

# 单元测试框架-TestNg

讲师：芳姐

金斧子

| 5 金斧子五周年  
ANNIVERSARY

科 技 提 升 投 资 品 质

## 一、什么是单元测试

单元测试 ( unit Testing ) 是指在计算机编程中，针对程序模块（软件设计的最小单位）来进行正确性校验的测试工作。

单元测试的特点如下：

- 程序单元是应用的最小可测试部件，通常采用基于类或者类的方法进行测试。
- 程序单元和其它单元是相互独立的
- 单元测试的执行速度很快
- 单元测试发现的问题，相对容易定位
- 单元测试通常由开发人员来完成
- 通过了解代码的实现逻辑进行测试，通常称之为白盒测试

## 二、TestNg单元测试框架

Testng单元测试框架比JUnit单元测试框架更强大，它提供了更多的扩展功能。目前，很大一部分自动化测试工程师已经开始转向使用Testng单元测试框架来运行更复杂的自动化测试用例。

### 2.1 什么是Testng

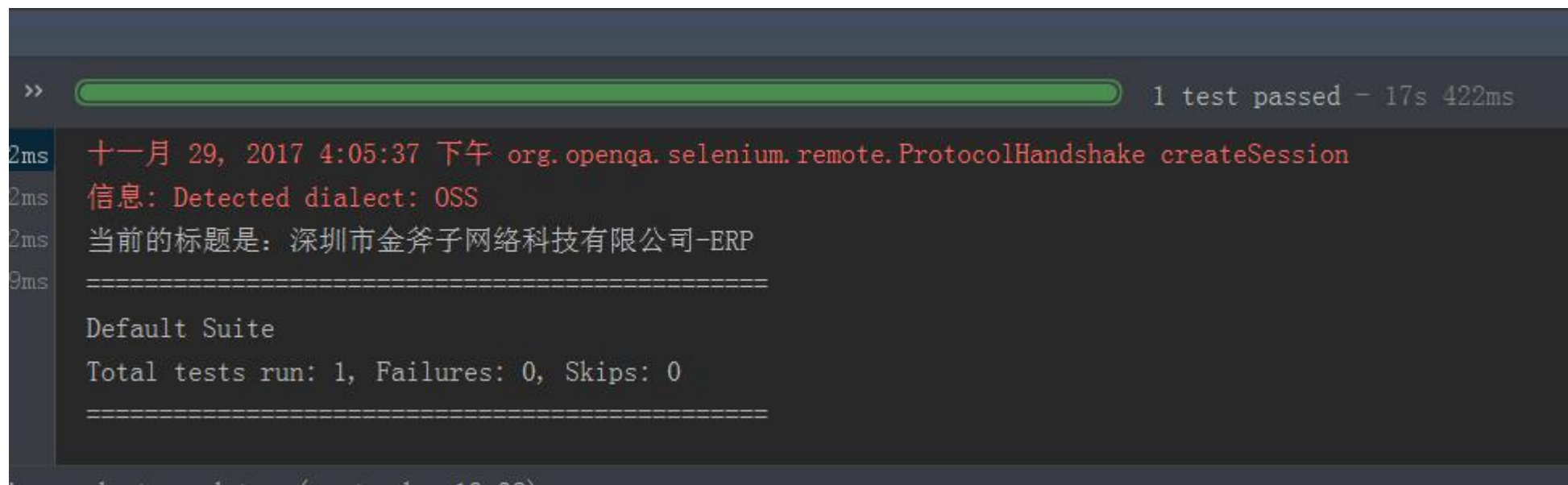
TestNg是一种单元测试框架，由Cedric Beust创建。其灵感来自JUnit和JUnit4的，但引入了一些新的功能，使其功能更强大，使用更方便。TestNG是一个开源自动化测试框架;NG表示下一代的意思（Next Generation）。TestNG是类似于JUnit（特别是JUnit 4），但它不是一个JUnit扩展。它的灵感来源于JUnit。它的目的是优于JUnit的，尤其是当测试集成的类。TestNG的创造者是Cedric Beust（塞德里克·博伊斯特）。TestNG消除了大部分的旧框架的限制，使开发人员能够编写更加灵活和强大的测试。因为它在很大程度上借鉴了Java注解（JDK5.0引入的）来定义的测试，它也可以告诉你如何使用这个新功能在真实的Java语言生产环境中。

- 2.2 TestNg的优点
- TestNg具有以下优点：
  - ( 1 ) 漂亮的HTML格式测试报告
  - ( 2 ) 支持并发测试
  - ( 3 ) 参数化测试更简单
  - ( 4 ) 支持输出日志
  - ( 5 ) 支持更多功能的注释

- 2.3 编写TestNg测试用例的步骤
- 编写TestNg测试用例的步骤如下：
- 第一步：使用idea生成TestNg的测试框架
- 第二步：在生成的程序框架中编写测试代码逻辑
- 第三步：根据测试代码逻辑，插入TestNg注解标签
- 第四部：配置testng.xml文件，设定测试类、测试方法、测试分组的执行信息。
- 第五步：执行TestNg的测试程序。

- 2.4 在TestNg中运行第一个WebDriver测试用例
- 运行过程如下：
- （1）启动idea，新建一个Java工程，命名为“Selenium\_Automated”，配置好运行WebDriver的相关jar文件，上一节已经配置好。
- （2）在src下面新建一个包名为“cn.TestScripts”
- （3）在包下面新建类，比如：“firstTestngDemo”，在idea里面新建testng类，没有eclipse那么方便，在eclipse里面会帮你生成一些注释方法，在idea里面需要自己去添加注释方法
- 比如：@BeforeMethod //注解的方法将每个测试方法之前运行

- (4) 运行测试方法
- (5) 查看运行的结果



The screenshot displays the Selenium IDE interface. At the top, a green progress bar indicates the test status, followed by the text "1 test passed - 17s 422ms". Below this, a log of test execution steps is visible, including timestamps and messages such as "十一月 29, 2017 4:05:37 下午 org.openqa.selenium.remote.ProtocolHandshake createSession", "信息: Detected dialect: OSS", and "当前的标题是: 深圳市金斧子网络科技有限公司-ERP". The bottom section shows the test results summary: "Default Suite", "Total tests run: 1, Failures: 0, Skips: 0", and a series of equals signs indicating a successful outcome.

```
>> 1 test passed - 17s 422ms
2ms 十一月 29, 2017 4:05:37 下午 org.openqa.selenium.remote.ProtocolHandshake createSession
2ms 信息: Detected dialect: OSS
2ms 当前的标题是: 深圳市金斧子网络科技有限公司-ERP
9ms =====
Default Suite
Total tests run: 1, Failures: 0, Skips: 0
=====
```

## 2.5 TestNg的常用注解

- TestNg的常见测试用例组织结构如下：
  - • Test Suite由一个或者多个Test组成
  - • Test由一个或者多个测试class组成
  - • 一个测试class由一个或者多个方法组成
- 在testing.xml中的配置层级结构如下：
  - <suite name="Suite2">
  - <test name="test2">
  - <classes>
  - <class name="cn.TestScripts.FirstTestngDemo"/>
  - <class name="cn.TestScripts.Annotation"/>
  - </classes>
  - </test>
  - </suite>
  - 运行不同层级的测试用例时，可通过不同注解实现测试前的初始化工作、测试用例执行工作和测试后的清理工作。



注解	描述
@BeforeSuite	表示此注解的方法会在当前测试集合（suite）中的任一测试用例开始运行之前执行
@AfterSuite	表示此注解的方法会在当前测试集合（Suite）中的所有测试程序运行结束之后执行
@BeforeClass	表示此注解的方法会在当前测试类的任一测试用例开始运行前执行
@AfterClass	表示此注解的方法会在当前测试类的任一测试用例开始运行后执行
@BeforeTest	表示此注解的方法会在Test中任一测试用例开始运行前执行
@AfterTest	表示此注解的方法会在Test中任一测试用例运行结束后执行
@BeforeGroups	表示此注解的方法会在分组测试用例的任一测试用例开始运行前执行
@AfterGroups	表示此注解的方法会在分组测试用例的所有测试用例运行结束后执行
@BeforeMethod	表示此注解的方法会在每个测试方法开始运行前执行

注解	描述
@AfterMethod	表示此注解的方法会在每个测试方法运行结束后执行
@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	表示此注解的方法会被认为一个测试方法，即一个测试用例

## 使用注解编写TestNg测试示例：

```
public class Annotation {  
    @Test  
    public void testCase1() { System.out.print("测试用例1被执行\n"); }  
  
    @Test  
    public void testCase2() { System.out.print("测试用例2被执行\n"); }  
  
    @BeforeMethod  
    public void beforeMethod() { System.out.print("在每个测试方法开始运行前执行\n"); }  
  
    @AfterMethod  
    public void afterMethod() { System.out.print("在每个测试方法运行结束后执行\n"); }  
  
    @BeforeClass  
    public void beforeClass() {  
        System.out.print("在当前测试类的第一个测试方法开始调用前执行\n");  
    }  
  
    @AfterClass  
    public void afterClass() {  
        System.out.print("在当前测试类的最后一个测试方法结束运行后执行\n");  
    }  
}
```

Annotation

运行结果:

```
> All 3 tests passed - 21s 730ms

测试用例1被执行
在每个测试方法运行结束后执行
在每个测试方法开始运行前执行
测试用例2被执行
在每个测试方法运行结束后执行
在当前测试类的最后一个测试方法结束运行后执行
在测试类中的Test运行结束后执行
在当前测试结合（Suite）中的所有测试程序运行结束之后执行

=====
Suite2
Total tests run: 3, Failures: 0, Skips: 0
=====
```

每个含有注解的类方法如果被调用，均会打印出对应的注解含义，从执行的结果可以分辨出不同的注解方法会在何时被调用，从此实例可以更好的理解注解的执行含义，以后的项目中会经常用到这些。

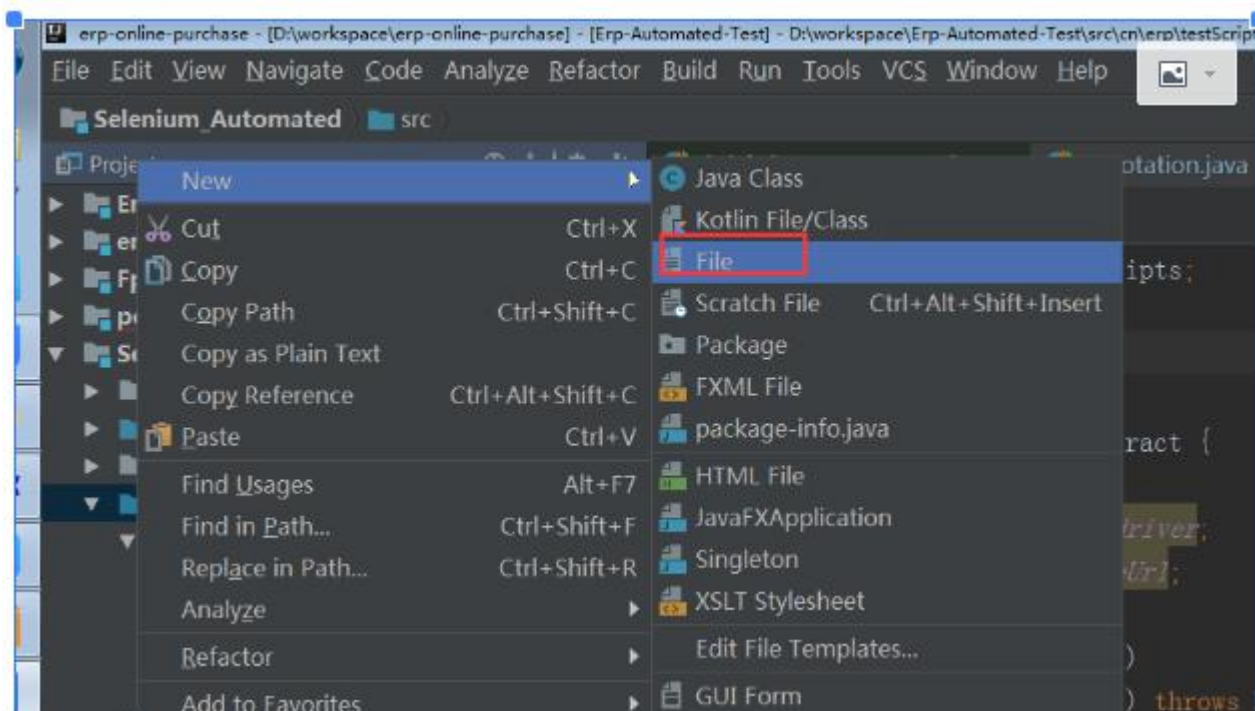
## 2.6 测试集合

在自动化测试的执行过程中，通常会产生批量运行多个测试用例的需求，此需要称为运行测试集合（Test Suite）。TestNg的测试用例可以是相互独立的，也可以按照特定的顺序来执行。

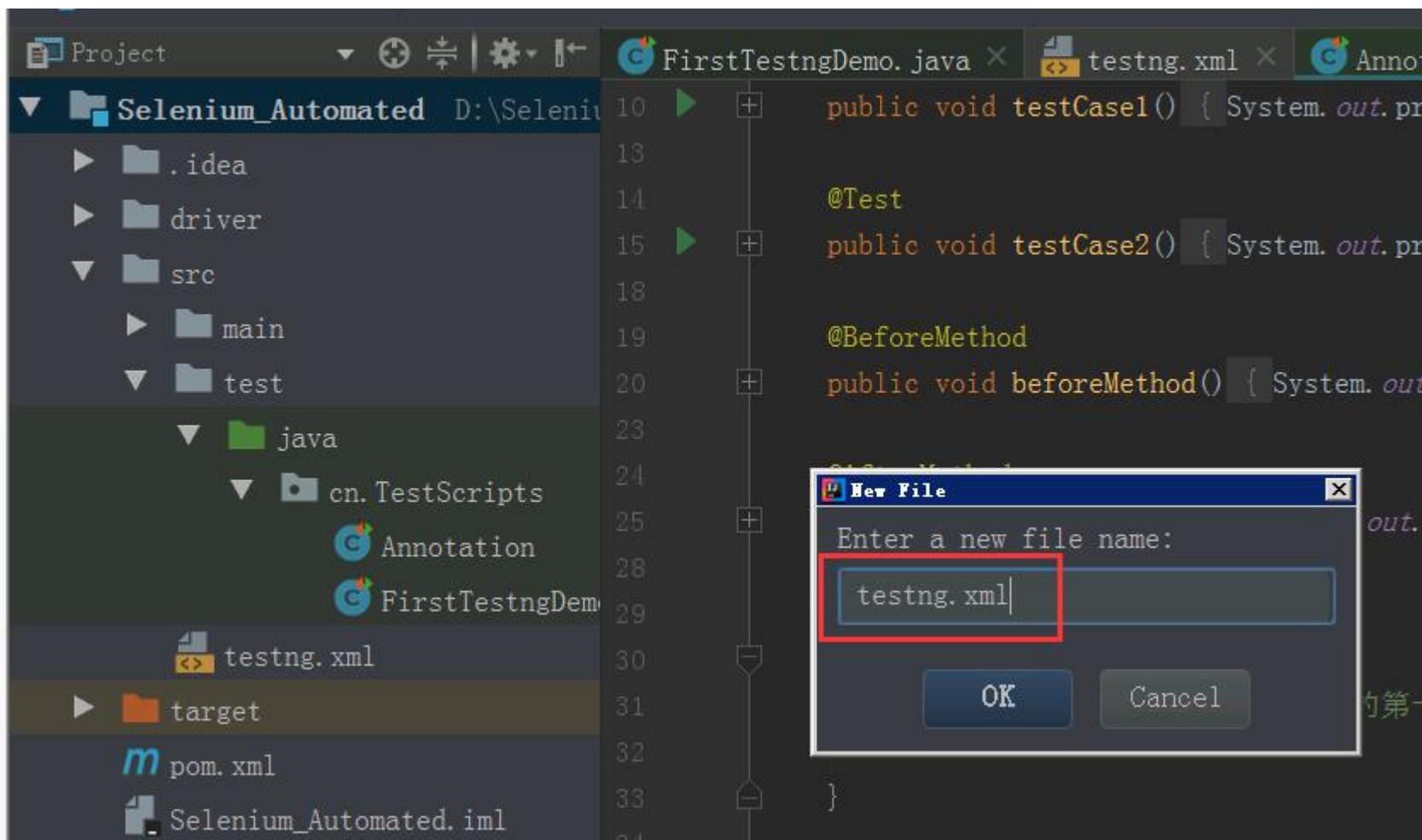
通过TestNg.xml的配置，可实现运行多个测试用例的不同组合。

操作步骤如下：

（1）在工程名字上面点击鼠标右键，在弹出的快捷菜单中选择New—>“file”的命令，如下图



( 2 ) 在new file输入框输入文档的名称：testng.xml

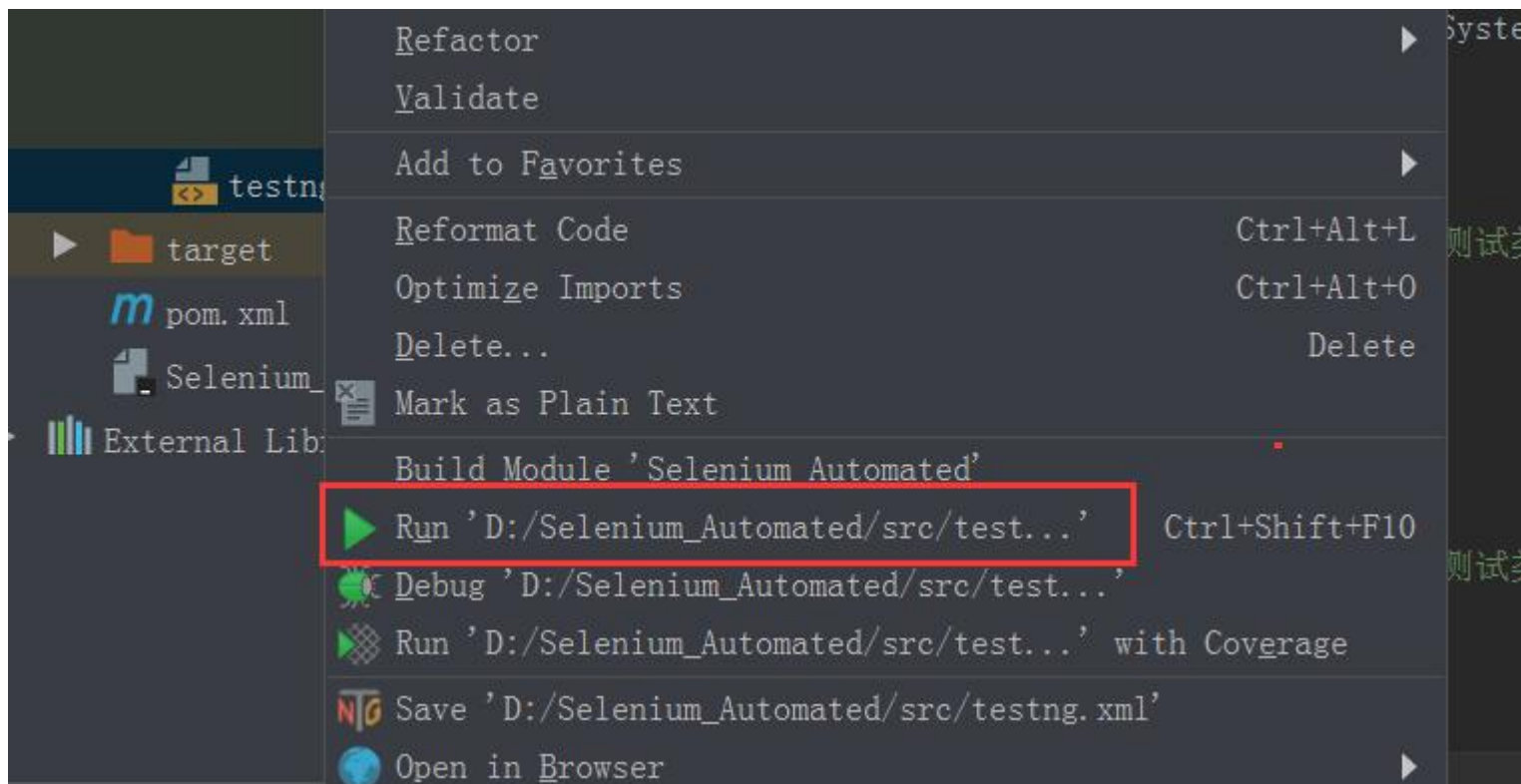




( 3 ) 在testng.xml的文件编辑区域输入如下内容并保存。其中，suite name是自定义的测试集合名称；test name定义测试名称；classes定义被运行的测试类，本示例中定义了当前工程中的两个测试类FirstTestngDemo和Annotation

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="Suite2">
  <test name="test2">
    <classes>
      <class name="cn.TestScripts.FirstTestngDemo"/>
      <class name="cn.TestScripts.Annotation"/>
    </classes>
  </test>
</suite>
```

( 4 ) 单击testng.xml , 右键选择Run 'D:\workspace\Selenium\_Automated'





运行测试结果：

[TestNG] Running:

D:\workspace\Selenium\_Automated\src\testng.xml

当前测试集合（Suite）中的所有测试程序开始执行之前执行

在测试类中的Test开始运行前执行

Starting ChromeDriver 2.29.461591 (62ebf098771772160f391d75e589dc567915b233) on port 42174

Only local connections are allowed.

在当前测试类的第一个测试方法开始调用前执行

在每个测试方法开始运行前执行

测试用例1被执行

在每个测试方法运行结束后执行

在每个测试方法开始运行前执行

测试用例2被执行

在每个测试方法运行结束后执行

在当前测试类的最后一个测试方法结束运行后执行

在测试类中的Test运行结束后执行

在当前测试结合（Suite）中的所有测试程序运行结束之后执行

=====

Suite2

Total tests run: 3, Failures: 0, Skips: 0

=====

Process finished with exit code 0

## 2.7 测试用例的分组

TestNG使用group关键字进行分组，用来执行多个Test的测试用例。

示例如下：

```
@Test(groups = {"合同管理"})  
public void generateContractadd()
```

//更新合同

```
@Test(groups = {"合同管理"})  
public void generateContractupdate()
```

//上传资产证明

```
@Test(groups = {"附件管理"})  
public void AssetProof() {
```

//上传风险揭示书

```
@Test(groups = {"附件管理"})
```

```
public void RiskStatement()
```

//上传风险测评书

```
@Test(groups = {"附件管理"})
```

```
public void RiskEvaluate()
```

testng.xml的内容配置如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="Suite1">
  <test name="ThriftsignContractTest">
    <groups>
      <run>
        <include name="合同管理"/>
      </run>
    </groups>
    <classes>
      <class name="com.jfz.erp.online.purchase.thrifttestng.ThriftsignContractTest"/>
    </classes>
  </test>
</suite>
```

运行查看结果

同时执行两个分组中的所有测试用例，请将testng.xml修改为如下内容

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="Suite1">
  <test name="ThrifftsignContractTest">
    <groups>
      <define name = "All">
        <include name="合同管理"/>
        <include name="附件管理"/>
      </define>
      <run>
        <include name="All"/>
      </run>
    </groups>
    <classes>
      <class name="com.jfz.erp.online.purchase.thrifttestng.ThrifftsignContractTest"/>
    </classes>
  </test>
</suite>
```

## 2.8 依赖测试

某些复杂的测试场景需要按照某个特定顺序执行测试用例，以此保证某个测试用例被执行之后才执行其它测试用例，此测试场景运行需求称为依赖测试。通过依赖测试，可在不同测试方法间共享数据和程序状态。testNg支持依赖测试，使用dependsOnMethods参数来实现。

```
testngDemo.java × testng.xml × Annotation.java × TestTestng.java ×  
@Test(dependsOnMethods = {"OpenBrowser"})  
public void FpLogin() {  
    driver.get(baseUrl + "");  
  
    driver.manage().window().maximize();  
  
    // 文本框内输入用户名  
    WebElement input_txt=driver.findElement(By.name("username"));  
    Assert.assertTrue(input_txt.isDisplayed());  
    input_txt.sendKeys(...charSequences: "defang1");  
    // 文本框内输入密码  
    driver.findElement(By.name("password")).sendKeys(...charSequences: "123456");  
    // 点击登录  
    driver.findElement(By.className("submit_wrap")).click();  
  
    System.out.print("FpLogin方法被调用\n");  
}  
  
@Test  
public void OpenBrowser() {  
    baseUrl = "http://10.1.2.211:8080/login.jsp";  
    // 设定连接chrome浏览器驱动程序所在的磁盘位置，并添加为系统属性值  
TestTestng > FpLogin()
```

## ( 1 ) 查看运行结果

## ( 2 ) 更多说明：

此测试代码中共有3个测试方法，分别实现了测试逻辑是打开浏览器，用户登录和关闭浏览器，此测试逻辑在测试工作中很常见，且必须按照固定顺序执行，否则测试用例就会无法执行成功。FpLogin方法使用了参数dependsOnMethods = {"OpenBrowser"},表示在OpenBrowser测试方法被调用后才能执行FpLogin方法；tearDown方法使用了参数@Test (dependsOnMethods = "FpLogin")，表示在FpLogin测试方法被调用后才能执行tearDown方法。

## 2.9 特定顺序执行测试用例

使用参数priority可实现按照特定熟悉执行测试用例

测试代码：

//新建合同

@Test(priority = 1)

public void generateContractadd()

//更新合同

@Test(priority = 2)

public void generateContractupdate()

//上传资产证明

@Test(priority = 3)

public void AssetProof() {

//上传风险揭示书

@Test(priority = 4)

public void RiskStatement()

//上传风险测评书

@Test(priority = 5)

public void RiskEvaluate()

运行结果：

## 2.10 跳过某个测试

使用参数enabled=false来跳过某测试方法

测试代码：

//新建合同

@Test(priority = 1)

public void generateContractadd()

//更新合同

@Test(priority = 2)

public void generateContractupdate()

//上传资产证明

@Test(priority = 3)

public void AssetProof() {

//上传风险揭示书

@Test (priority = 4,enabled = false)

public void RiskStatement()

//上传风险测评书

@Test(priority = 5)

public void RiskEvaluate()

运行测试结果：



## 2.11 断言

TestNg允许在测试执行过程中对测试程序变量的中间状态进行断言（Assert）判断，从而辅助判断测试用例的执行是成功还是失败

TestNg中常用的断言方法如下：

- assertTrue: 判断是否为true
- assertFalse: 判断是否为false
- assertEquals: 判断引用地址是否相同
- assertNotEquals: 判断引用地址是否不相同
- assertNull: 判断是否为null
- assertNotNull: 判断是否不为null
- assertEquals: 判断是否相等，object类型的对象需要实现hashCode及equals方法
- assertNotEquals: 判断是否不相等
- assertEqualsNoOrder: 判断忽略顺序是否相等

下面用测试实例来说明断言的使用方法

测试用例：

```
Assert.assertEquals("新增合同成功!", message);
```

```
Assert.assertEquals("更新合同成功!", message);
```

```
// 文本框内输入用户名
```

```
WebElement input_txt=driver.findElement(By.name("username"));
```

```
Assert.assertTrue(input_txt.isDisplayed());
```

```
input_txt.sendKeys("defang1");
```

input\_txt.isDisplayed()：用例判断用户名输入框是否在页面存在，若显示则此函数返回值为true，若无显示则返回值为false。

Assert.assertTrue(input\_txt.isDisplayed());用来判断input\_txt.isDisplayed()：函数返回值是否为true。若函数返回实际值为true，则断言测试用例执行成功，测试程序会继续执行后续语句；否则当前测试用例会被设定为执行失败。

# 作业

- 1、TestNg测试集合的使用
- 2、TestNg分组执行测试用例
- 3、TestNg（依赖测试，特定顺序，跳过某个测试，断言）的使用

# Thanks!

科技提升投资品质

