TestNG是什么

简单的说TestNG和Junit一样都是单元测试的一种框架。但是因为JUnit在过去几年的版本迭代比较慢，已经没办法很好的满足最新的测试任务。相对于JUnit而言，TestNG具有以下优势：

支持依赖测试方法；并行测试；负载测试；局部故障；灵活的插件API;支持多线程测试；

TestNG插件的安装

eclipse中插件的安装

eclipse--Help--Install New Software 在地址栏输入：http://beust.com/eclipse,选择TestNG进行安装。

https://www.cnblogs.com/yigedapangzhi/p/10203981.html

IDEA中插件的安装

在testNG中一般默认会自带testNG插件，如果未安装插件，可以参考以下的教材进行安装。

https://blog.csdn.net/u010270891/article/details/82978260

HelloTestNG

创建一个Maven项目，项目中引入testNG。

```
        <dependency>
            <groupId>org.testng</groupId>
            <artifactId>testng</artifactId>
            <version>6.13</version>
            <scope>test</scope>
        </dependency>
```

在项目中创建第一个"Hello World"的模拟测试类com.h3c.hello.HelloTestNG.java

```
package com.h3c.hello;

import org.testng.annotations.Test;

public class HelloTestNG {
    @Test
    public void hello() {
        System.out.println("hello testNG");
    }
}
```

创建testNG的核心配置文件helloTestNG.xml

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="testNG第一个测试" verbose="1" >
    <test name="HelloTestNG">
        <classes>
            <class name="com.h3c.hello.HelloTestNG"/>
        </classes>
    </test>
</suite>
```

右键选择 helloTestNG.xml 文件，选择 Run as ---- TestNG Suit 进行执行。

本质是执行了这个命令：java org.testng.TestNG testng1.xml [testng2.xml testng3.xml ...]

testNG注解介绍

```
@BeforeSuite      在该套件的所有测试都运行在注释的方法之前，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都运行在注释方法之后，仅运行一次。
@BeforeClass      在调用当前类的第一个测试方法之前运行，注释方法仅运行一次。
@AfterClass       在调用当前类的第一个测试方法之后运行，注释方法仅运行一次
@BeforeTest       注释的方法将在属于<test>标签内的类的所有测试方法运行之前运行。
@AfterTest        注释的方法将在属于<test>标签内的类的所有测试方法运行之后运行。
@BeforeGroups     配置方法将在之前运行组列表。 此方法保证在调用属于这些组中的任何一个的第一个测试方法之前不久运行。
@AfterGroups      此配置方法将在之后运行组列表。该方法保证在调用属于任何这些组的最后一个测试方法之后不久运行。
@BeforeMethod     注释方法将在每个测试方法之前运行。
@AfterMethod      注释方法将在每个测试方法之后运行。
@Parameters       描述如何将参数传递给@Test方法。
@DataProvider     标记一种方法来提供测试方法的数据。 注释方法必须返回一个Object [] []，其中每个Object []可以被分配给测试方法的参数列表。 要从该DataProvider接收数据的@Test方法需要使用与此注释名称相等的dataProvider名称。
@Factory          将一个方法标记为工厂，返回TestNG将被用作测试类的对象。 该方法必须返回Object []。
@Listeners        定义测试类上的侦听器。
@Test             将类或方法标记为测试的一部分。
```

testNG的配置文件介绍

```
<suite>  套件，根标签，通常由几个<test组成>
  属性：
    name              套件的名称，必须属性；
    verbose           运行的级别或详细程度；
    parallel          是否运行多线程来运行这个套件；
    thread-count      如果启用多线程，用于指定开户的线程数；
    annotations       在测试中使用的注释类型；
    time-out          在本测试中的所有测试方法上使用的默认超时时间；
<test>       测试用例，name为必须属性；
<classes>    用例中包含的类，子标签为<class name=”className”>;
<class>      测试类，其中属性name为必须属性；;
<packages>   用例中包含的包，包中所有的方法都会执行，子标签为<package name=”packageName”>;
<package>    测试包，name为必须属性；
<methods>    指定测试类中包含或排除的方法，子类为<include>,<exclude>;
<include>    指定需要测试的方法，name为必须属性；
```

<exclude> 指定类中不需要测试的方法，name为必须属性；
<groups> 指定测试用例中要运行或排除运行的分组，子标签为<run>,<run>下包含<include>,
<exclude>标签，<include>,<exclude>的name指定运行、不运行的分组；

testNG代码演示

testNG 常用测试注释 @Before* @After*

java:

```
package com.h3c;
import org.testng.annotations.*;

public class FirstTestNgTest {
    @Test
    public void testCase(){
        System.out.println("hello testNG");
    }

    @Test
    public void testCase1(){
        System.out.println("test case 1");
    }

    @BeforeSuite
    public void beforeSuite(){
        System.out.println("@beforeSuite");
    }

    @AfterSuite
    public void afterSuite(){
        System.out.println("@AfterSuite");
    }

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

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

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

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

}
```

xml:

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="测试套件" verbose="1" >
    <test name="简单测试">
        <classes>
            <class name="com.h3c.FirstTestNgTest"/>
        </classes>
    </test>
</suite>
```

测试用例分组（group）

java

```
package com.h3c.group;

import org.testng.annotations.Test;

public class GroupTest {

    @Test(groups={"1"})
    public void test1(){
        System.out.println("group test 1");
    }

    @Test(groups={"2"})
    public void test2(){
        System.out.println("group test 2");
    }

    @Test(groups={"3"})
    public void test3(){
        System.out.println("group test 3");
    }
}
```

xml

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="测试套件" verbose="1" >
    <test name="组测试">
        <groups>
            <run>
                <include name="1"/>
               <!-- <include name="2"/> -->
               <!-- <include name="3"/> -->
            </run>
        </groups>
        <classes>
            <class name="com.h3c.group.GroupTest"/>
        </classes>
```

```
    </test>
</suite>
```

根据包路径进行测试（package）

java

pg1:

```
package com.h3c.pg1;

import org.testng.annotations.Test;

public class Pg1Test {
    @Test
    public void printTestOut(){
        System.out.println("pg1 test");
    }
}
```

pg2:

```
package com.h3c.pg2;

import org.testng.annotations.Test;

public class Pg2Test {
    @Test
    public void printTestOut(){
        System.out.println("pg2 test");
    }
}
```

xml:

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="测试套件" verbose="1" >
    <test name="包测试">
        <packages>
            <package name="com.h3c.pg1"/>
            <package name="com.h3c.pg2"/>
        </packages>
    </test>
</suite>
```