**TestNG**是开源的Java自动化测试框架，框架的设计灵感来源于JUnit 和 NUnit。其消除了大部分的旧框架的限制，使测试开发人员能够编写更加灵活和强大的测试。

注解 Annotation 是从JDK1.5 开始引入到Java语言中，TestNG 借鉴了Java注解来定义测试。

**TestNG Maven依赖**

使用Maven作为项目管理工具，可以对 Java 项目进行构建、依赖管理。我们需要在pom.xml中添加 testng 依赖，如下：

<dependencies>

   <!-- https://mvnrepository.com/artifact/org.testng/testng -->

   <dependency>

       <groupId>org.testng</groupId>

       <artifactId>testng</artifactId>

       <version>7.0.0-beta1</version>

       <scope>test</scope>

   </dependency>

</dependencies>

**一个简单的示例**

如下，我们创建 **baseDemoOtherTest**测试类，通过@Test注解来实现。

// baseDemoOtherTest.java

package testng.base.demo;

import org.testng.annotations.Test;

public class baseDemoOtherTest {

   @Test

   public void testOtherDemo1() {

       System.out.println("           运行 testOtherDemo1()测试方法");

  }

   @Test

   public void testOtherDemo2() {

       System.out.println("           运行 testOtherDemo2()测试方法");

  }

}

**TestNG常用注解及使用**

@Test            将类或方法标记为测试的一部分。

@BeforeSuite     在该套件的所有测试之前运行注解方法，仅运行一次（套件测试是一起运行的多个测试类）。

@AfterSuite      在该套件的所有测试之后运行注解方法，仅运行一次（套件测试是一起运行的多个测试类）。

@BeforeClass     在调用当前类的第一个测试方法之前运行注解方法，注解方法仅运行一次。

@AfterClass      在调用当前类的所有测试方法之后运行注解方法，注解方法仅运行一次。

@BeforeTest      在属于<test>标签内的类的所有测试方法之前运行注解方法。

@AfterTest       在属于<test>标签内的类的所有测试方法之后运行注解方法。

@BeforeMethod    在每个测试方法之前运行注解方法。

@AfterMethod     在每个测试方法之后运行注解方法。

@BeforeGroups    在配置的分组中第一个方法运行之前运行注解方法。

@AfterGroups     在配置的分组中所有的方法运行之后运行注解方法

增加一个测试类 baseDemoTest

package testng.base.demo;

import org.testng.annotations.\*;

public class baseDemoTest {

   @BeforeSuite

   public void beforeSuite() {

       System.out.println("@BeforeSuite:测试套件(当前xml中<suite>标签)之前运行@BeforeSuite注释方法");

  }

   @AfterSuite

   public void afterSuite() {

       System.out.println("@AfterSuite:测试套件(当前xml中<suite>标签)之后运行@AfterSuite注释方法");

  }

   @BeforeTest

   public void beforeTest() {

       System.out.println("   @BeforeTest:测试用例(当前xml中<test>标签)之前运行@BeforeTest注释方法");

  }

   @AfterTest

   public void afterTest() {

       System.out.println("   @AfterTest:测试用例(当前xml中<test>标签)之后运行@AfterTest注释方法");

  }

   @BeforeMethod

   public void beforeMethod() {

       S

ystem.out.println("       @BeforeMethod:当前类每个测试方法(@Test)之前运行@BeforeMethod注释方法");

  }

   @AfterMethod

   public void AfterMethod() {

       System.out.println("       @AfterMethod:当前类每个测试方法(@Test)之后运行@AfterMethod注释方法");

  }

   @BeforeGroups(value = {"A"})

   public void beforeGroups() {

       System.out.println("       @BeforeGroups:在A组第一个方法运行之前运行@BeforeGroups注释方法");

  }

   @AfterGroups(value = {"A"})

   public void afterGroups() {

       System.out.println("       @AfterGroups:在A组最后一个方法运行之后运行@AfterGroups注释方法");

  }

   @Test

   public void testDemo1() {

       System.out.println("           @Test:运行testDemo1()测试方法");

  }

   @Test(groups = {"A"})

   public void testDemo2() {

       System.out.println("           @Test:运行testDemo2()测试方法,归属A组");

  }S

   @Test(groups = {"A"})

   public void testDemo3() {

       System.out.println("           @Test:运行testDemo3()测试方法,归属A组");

  }

}

在工程目录中新建一个自定义xml配置文件testng.xml

<?xml version="1.0" encoding="UTF-8"?>

<!-- @BeforeSuite -->

<suite name="suiteDemo">

   <!-- @BeforeTest -->

   <test name="caseDemo">

       <classes>

           <class name="testng.base.demo.baseDemoTest" />

       </classes>

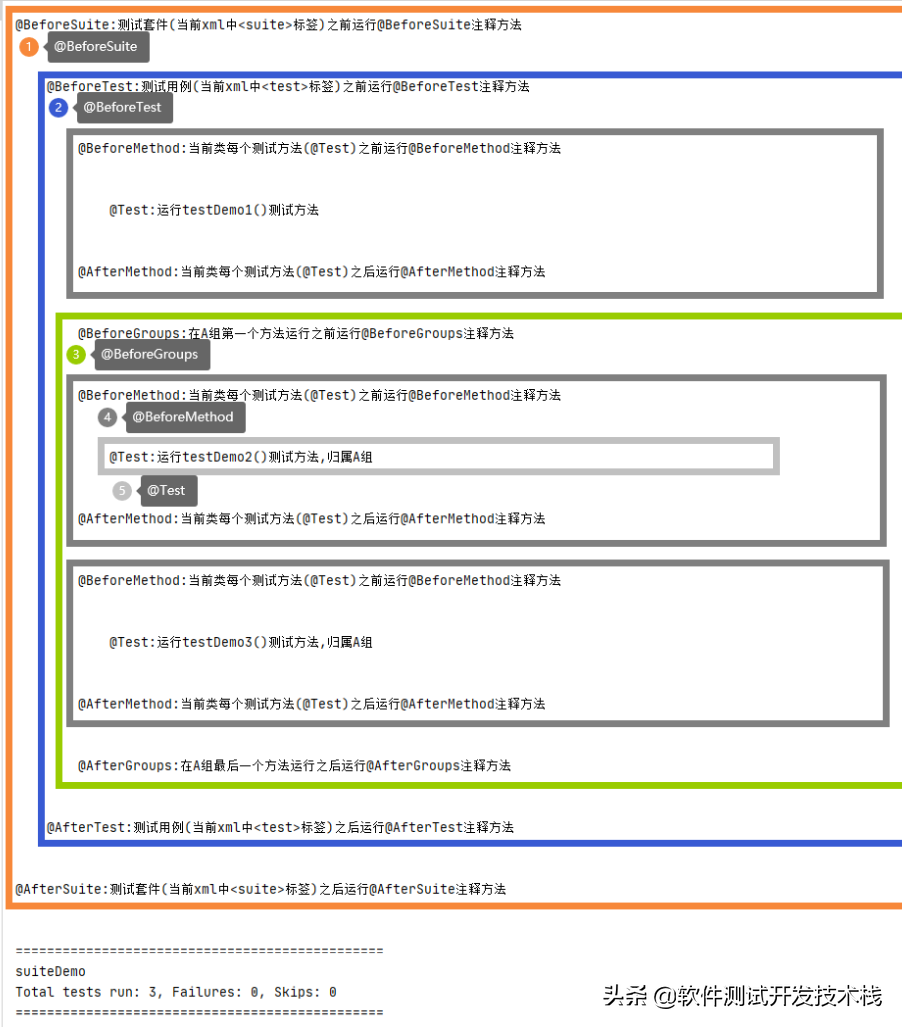
   </test>

   <!-- @AfterTest -->

</suite>

<!-- @AfterSuite -->

运行testng.xml 配置测试，结果如下，我们可以非常清晰的看到执行顺序。



总结一下执行顺序如下：**@BeforeSuite->@BeforeTest->@BeforeClass-> {@BeforeMethod->@Test->@AfterMethod} ->@AfterClass->@AfterTest->@AfterSuite**。其中{}内的与多少个@Test注解的测试方法，就循环执行多少次。

**注解的作用域**

我们接下来改造一下第一个demo测试类，如下，我们将其中一个测试方法设置为分组A。

package testng.base.demo;

import org.testng.annotations.Test;

public class baseDemoOtherTest {

   @Test

   public void testOtherDemo1() {

       System.out.println("           运行 testOtherDemo1()测试方法");

  }

   @Test(groups = {"A"})

   public void testOtherDemo2() {

       System.out.println("           运行 testOtherDemo2()测试方法");

  }

}

更新下testng.xml配置，如下：

<?xml version="1.0" encoding="UTF-8"?>

<!-- @BeforeSuite -->

<suite name="suiteDemo">

   <!-- @BeforeTest -->

   <test name="caseDemo">

       <classes>

           <class name="testng.base.demo.baseDemoOtherTest" />

           <class name="testng.base.demo.baseDemoTest" />

       </classes>

   </test>

   <!-- @AfterTest -->

   <!-- @BeforeTest -->

   <test name="caseDemo2">

       <classes>

           <class name="testng.base.demo.baseDemoOtherTest" />

       </classes>

   </test>

   <!-- @AfterTest -->

</suite>

<!-- @AfterSuite -->

运行testng.xml 配置测试，从如下结果可以看出，@BeforeGroups、@AfterGroups的作用域是可以跨类的，但类必须是在testng.xml中同一个测试用例(<test>标签)范围内；



**配置文件xml常用标签**

<!-- 测试套件,通常由多个<test>组成 -->

<suite  name="测试套件名称 必填属性" verbose="运行的级别" parallel="是否运行多线程来运行这个套"

      thread="线程数量" annotations="在测试中使用的注释类型"

      time-out="所有测试方法上使用的默认超时时间" >

   <!-- 测试用例,属性name为必填属性 -->

   <test name="caseDemo1">

      <!-- 用例中包含的类-->

       <classes>

           <!-- 测试类，属性name为必填属性-->

           <class name="testng.base.demo.baseDemoOtherTest" >

               <!-- 指定测试类中包含或排除的方法-->

               <methods>

                   <!-- 运行 testDemo1()方法 属性name为必填属性-->

                   <include name="testDemo1" />

                   <!-- 排除 testDemo1()方法 属性name为必填属性-->

                   <exclude name="testDemo2" />

               </methods>

           </class>

       </classes>

   </test>

   <test name="caseDemo2">

       <!-- 用例中包含的包，包中所有的方法都会执行 -->

       <packages>

          <!-- 测试包，属性name为必填属性-->

           <package name="testng.base.demo" />

       </packages>

   </test>

    <test name="caseDemo3">

      <!-- 指定测试用例中要运行或排除运行的分组-->

       <groups>

            <run>

              <!-- 运行 A组所有测试方法 属性name为必填属性-->

            <include name="A" />

               <!-- 排除 B组所有测试方法 属性name为必填属性-->

               <exclude name="B" />

           </run>

       </groups>

</test>

</suite>