```java
package practicetwo;

import org.openqa.selenium.WebDriver;

public class LaunchIE {

    private WebDriver iewb = null;
    private DesiredCapabilities caps = null;
    private String projectpath = System.getProperty("user.dir");


    @BeforeClass
    public void startIE(){
        System.setProperty("webdriver.ie.driver", projectpath+"/tool/IEDriverServer64.exe");
        caps = DesiredCapabilities.internetExplorer();
        caps.setCapability(InternetExplorerDriver.FORCE_CREATE_PROCESS, false);
        caps.setCapability(InternetExplorerDriver.IE_SWITCHES, "-private");
        caps.setCapability("ignoreZoomSetting", true);
    }


    @Test
    public void searchOnBaidu(){
        iewb = new InternetExplorerDriver(caps);
        iewb.get("http://www.baidu.com");
    }

    @AfterClass
    public void releaseIEDriver(){
        iewb.quit();
    }

}
```



IE启动注意点：

①如我们运行以上的脚本出现这样的问题：
org.openqa.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)

我们还需要设置一下：

```
caps.setCapability(InternetExplorerDriver.INTRODUCE_FLAKINESS_BY_IGNORING_SECURITY_DOMAINS, true);
```

②ie怎样承认不信任网址我们需要添加一下两行:

```
iewb.get(url);
iewb.get("javascript:document.getElementById('overridelink').click();");
```

③Ie代理设置：

```
import org.snipecode.reg.RegUtil;
int handle =
RegUtil.RegOpenKey(RegUtil.HKEY_CURRENT_USER,"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Internet Settings",RegUtil.KEY_ALL_ACCESS)[RegUtil.NATIVE_HANDLE];
 RegUtil.RegSetValueEx(handle, "ProxyEnable", "dword:00000000");   //if need to set proxy ,=>dword:00000001
 RegUtil.RegSetValueEx(handle, "ProxyServer", PROXY);
```

启动chrome需要有chromedriver，下载地址：
http://chromedriver.storage.googleapis.com/index.html

Example usage:

```
ChromeOptions options = new ChromeOptions()
options.addExtensions(new File("/path/to/extension.crx"))
options.setBinary(new File("/path/to/chrome"));

// For use with ChromeDriver:
ChromeDriver driver = new ChromeDriver(options);

// or alternatively:
DesiredCapabilities capabilities = DesiredCapabilities.chrome();
capabilities.setCapability(ChromeOptions.CAPABILITY, options);
ChromeDriver driver = new ChromeDriver(capabilities);

// For use with RemoteWebDriver:
DesiredCapabilities capabilities = DesiredCapabilities.chrome();
capabilities.setCapability(ChromeOptions.CAPABILITY, options);
RemoteWebDriver driver = new RemoteWebDriver(
    new URL("http://localhost:4444/wd/hub"), capabilities);
```

比较一下以后chrome的启动方式与ie的差异

LaunchChrome.java

```
1  package practicetwo;
2
3  import java.util.Arrays;
4
5  import org.openqa.selenium.WebDriver;
6  import org.openqa.selenium.chrome.ChromeDriver;
7  import org.openqa.selenium.chrome.ChromeDriverService;
8  import org.openqa.selenium.ie.InternetExplorerDriver;
9  import org.openqa.selenium.remote.DesiredCapabilities;
10 import org.testng.annotations.*;
11
12 public class LaunchChrome {
13
14     private WebDriver chromewb = null;
15     private DesiredCapabilities caps = null;
16     private String projectpath = System.getProperty("user.dir");
17
18
19     @BeforeClass
20     public void startIE(){
21         System.setProperty("webdriver.chrome.driver", projectpath+"/tool/chromedriver.exe");
22         caps = DesiredCapabilities.chrome();
23         caps.setCapability("chrome.switches",Arrays.asList("--start-maximized"));  //最大化browser
24         //capabilities.setCapability("chrome.switches", Arrays.asList("--proxy-server=http://your-proxy-domain:4443")); //设置代理
25
26     }
27
28
29
30     @Test
31     public void searchOnBaidu(){
32         chromewb = new ChromeDriver(caps);
33         chromewb.get("http://www.baidu.com");
34     }
35
36     @AfterClass
37     public void releaseChromeDriver(){
38         chromewb.quit();
39     }
40
41 }
```

启动FF无需额外的driver驱动文件也就是你看到的那个exe文件：

以后主要的操作都在firefox运行，那么在webdriver启动ff后，怎么同时让此instance安装有firebug和firepath以便在运行时进行xpath或css锁定元素呢？

1)先下载firebug.xpi和firepath

https://getfirebug.com/downloads/

https://code.google.com/p/firepath/

2)设置FirefoxProfile

注意要是firefox不是默认安装需要设置

```
System.setProperty("webdriver.firefox.bin","D:\\Program Files\\Mozilla Firefox\\firefox.exe");
```

```
private WebDriver ffwb = null;
private FirefoxProfile firefoxprofile = null;
private String projectpath = System.getProperty("user.dir");


@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); 安装firebug,firepath
        firefoxprofile.addExtension(firepath);
        firefoxprofile.setPreference("webdriver.accept.untrusted.certs", "true"); 绕过证书问题
        firefoxprofile.setPreference("extensions.firebug.currentVersion", "1.12.1");
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

```
@Test
public void searchOnBaidu(){
    ffwb = new FirefoxDriver(firefoxprofile);
    ffwb.get("https://login.providerlink.healthcare.stg.covisint.com/providerlink");
}

@AfterClass
public void releaseFFDriver(){
    ffwb.quit();
}
```

在调试上段脚本时候按F12，你就会看到你的firebug也是有的。

Firefox 的代理设置：
profile.setPreference('network.proxy.type', 0) //不需要代理

profile.setPreference('network.proxy.type', 1)

那个端口号3128是整数
profile.setPreference('network.proxy.http', 'ip')
profile.setPreference('network.proxy.http_port', 3128)

8)Testng groups

有一种场景可能需要将一系类的测试脚分类,如platform是:windows,linux,unix的或者browser types是：ie,firefox,chrome的。

我们可以使用testng的groups来实现一个主模块下的一系列子模块：

```xml
<suite name="FireflyAutomation" >
    <test name="P3" preserve-order="true" >
        <groups>
            <define name="submodule1">
            </define>

            <define name="submodule2">
            </define>

            <define name="submodule3">
            </define>

            <define name="module1">
                <include name="submodule1"/>
                <include name="submodule2"/>
            </define>

             <define name="module2">
                <include name="submodule1"/>
                <include name="submodule3"/>
            </define>

            <run>
               <include name="module1"/>
            </run>

        </groups>

        <classes>
            <class name="practiceone.Practice3OnTestng" />
        </classes>
    </test>


</suite>
```

Terry:可以看到module1和2相对应调用的子模块。

```java
public class Practice3OnTestng {


    @Test(groups={"submodule1"})
    public void testSubModule1(){
        System.out.println("---testsubmodule1---");
    }

    @Test(groups={"submodule2"})
    public void testSubModule2(){
        System.out.println("---testsubmodule2---");
    }

    @Test(groups={"submodule3"})
    public void testSubModule3(){
        System.out.println("---testsubmodule3---");
    }
}
```

我们运行一下该xml文件后
所得结果：
---testsubmodule1---
---testsubmodule2---

一般的设置我们会把ie,ff,chrome放入beforesuite中进行初始化，这样可以在以后的@Test中使用，为了可任意选择我们预定义的browsers我也可以将其归入groups中。
我们将上面的程序稍加改动
xml文件中加入两处如下模块。

```
<suite name="FireflyAutomation" >
    <test name="P3" preserve-order="true" >
        <groups>
            <define name="submodule1">
            </define>

            <define name="submodule2">
            </define>

            <define name="submodule3" >
            </define>
            <define name="module1">
                <include name="submodule1"/>
                <include name="submodule2"/>
            </define>

             <define name="module2">
                <include name="submodule1"/>
                <include name="submodule3"/>
            </define>

            <run>
               <include name="module1"/>
            </run>
        </groups>

        <classes>
            <class name="practiceone.Practice30nTestng" />
        </classes>
    </test>


</suite>
```

```
<define name="ff"/>
<define name="ie"/>
<define name="chrome"/>
```

```
<include name="ff"/>
```

```
@BeforeSuite(groups={"ff"})
public void beforesuite_runonFF(){
    System.out.println("---Run On FF---");
}

@BeforeSuite(groups={"ie"})
public void beforesuite_runonIE(){
    System.out.println("---Run On IE---");
}

@Test(groups={"submodule1"})
public void testSubModule1(){
    System.out.println("---testsubmodule1---");
}

@Test(groups={"submodule2"})
public void testSubModule2(){
    System.out.println("---testsubmodule2---");
}

@Test(groups={"submodule3","ff"})
public void testSubModule3(){
    System.out.println("---testsubmodule3---");
}
```

我们运行一下该xml文件后
所得结果：

---Run On FF---
---testsubmodule1---
---testsubmodule2---
---testsubmodule3---

9)和Terry老师一起实践
要将我们今天学习的testng流程管理和webdriver的启动browser用上

要求：
BeforeClass中初始化Webdriver和Firefox的profile。

AfterClass中quit打开的Webdriver

我们有两个模块需要测试也就是说有两个<test>
两个模块需要都要登陆126.com FireflyAutomation@126.com/Firefly

模块一：查看收件箱
模块二：查看发件箱