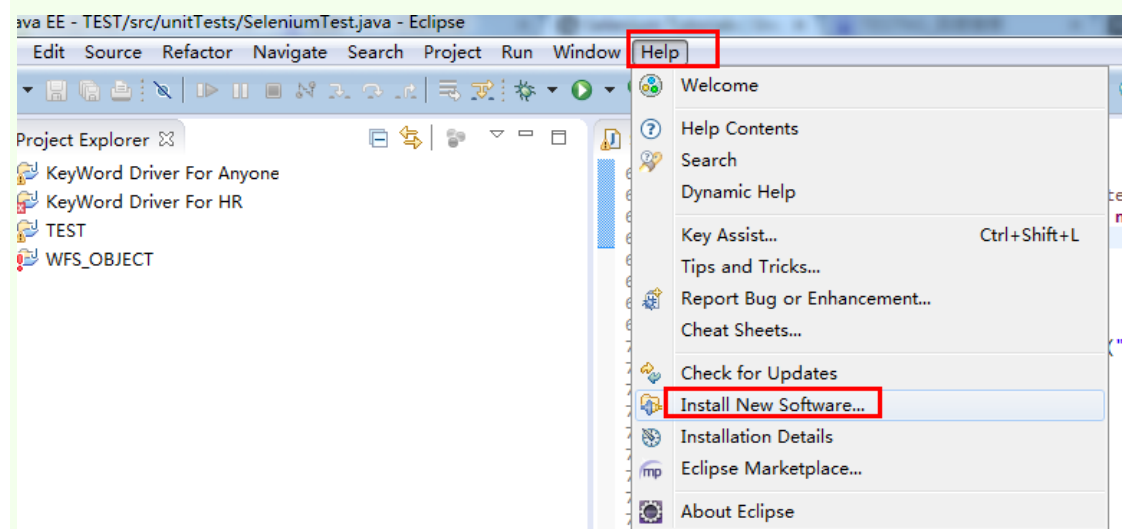


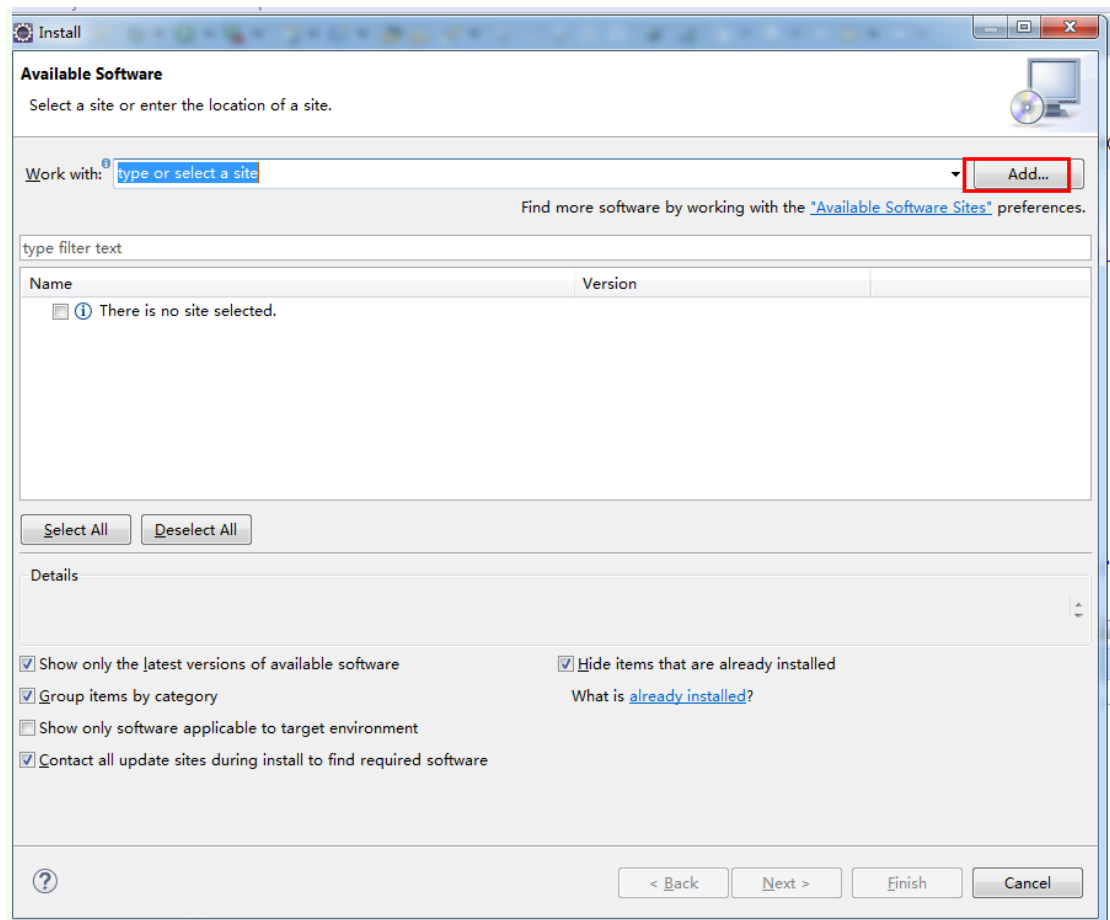
TestNG 图解说明

TestNG 是什么：

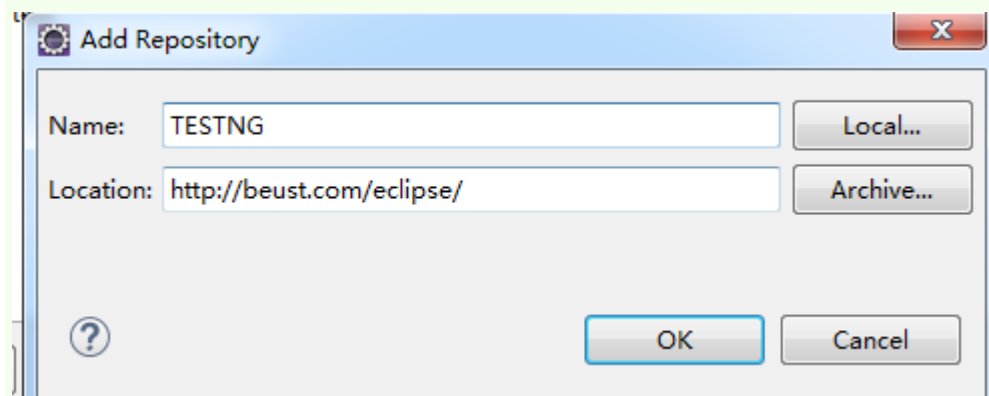
TestNG，即Testing, Next Generation，下一代测试技术，是一套根据JUnit和JUnit思想而构建的利用注释来强化测试功能的一个测试框架，即可以用来做单元测试，也可以用来做集成测试。

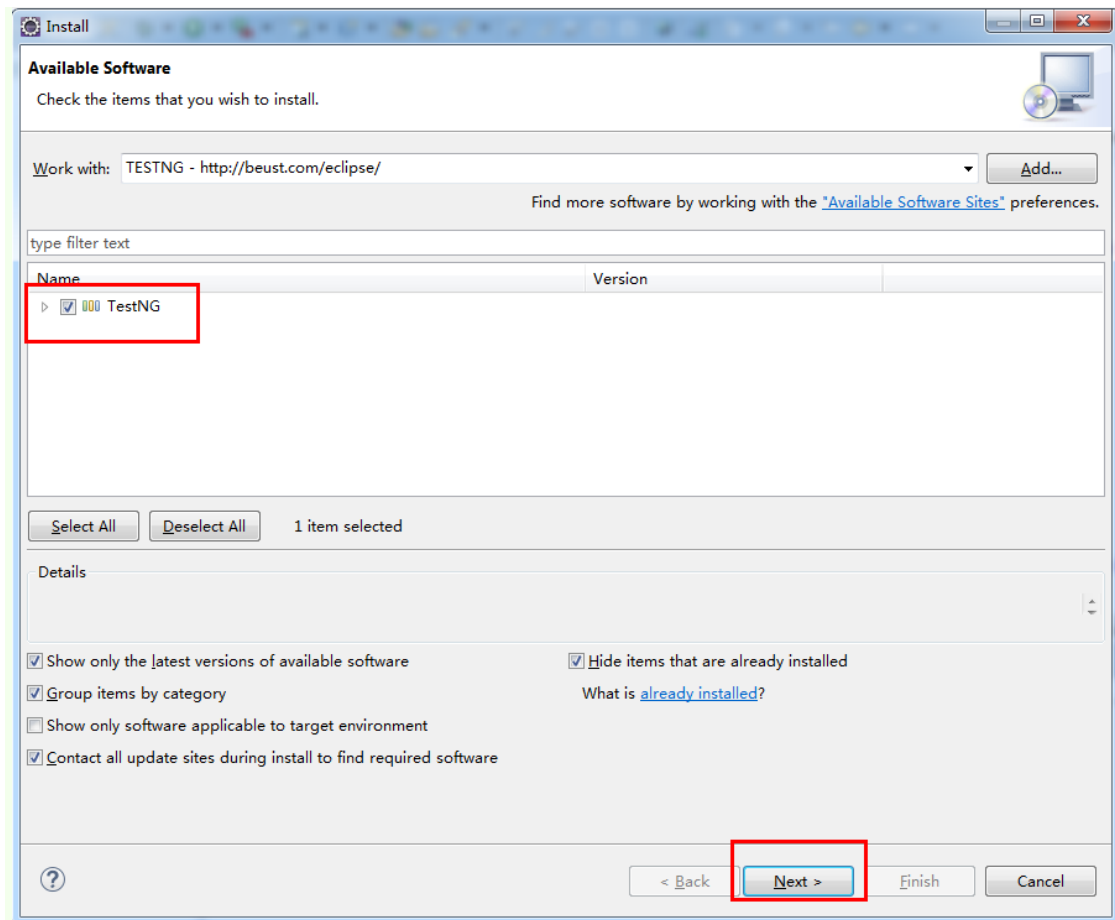
TestNG 如何安装

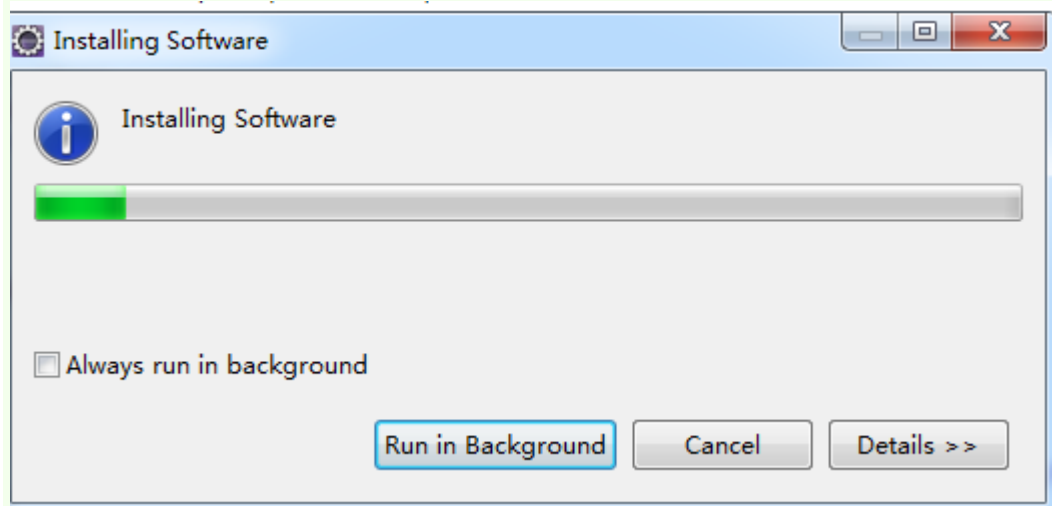
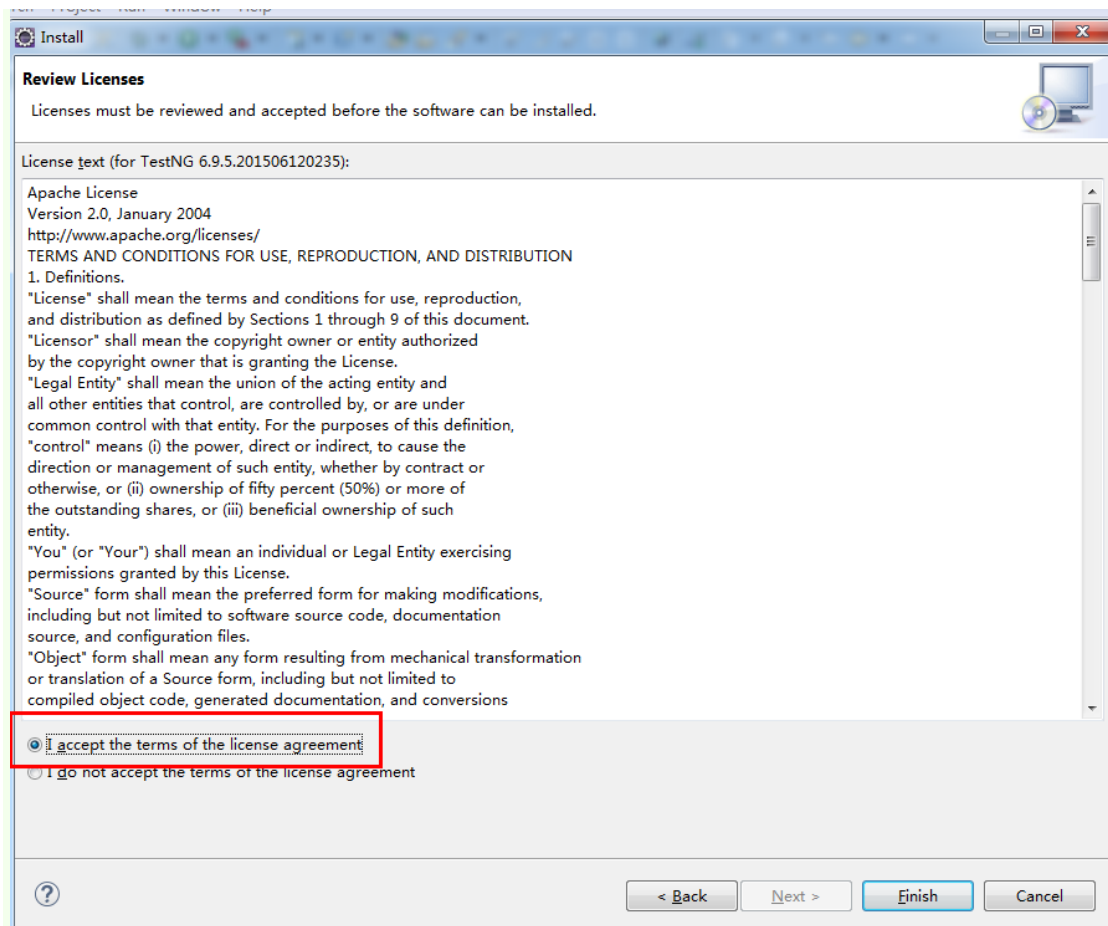


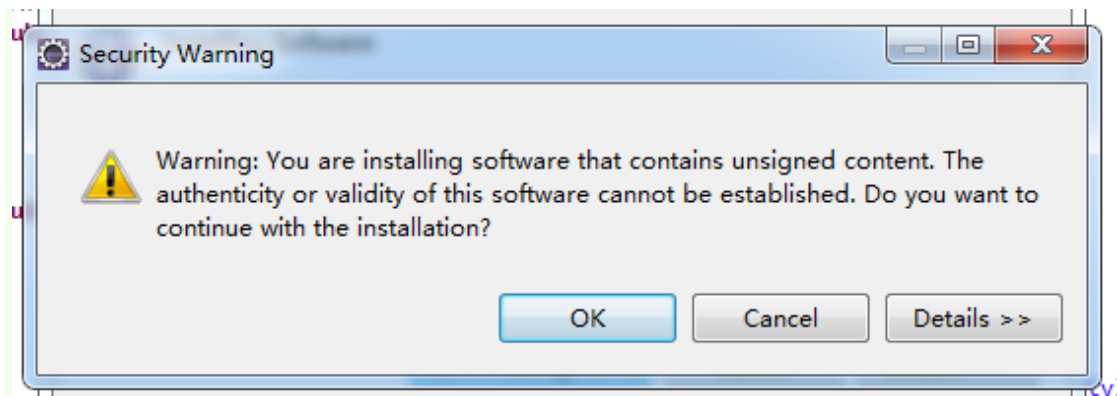


输入 <http://beust.com/eclipse/>

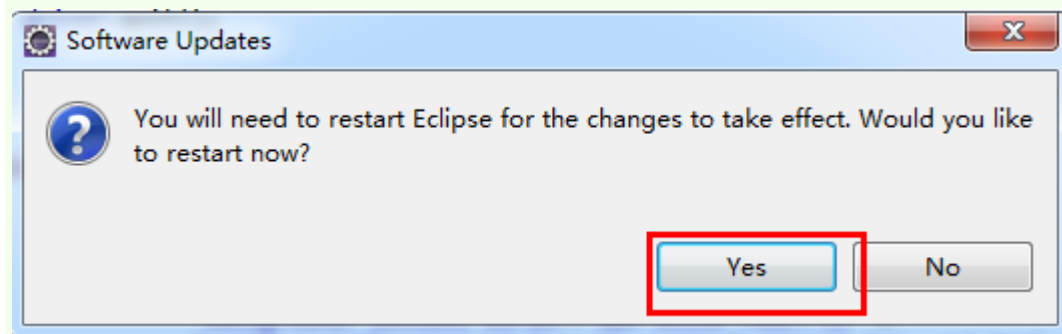








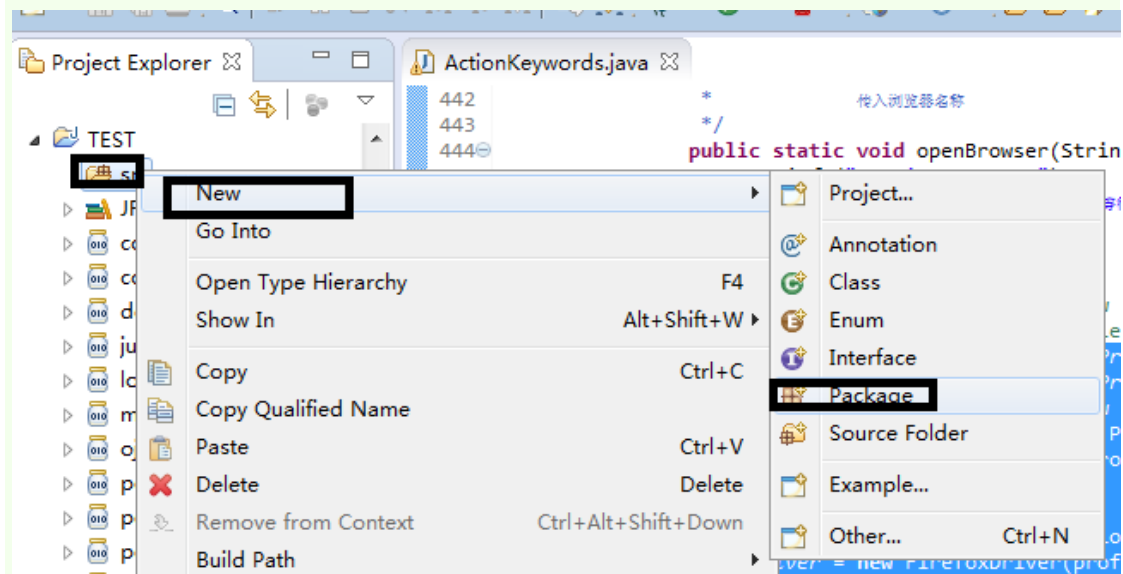
如果出现就点ok

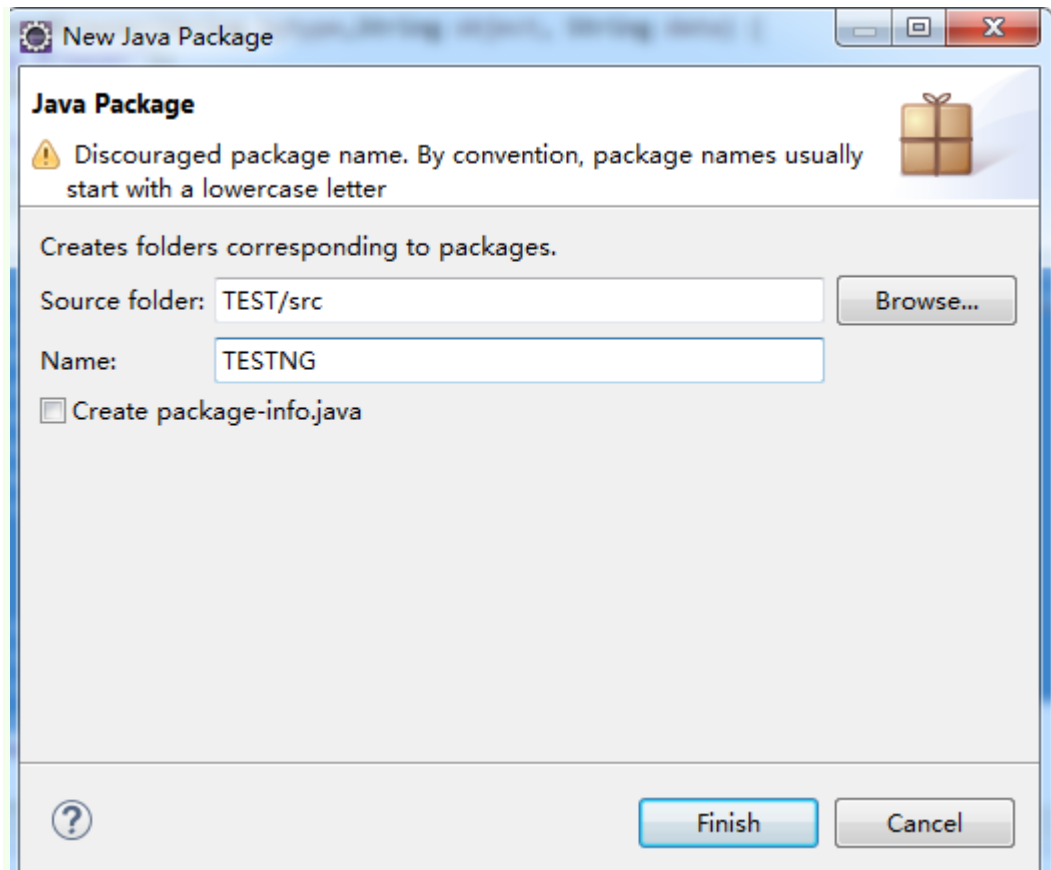


点yes

TESTNG 类新建

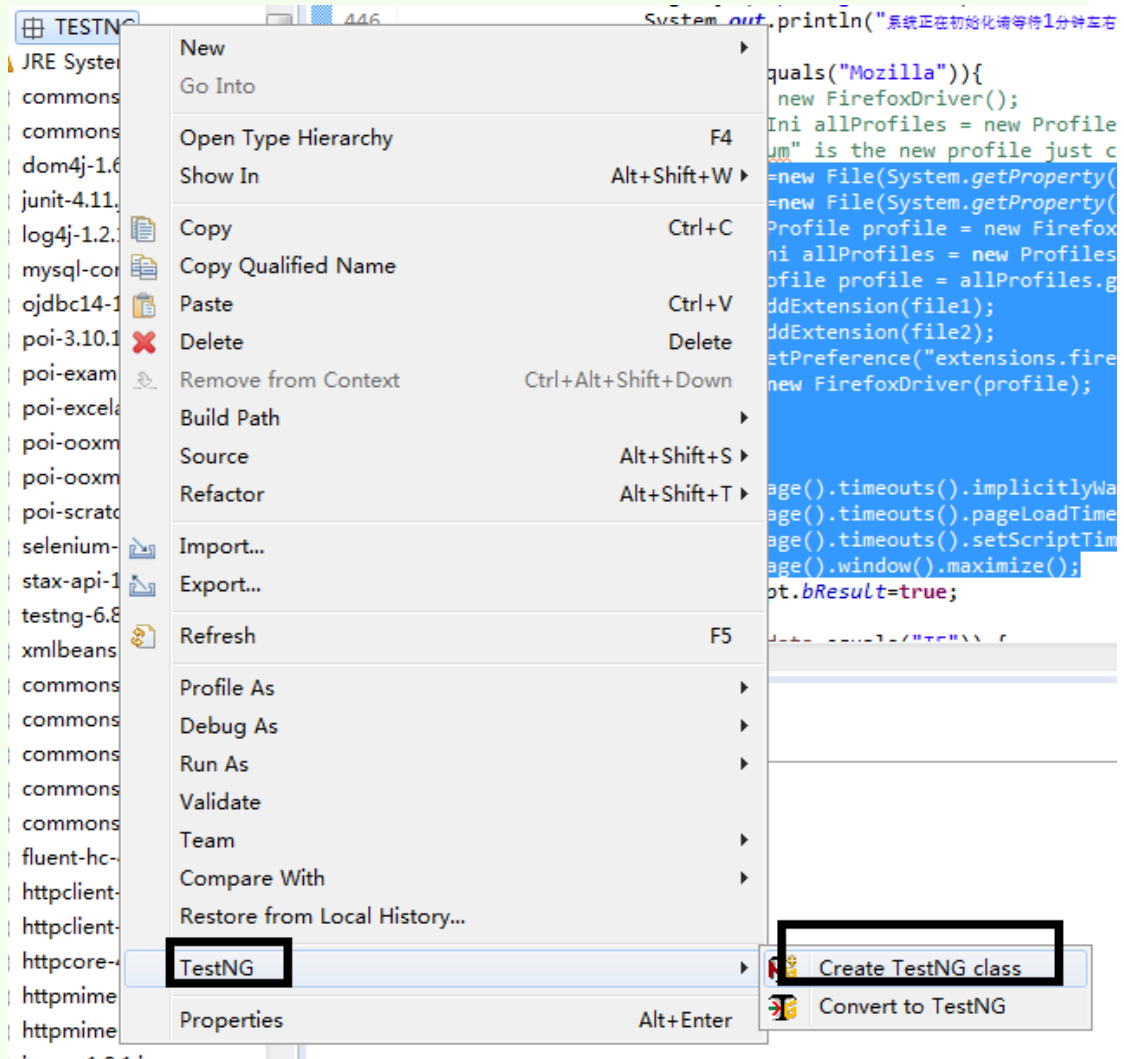
重启之后我们在测试工程中新建一个TESTNG 的包名

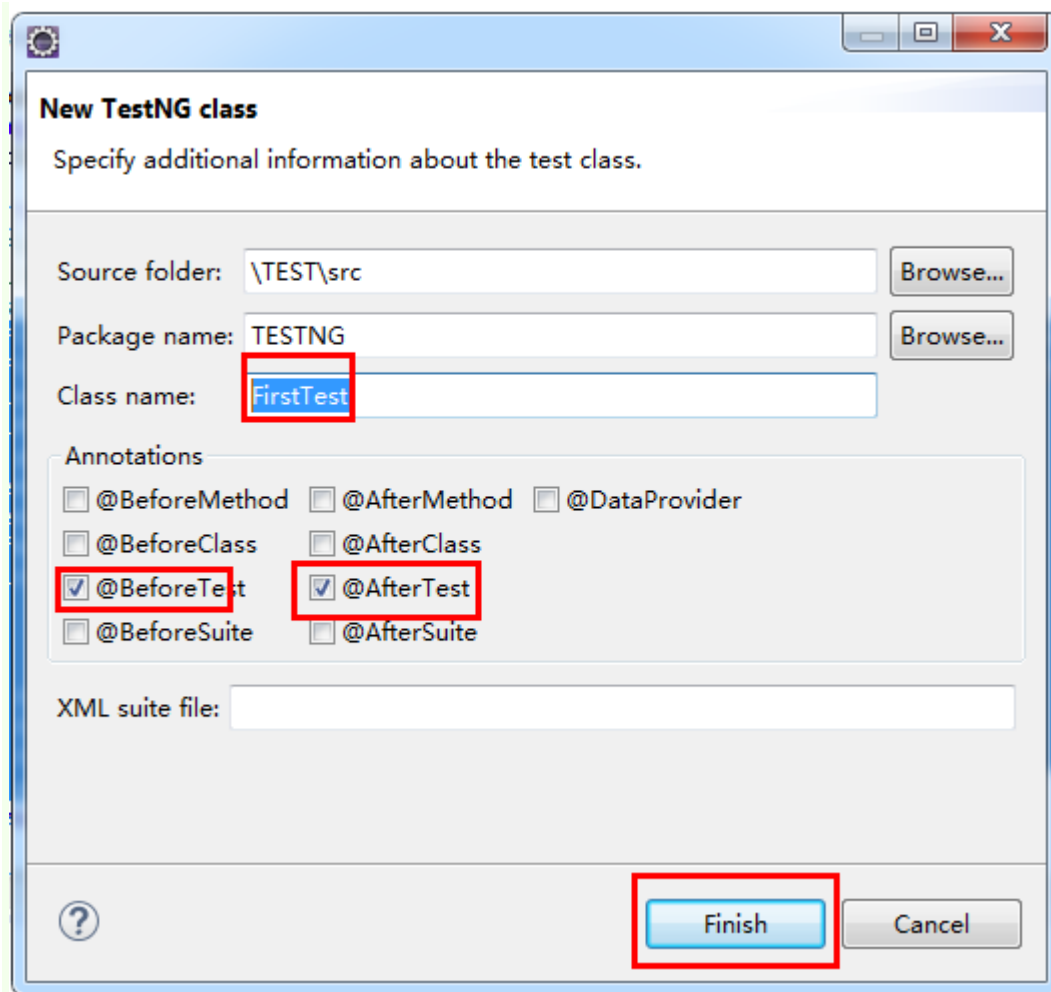




在TESTNG 包名处右击鼠标

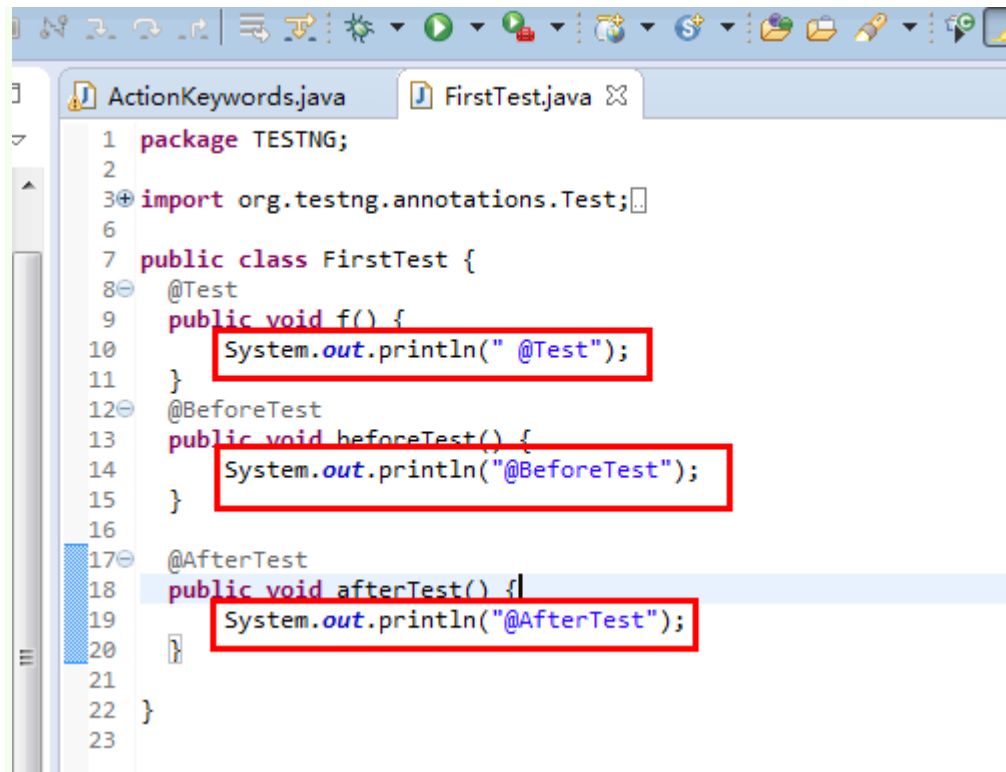
TESTNG——Create Testng





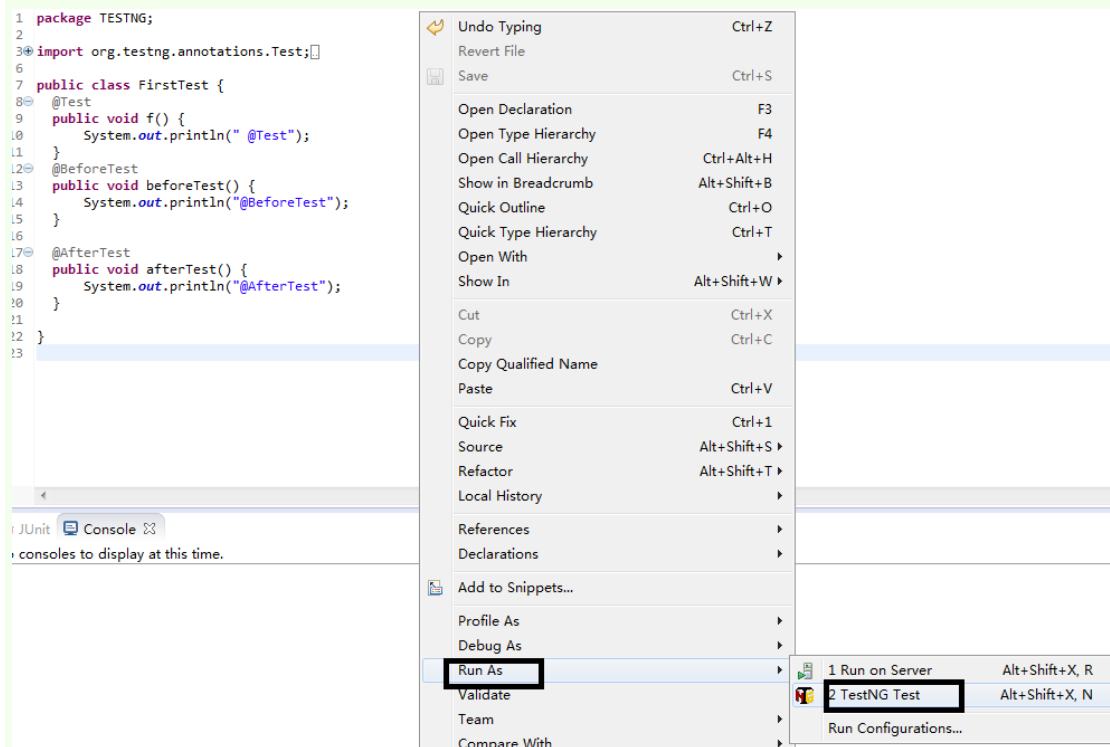
```
1 package TESTNG;
2
3 import org.testng.annotations.Test;
4
5
6
7 public class FirstTest {
8     @Test
9     public void f() {
10    }
11    @BeforeTest
12    public void beforeTest() {
13    }
14
15    @AfterTest
16    public void afterTest() {
17    }
18
19 }
20
```

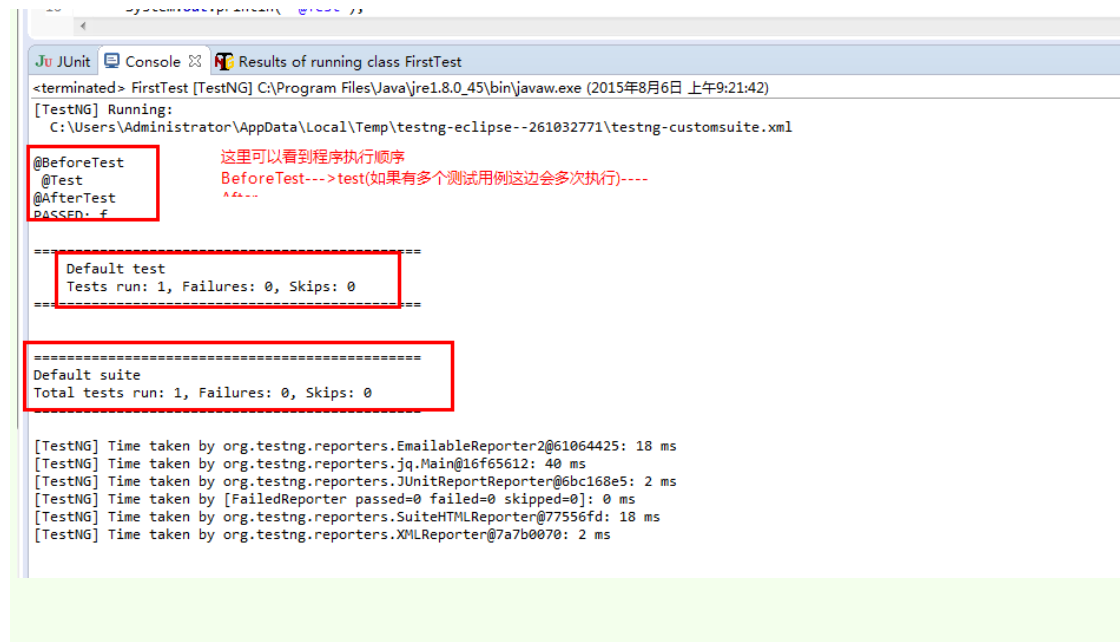
现在增加部分代码



```
1 package TESTING;
2
3 import org.testng.annotations.Test;
4
5
6
7 public class FirstTest {
8     @Test
9     public void f() {
10         System.out.println(" @Test");
11     }
12     @BeforeTest
13     public void beforeTest() {
14         System.out.println("@BeforeTest");
15     }
16
17     @AfterTest
18     public void afterTest() {
19         System.out.println("@AfterTest");
20     }
21 }
22
23
```

选择Run——TESTNG





```
<terminated> FirstTest [TestNG] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015年8月6日 上午9:21:42)
[TestNG] Running:
  C:\Users\Administrator\AppData\Local\Temp\testng-eclipse--261032771\testng-customsuite.xml

@BeforeTest
@Test
@AfterTest
PASSED: f

====
Default test
Tests run: 1, Failures: 0, Skips: 0
====

=====
Default suite
Total tests run: 1, Failures: 0, Skips: 0
=====

[TestNG] Time taken by org.testng.reporters.EmailableReporter2@61064425: 18 ms
[TestNG] Time taken by org.testng.reporters.jq.Main@16f65612: 40 ms
[TestNG] Time taken by org.testng.reporters.JUnitReportReporter@6bc168e5: 2 ms
[TestNG] Time taken by [FailedReporter passed=0 failed=0 skipped=0]: 0 ms
[TestNG] Time taken by org.testng.reporters.SuiteHTMLReporter@77556fd: 18 ms
[TestNG] Time taken by org.testng.reporters.XMLReporter@7a7b0070: 2 ms
```

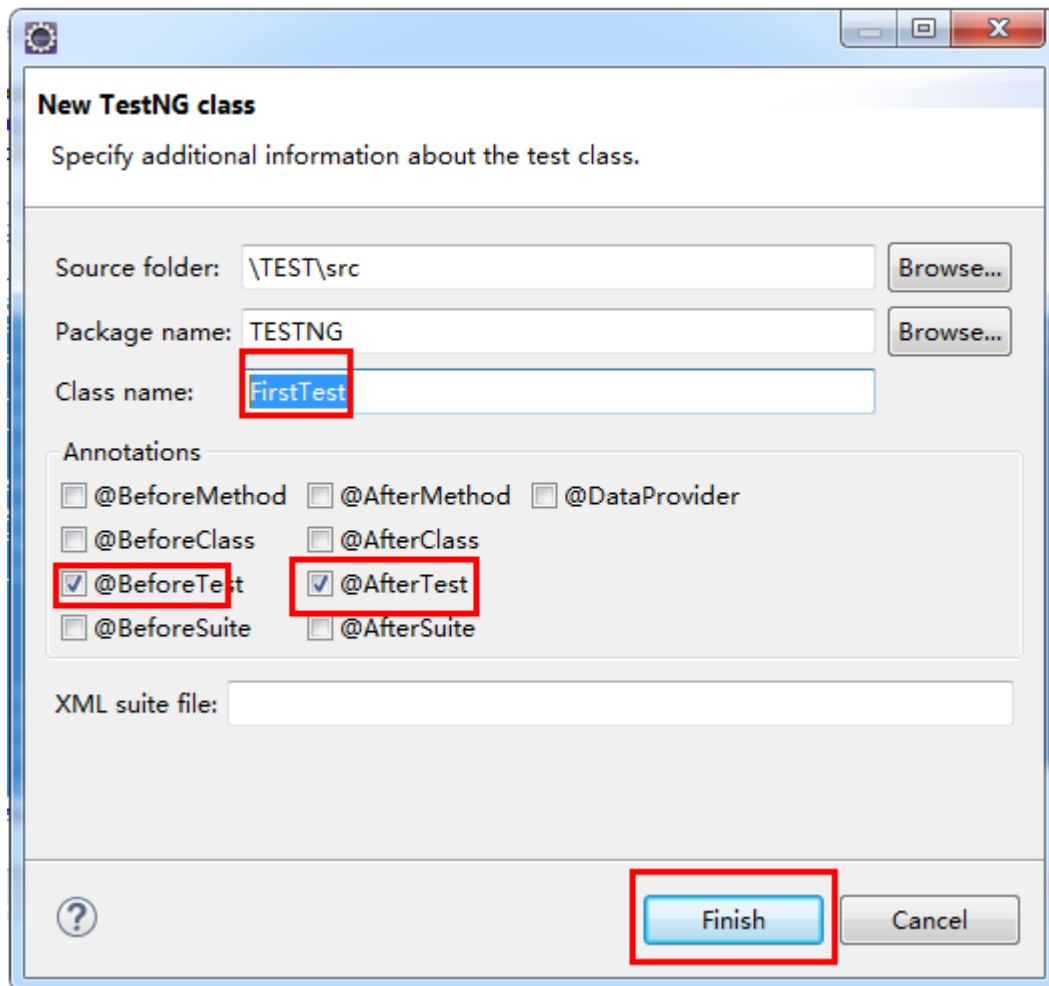
执行上面最简单的一个代码时大家一定发现了在每个方法前有一个@

标签, 这个标签在TESTNG中叫注解

现在跟大家普及下TESTNG 中的标签是什么意思, 执行顺序是怎样的呢

TESTNG 注解

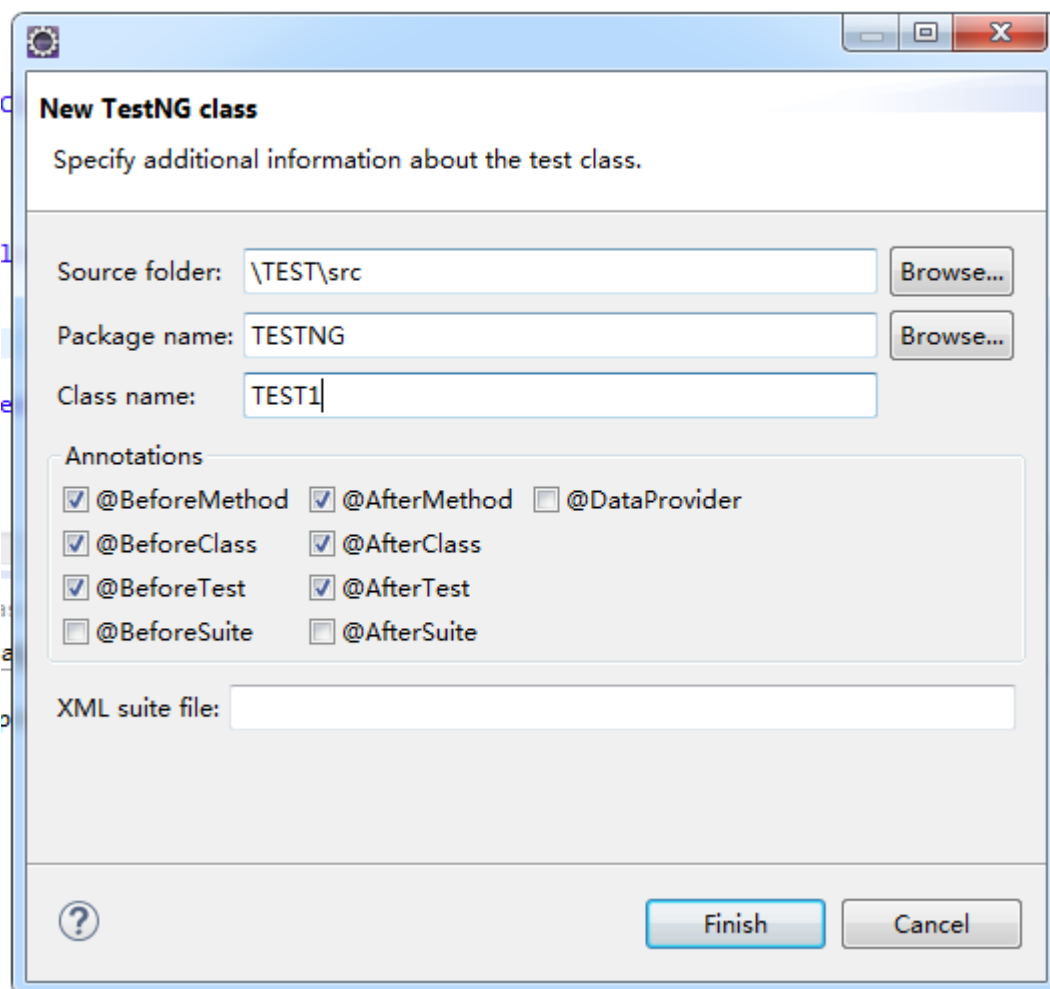
首先我们在创建TESTNG时出现以下界面对吧



以下是每个注解的具体描述信息

注解	描述
@Before Suite	注解的方法将只运行一次，运行所有测试前此套件中。
@After Suite	注解的方法将只运行一次此套件中的所有测试都运行之后。
@Before Class	注解的方法将只运行一次先行先试在当前类中的方法调用。
@After Class	注解的方法将只运行一次后已经运行在当前类中的所有测试方法。
@Before Test	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。
@After Test	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。
@Before Groups	组的列表，这种配置方法将之前运行。此方法是保证在运行属于任何这些组第一个测试方法，该方法被调用。
@After Groups	组的名单，这种配置方法后，将运行。此方法是保证运行后不久，最后的测试方法，该方法属于任何这些组被调用。
@Before Method	注解的方法将每个测试方法之前运行。
@After Method	被注释的方法将被运行后，每个测试方法。
@DataProvider	标志着一个方法，提供数据的一个测试方法。注解的方法必须返回一个 Object[] []，其中每个对象[]的测试方法的参数列表中可以分配。该@Test 方法，希望从这个DataProvider的接收数据，需要使用一个 dataProvider名称等于这个注解的名字。
@Factory	作为一个工厂，返回TestNG的测试类的对象将被用于标记的方法。该方法必须返回Object[]。
@Listeners	定义一个测试类的监听器。
@Parameters	介绍如何将参数传递给@Test方法。
@Test	标记一个类或方法作为测试的一部分。

为了便于了解上述注解的使用方式我这边在创建Class□时全选上所有注解



模板生成之后增加打印语句

```
13 public class TEST1 {
14     @Test
15     public void f() {
16         System.out.println("f0");
17     }
18
19     @Test
20     public void f1() {
21         System.out.println("f1");
22     }
23
24     @Test
25     public void f3() {
26         System.out.println("f3");
27     }
28
29     @BeforeMethod
30     public void beforeMethod() {
31         System.out.println(" @BeforeMethod");
32     }
33
34     @AfterMethod
35     public void afterMethod() {
36         System.out.println(" @AfterMethod");
37     }
38
39     @BeforeClass
40     public void beforeClass() {
41         System.out.println(" @BeforeClass");
42     }
43
44     @AfterClass
45     public void afterClass() {
46         System.out.println(" @AfterClass");
47     }
48
49     @BeforeTest
50     public void beforeTest() {
51         System.out.println(" beforeTest");
52     }
53
54 }
```

运行TESTNG

```
JUnit Console Results of running class TEST1
<terminated> TEST1 [TestNG] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015年8月6日 上午9:46:14)
[TestNG] Running:
C:\Users\Administrator\AppData\Local\Temp\testng-eclipse-167896546\testng-customsuite.xml

beforeTest
@BeforeClass
@BeforeMethod
f0
@AfterMethod
@BeforeMethod
f1
@AfterMethod
@BeforeMethod
f3
@AfterMethod
@AfterClass
@AfterTest
PASSED: f
PASSED: f1
PASSED: f3

=====
Default test
Tests run: 3 Failures: 0, Skips: 0
=====
Default suite
Total tests run: 3 Failures: 0, Skips: 0
=====

[TestNG] Time taken by org.testng.reporters.EmailableReporter2@61064425: 11 ms
[TestNG] Time taken by org.testng.reporters.jq.Main@16f65612: 26 ms
[TestNG] Time taken by org.testng.reporters.JUnitReportReporter@6bc168e5: 4 ms
[TestNG] Time taken by [FailedReporter passed=0 failed=0 skipped=0]: 0 ms
[TestNG] Time taken by org.testng.reporters.SuiteHTMLReporter@77556fd: 28 ms
[TestNG] Time taken by org.testng.reporters.XMLReporter@7a7b0070: 4 ms
```

其中的BeforeMethod/AfterMethod
会在每个方法中调用一次, 这种适用场所, 比如我执行完第一个用例之后要重置
数据才能执行第二条用例时, 可以用这种方式, BeforeMethod
就放重置数据的方法, 让每个方法执行之前都去调用这个方法

现在再用实际例子跑一次苏宁易购网
新建以下代码

```
6
7 import org.openqa.selenium.By;
8 import org.openqa.selenium.WebDriver;
9 import org.openqa.selenium.firefox.FirefoxDriver;
10 import org.testng.annotations.Test;
11 import org.testng.annotations.BeforeTest;
12 import org.testng.annotations.AfterTest;
13
14 public class Sun {
15     public WebDriver driver;
16     @Test
17     public void f() {
18         //输入网址
19         driver.get("http://www.suning.com/?utm_source=baidu&utm_medium=brand&utm_campaign=title");
20         //点登录
21         driver.findElement(By.xpath("//a[class='login']")).click();
22         //输入用户名
23         driver.findElement(By.id("userName")).sendKeys("13764142840");
24         //输入密码
25         driver.findElement(By.id("password")).sendKeys("ASUS!1234");
26         //点击登录
27         driver.findElement(By.id("submit")).click();
28         //获取用户信息文本
29         String TXT=driver.findElement(By.xpath("//span[@id='usernameHtml02']")).getText();
30         System.out.println("打印用户信息"+TXT);
31         //字符串断言：预期结果与实际结果比对
32         Assert.assertEquals("137*****40",TXT);
33     }
34     @BeforeTest
35     public void beforeTest() {
36         driver = new FirefoxDriver();
37         driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
38     }
39
40     @AfterTest
41     public void afterTest() {
42         driver.quit();
43     }
44
45 }
```

用TESTNG 运行该类

查看记录运行结果是PASS□的

<terminated> Sun [TestNG] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015年8月6日 上午10:17:53)

[TestNG] Running:

C:\Users\Administrator\AppData\Local\Temp\testng-eclipse--1481458524\testng-customsuite.xml

log4j:WARN No appenders could be found for logger (org.apache.http.client.protocol.RequestAddCookies).

log4j:WARN Please initialize the log4j system properly.

打印用户信息137*****40

八月 06, 2015 10:19:33 上午 org.openqa.selenium.os.UnixProcess\$SeleniumWatchDog destroyHarder

信息: Command failed to close cleanly. Destroying forcefully (v2). org.openqa.selenium.os.UnixProcess\$Seleni

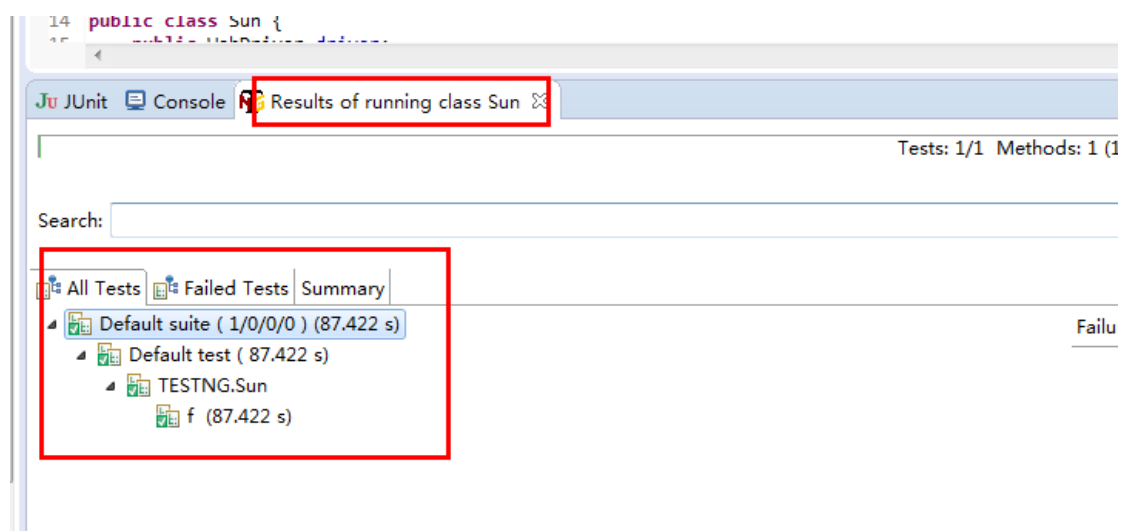
PASSED: f

Default test
Tests run: 1, Failures: 0, Skips: 0

Default suite
Total tests run: 1, Failures: 0, Skips: 0

[TestNG] Time taken by org.testng.reporters.EmailableReporter2@61064425: 4 ms
[TestNG] Time taken by org.testng.reporters.jq.Main@16f65612: 18 ms
[TestNG] Time taken by org.testng.reporters.JUnitReportReporter@6bc168e5: 2 ms
[TestNG] Time taken by [FailedReporter passed=0 failed=0 skipped=0]: 0 ms
[TestNG] Time taken by org.testng.reporters.SuiteHTMLReporter@77556fd: 18 ms
[TestNG] Time taken by org.testng.reporters.XMLReporter@7a7b0070: 3 ms

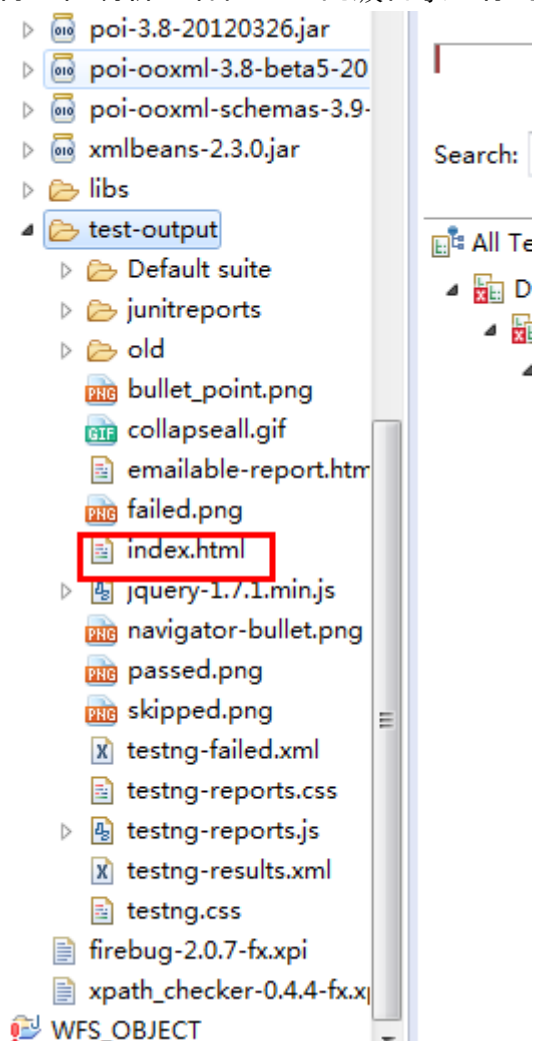
切换到另一视图

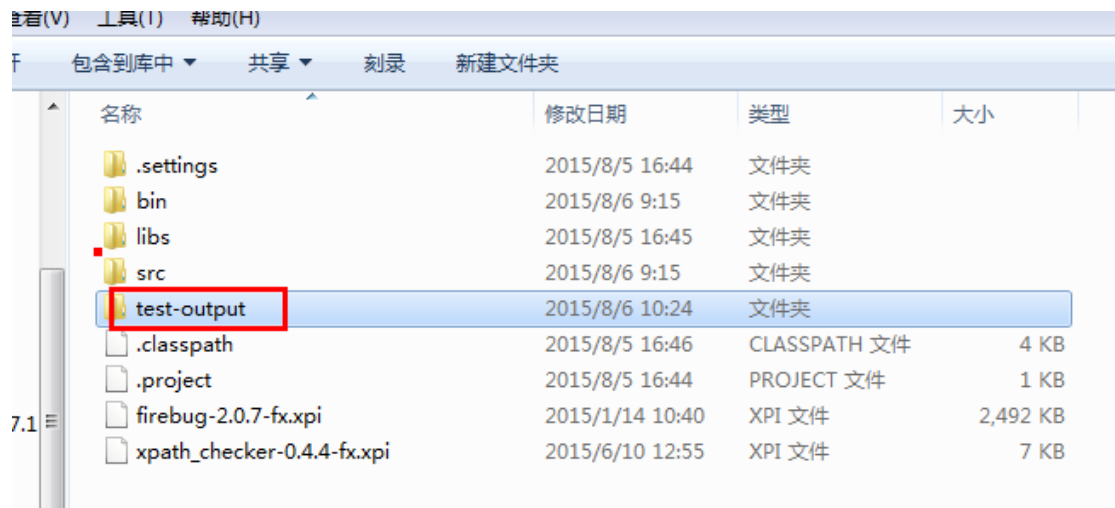


可以精确看到每个方法执行花费多长时间, 也可以看到执行是通过还是不通过

TESTNG 结果展示

将工程刷新之后在SRC 同级目录下你可以看到一个TEST-OUTPUT 的目录





打开该目录选择'emailable-report.html',

TestNG Report

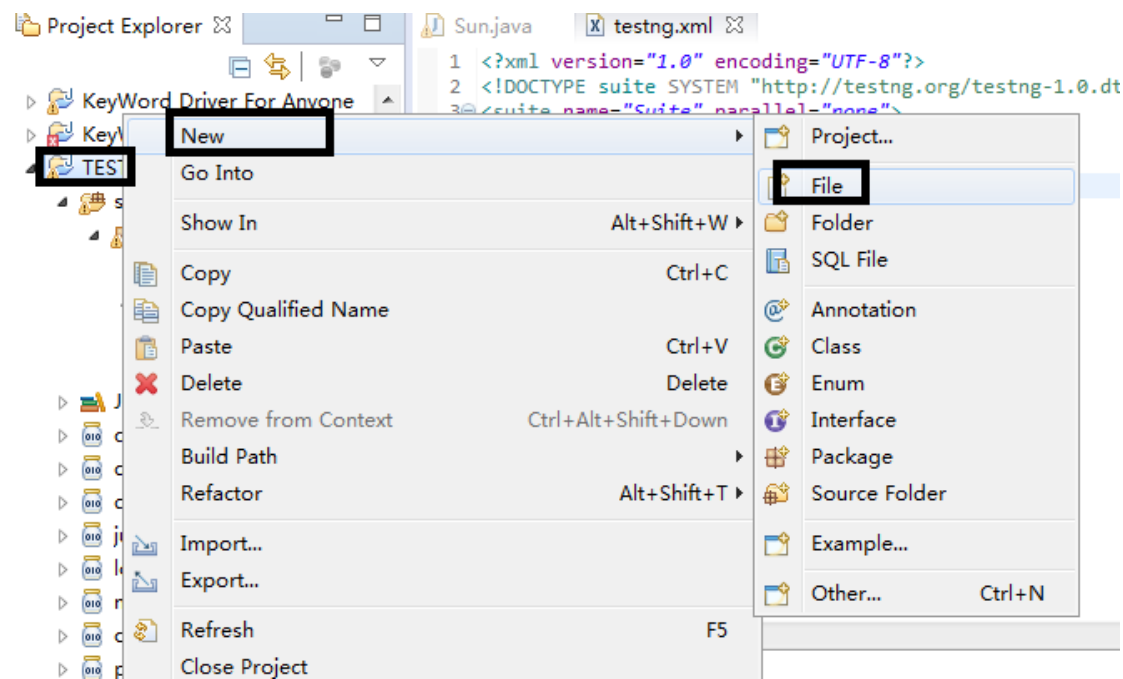
file:///E:/eclipse1/workspace/TEST/test-output/emailable-report.html

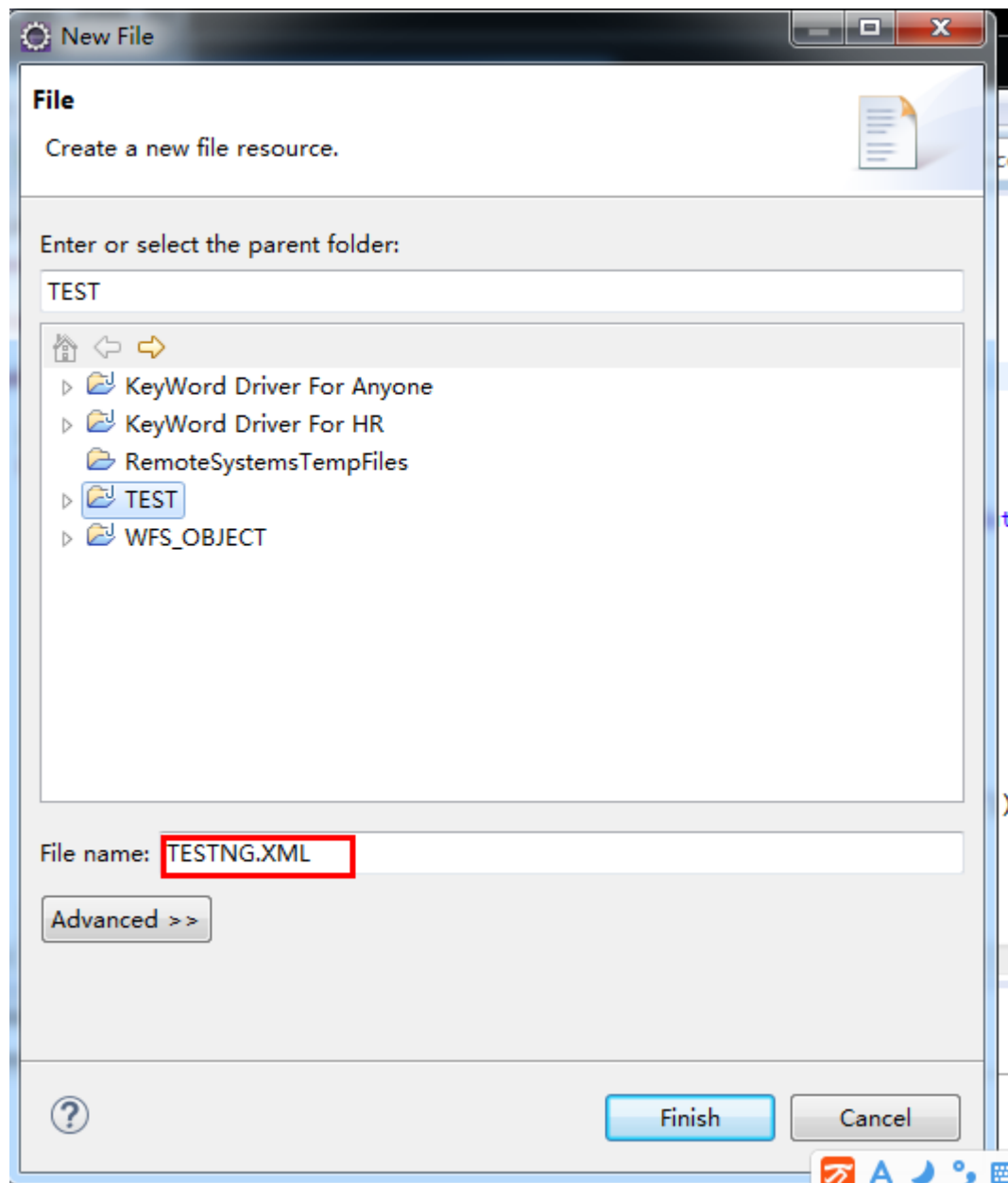
Test	# Passed	# Skipped	# Failed	Time (ms)	Included Groups	Excluded Groups
Default suite						
Default test	1	0	0	80,405		

Class	Method	Start	Time (ms)
Default suite			
Default test — passed			
TESTNG.Sun	f	1438828225029	66720

TESTNG 中XML 简单配置

现在对上述测试类稍作修改, 做成XML 调用方式

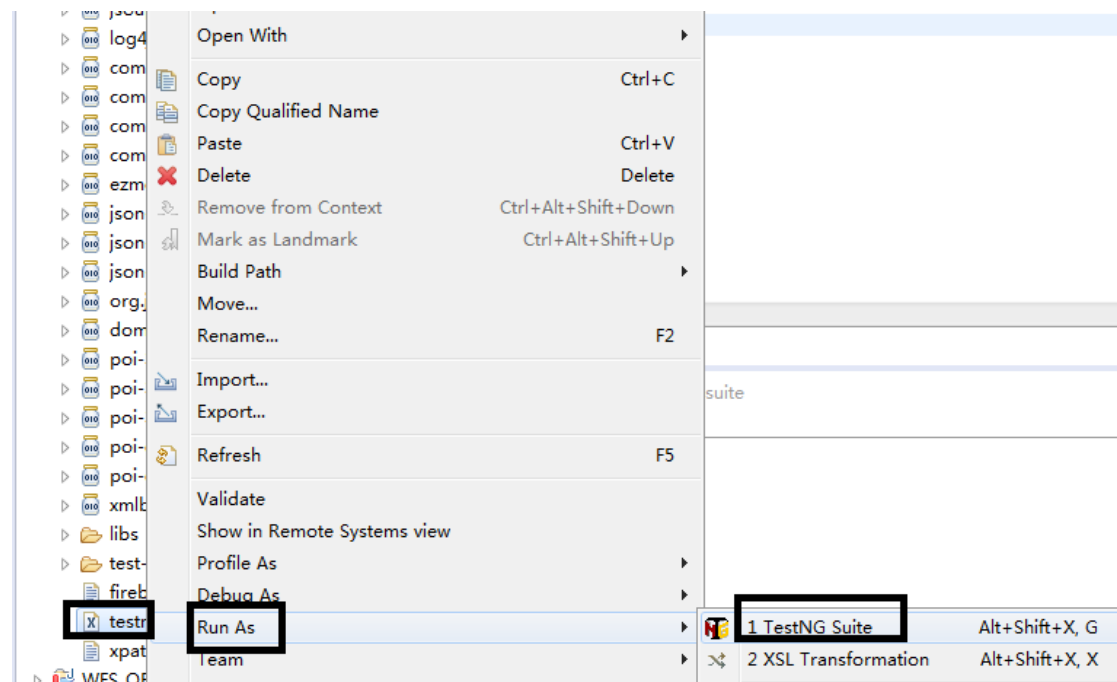




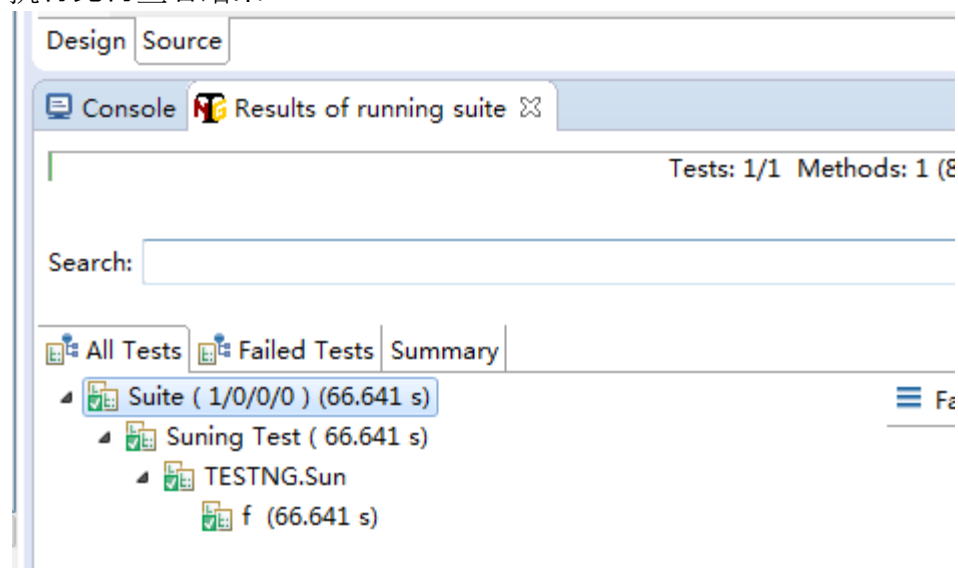
在TESTNG.XML 中输入以下代码

```
Sun.java testng.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
3 <suite name="Suite" parallel="none">
4   <test name="Suning Test">
5     <classes>
6       <class name="TESTNG.Sun"/>
7     </classes>
8   </test> <!-- Suning Test -->
9 </suite> <!-- Suite -->
10
```

选择TESTNG. XML 后点右键选择RUN -TESTNG TEST



执行完再查看结果



TESTNG 执行顺序

TESTNG 在实际执行过程中顺序是如何排的呢

其实他并不是按代码中从上往下执行的, 他是按字母 + 数字排序

新建以下类

```
Sun.java depends.java *MultipleTest.java
3 import org.testng.annotations.Test;
4
5 public class MultipleTest {
6
7     @Test
8     public void One() {
9         System.out.println("This is the Test Case number One");
10    }
11    @Test
12    public void Two() {
13        System.out.println("This is the Test Case number Two");
14    }
15
16    @Test
17
18    public void Three() {
19        System.out.println("This is the Test Case number Three");
20    }
21
22    @Test
23    public void Four() {
24        System.out.println("This is the Test Case number Four");
25    }
26 }
```

运行TESTNG□TEST

```
<terminated> MultipleTest [TestNG] C:\Program Files\Java\jre1.8.0
[TestNG] Running:
  C:\Users\Administrator\AppData\Local\Temp\testng-ecli

This is the Test Case number Four
This is the Test Case number One
This is the Test Case number Three
This is the Test Case number Two
PASSED: Four
PASSED: One
PASSED: Three
PASSED: Two
```

查看结果

发现他并没有按我们代码中1, 2, 3, 4□的顺序执行

那如果我想按代码写的顺序执行怎么办呢

可以在注解中增加顺序

(priority = XX) □XX□代表第几次执行

我们将上述代码再修改下, 先将1, 2, 3, 4 的顺序再打乱下

再用标记的方式让系统执行按1, 2, 3, 4 的顺序执行

```
1 package TESTNG;
2
3 import org.testng.annotations.Test;
4
5 public class MultipleTest {
6
7     @Test(priority = 1)
8     public void One() {
9         System.out.println("This is the Test Case number One");
10    }
11
12
13    @Test (priority = 3)
14
15    public void Three() {
16        System.out.println("This is the Test Case number Three");
17    }
18
19    @Test (priority = 2)
20    public void Two() {
21        System.out.println("This is the Test Case number Two");
22    }
23
24    @Test (priority = 4)
25    public void Four() {
26        System.out.println("This is the Test Case number Four");
27    }
28 }
```

Console Results of running class MultipleTest

terminated> MultipleTest [TestNG] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015年8月6日 上午11:26:32)

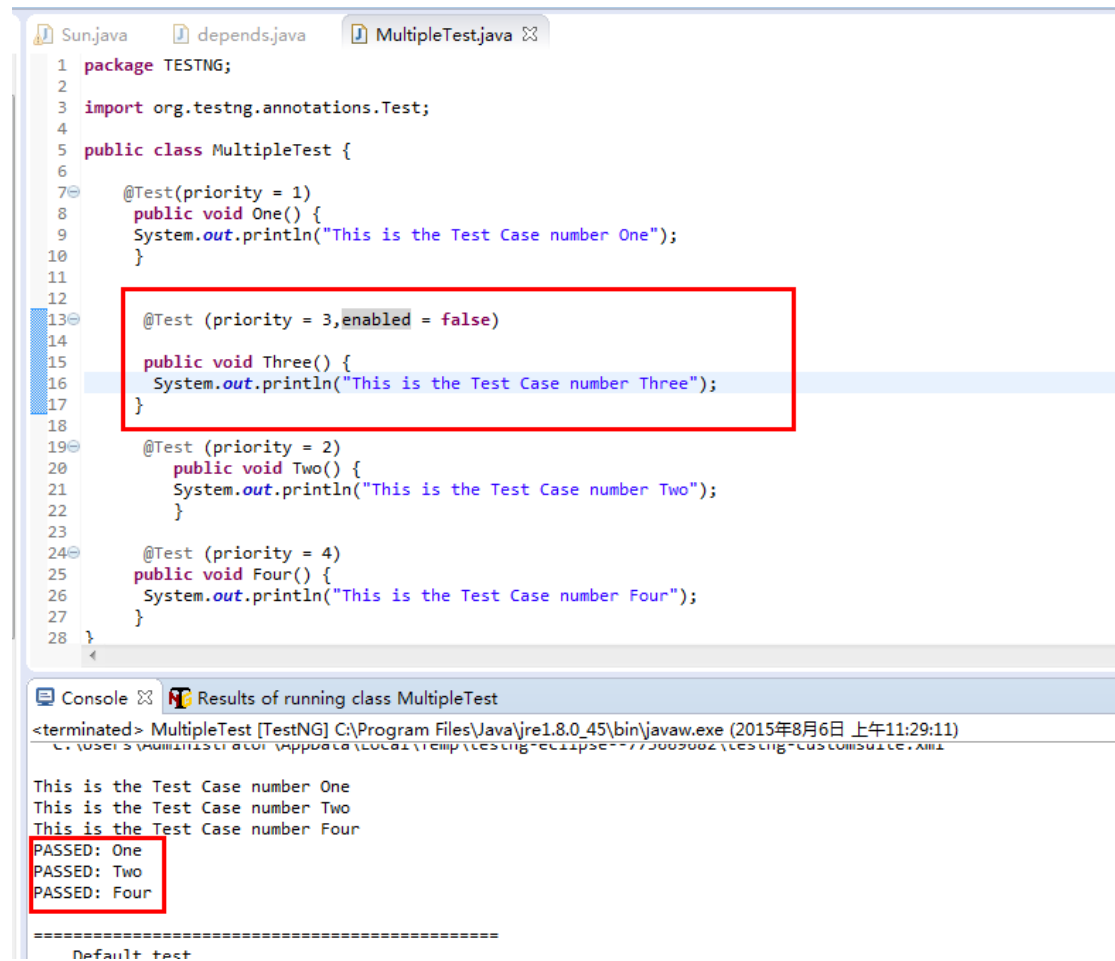
TestNG] Running:
C:\Users\Administrator\AppData\Local\Temp\testng-eclipse-1473265539\testng-customsuite.xml

'his is the Test Case number One
'his is the Test Case number Two
'his is the Test Case number Three
'his is the Test Case number Four
'ASSED: One
'ASSED: Two
'ASSED: Three
'ASSED: Four

假如调试时我可能只要执行部分方法, 也就是说要跳过一部分方法怎么办呢, 同样是在注解中增加参数

这时我们可以增加 `enabled = false` 表示该方法不执行

在执行过程中3的方法就没有执行了



The screenshot shows an IDE with a Java file named `MultipleTest.java` and a console window. The code defines a class `MultipleTest` with four test methods: `One`, `Three`, `Two`, and `Four`. The `Three` method is annotated with `@Test (priority = 3, enabled = false)`. The console output shows the execution of the tests, with the `Three` test being skipped due to being disabled. The output is as follows:

```
1 package TESTNG;
2
3 import org.testng.annotations.Test;
4
5 public class MultipleTest {
6
7     @Test(priority = 1)
8     public void One() {
9         System.out.println("This is the Test Case number One");
10    }
11
12
13    @Test (priority = 3,enabled = false)
14
15    public void Three() {
16        System.out.println("This is the Test Case number Three");
17    }
18
19    @Test (priority = 2)
20    public void Two() {
21        System.out.println("This is the Test Case number Two");
22    }
23
24    @Test (priority = 4)
25    public void Four() {
26        System.out.println("This is the Test Case number Four");
27    }
28 }
```

Console Output:

```
<terminated> MultipleTest [TestNG] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015年8月6日 上午11:29:11)
C:\Users\Administrator\AppData\Local\Temp\testng-eclipse-773009002\testng-customsuite.xml

This is the Test Case number One
This is the Test Case number Two
This is the Test Case number Four
PASSED: One
PASSED: Two
PASSED: Four

=====
Default test
```

TESTNG Resport 记录展示

在RESTPORT 中展示Log

再新增一个 `ReporterLogs` TESTNG类

输入以下内容


```
18
19 private static Logger Log = Logger.getLogger(Log.class.getName());
20
21 @BeforeTest
22 public void beforeTest() {
23     DOMConfigurator.configure("log4j.xml");
24     driver = new FirefoxDriver();
25     driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
26     Reporter.Log("浏览器已打开");
27 }
28
29 @Test
30 public void f() {
31     //输入网址
32     driver.get("http://www.suning.com/?utm_source=baidu&utm_medium=brand&utm_campaign=title");
33     Log.info("网址已输入");
34
35     //点登录
36     driver.findElement(By.xpath("//a[@class='login']")).click();
37     Log.info("点击登录");
38     //输入用户名
39     driver.findElement(By.id("userName")).sendKeys("13764142840");
40     Log.info("用户名输入");
41     //输入密码
42     driver.findElement(By.id("password")).sendKeys("ASUS!1234");
43     Log.info("密码输入");
44     //点击上登录
45     driver.findElement(By.id("submit")).click();
46     Log.info("点击马上登录");
47     //获取用户信息文本
48     String TXT=driver.findElement(By.xpath("//span[@id='usernameHtml02']")).getText();
49     System.out.println("打印用户信息"+TXT);
50     //字符串断言, 预期结果与实际结果对比
51     Assert.assertEquals("137*****40", TXT);
52     Log.info("信息验证Ok");
53
54     Reporter.Log("测试方法已验证通过");
55 }
56
57 }
```

执行之后打开 emailable-report.html 会发现如图所示的画面

Test	# Passed	# Skipped	# Failed	Time (ms)	Included Groups	Excluded Groups
Default suite						
Default test	1	0	0	75,452		

Class	Method	Start	Time (ms)
Default suite			
Default test - passed			
TESTNG.ReporterLogs	f	1438832825390	66428

Messages
测试方法已验证通过

Default test

TESTNG.ReporterLogs#f

Messages
测试方法已验证通过

Log.info("信息验证Ok");

Reporter.Log("测试方法已验证通过");

可以看出两个log 展示的地方不一样, Reporter.log

只是在结果中展示, 用于展示一些概述性的东西时比较好, 比如一个方法执行通过就打印下, 具体步骤我们就用log.info 方式记录

TESTNG 参数化

TESTNG 中的参数与数据提供者介绍

参数需要在注解中加入这个格式

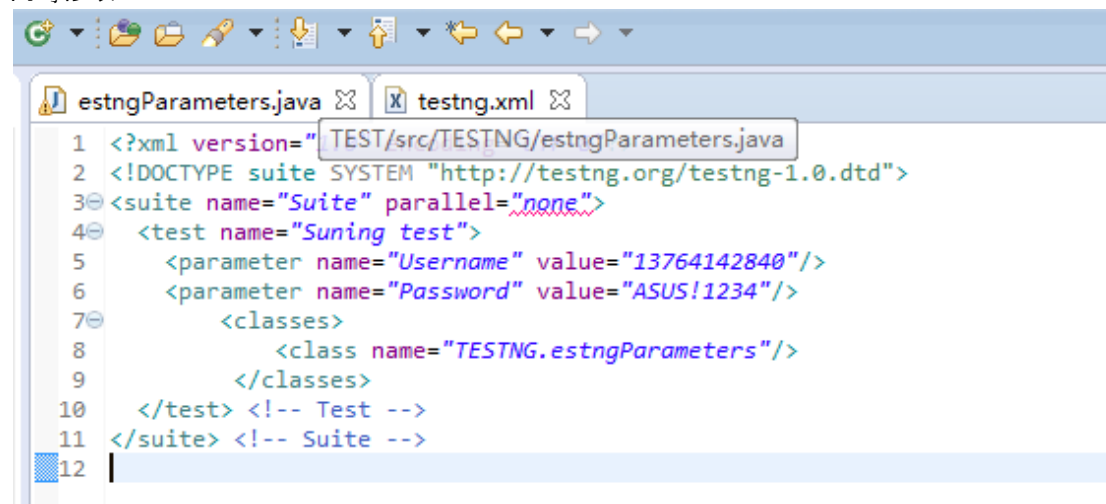
```
@Parameters({ "参数1", "参数2" })
```

现在我们还是以苏宁易购网站为例稍作修改

新建一个测试类

```
19
20 private static Logger Log = Logger.getLogger(Log.class.getName());
21
22 @BeforeTest
23 public void beforeTest() {
24     DOMConfigurator.configure("log4j.xml");
25     driver = new FirefoxDriver();
26     driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
27     Reporter.Log("浏览器已打开");
28 }
29
30 @Test
31 @Parameters({ "Username", "Password" })
32 public void f(String Username, String Password) {
33     // 输入网址
34     driver.get("http://www.suning.com/?utm_source=baidu&utm_medium=brand&utm_campaign=title");
35     Log.info("网址已输入");
36
37     // 点登录
38     driver.findElement(By.xpath("//a[@class='login']")).click();
39     Log.info("点击登录");
40     // 输入用户名
41     driver.findElement(By.id("userName")).sendKeys(Username);
42     Log.info("用户名输入");
43     // 输入密码
44     driver.findElement(By.id("password")).sendKeys>Password);
45     Log.info("密码输入");
46     // 点击登录
47     driver.findElement(By.id("submit")).click();
48     Log.info("点击马上登录");
49     // 获取用户信息文本
50     String TXT=driver.findElement(By.xpath("//span[@id='usernameHtml02']")).getText();
51     System.out.println("打印用户信息"+TXT);
52     // 字符串断言，预期结果与实际结果对比
53     Assert.assertEquals("137*****40", TXT);
54     Log.info("信息验证Ok");
55
56     Reporter.Log("测试方法已验证通过");
57 }
58
59 }
```

同时修改TESTNG.XML



这样设置代表数据是从xml 中取, 之后传级测试类中的参数

点击TESTNG.XML 运行testng 之后结果展示:

Console

Results of running suite

Search:

All Tests

Failed Tests

Summary

Suite (1/0/0/0) (66.71 s)

Suning test (66.71 s)

TESTNG.estngParameters

f (66.71 s)

"13764142840","ASUS!1234" (66.71 s)

火狐主页

TestNG Report

+

file:///E:/eclipse1/workspace/TEST/test-output/emailable-report.html

Test	# Passed	# Skipped	# Failed	Time (ms)	Included Groups
Suite					
Suning test	1	0	0	74,591	

Class	Method	Start	Time (ms)
Suite			
Suning test — passed			
TESTNG. estngParameters	f	1438835041433	66710

Suning test

TESTNG. estngParameters#f

Parameter #1	Parameter #2
13764142840	ASUS!1234

Messages
测试方法已验证通过

另一种数据提供者
也是通过增加注解方式实现
标准格式是:

```
@DataProvider(name = "XXXX")
```

现再新建一个测试类
输入以下代码

```
@DataProvider(name = "Data")

public static Object[][] credentials() {

    return new Object[][] { { "13764142840", "ASUS!1234" }, { "13764142841", "ASUS!1234" } };

}

@BeforeTest
public void beforeTest() {
    DOMConfigurator.configure("log4j.xml");
    driver = new FirefoxDriver();
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    Reporter.log("浏览器已打开");
}

@Test(dataProvider = "Data")
public void f(String Username, String Password) {
    System.out.println("用户名与密码"+Username+" "+Password);
    //输入网址
    driver.get("http://www.suning.com/?utm_source=baidu&utm_medium=brand&utm_campaign=title");
    Log.info("网址已输入");

    //点登录
    driver.findElement(By.xpath("//a[@class='login']")).click();
    Log.info("点击登录");
    //输入用户名
    driver.findElement(By.id("userName")).sendKeys(Username);
    Log.info("用户名输入");
    //输入密码
    driver.findElement(By.id("password")).sendKeys>Password);
    Log.info("密码输入");
    //点击马上登录
    driver.findElement(By.id("submit")).click();
    Log.info("点击马上登录");
    //获取用户信息文本
    String TXT=driver.findElement(By.xpath("//span[@id='usernameHtml02']")).getText();
    System.out.println("打印用户信息"+TXT);
    //字符串断言, 预期结果与实际结果比对
    AssertJUnit.assertEquals("137*****40", TXT);
    Log.info("测试成功");
}
```

运行TESTNG

The screenshot shows the IDE interface with the code editor and the TestNG console. The code is the same as above. The console shows the test results. Two red boxes highlight the data provider values: '13764142840', 'ASUS!1234' and '13764142841', 'ASUS!1234'. Arrows point from these boxes to the test results in the console. The console shows two test runs, both passing. The first run is for '13764142840', 'ASUS!1234' and the second run is for '13764142841', 'ASUS!1234'.

我们虽然只有一个测试方法, 但同一个测试方法有两组数据, 所以被执行两次

理财产品介绍

公司官网: <https://www.epaybank.com/>





首先, 公司背景强大, 资金雄厚、平台安全, 可信!

注册资金2.08亿,

全国50多家分公司

管理团队均有5年以上行业高管经验

其次, 产品安全安全性极高, 收益稳定, 投资门槛低!

壹定盈——百万级信托理财产品!

第三方回购, 100%保本付息!

年化收益高达13%, 500元即可投资!

安全稳定, 零风险!

年化率: 13%, 有需要的同仁加我 QQ:1334862845



理财信息确认

产品名称: 壹定盈-C150623004期

投资期限: 6 个月

购买金额: 10000元

到期结算日: 2016-01-15

预期收益: 650.00元

推荐人手机号码

没有可不填

开始投资

购买时填写手机号码 137 6414 2840