

07 TestNG基础使用

本章大纲

7.1 TestNG 介绍

7.2 编写测试用例的步骤

7.3 TestNG的常用注解

7.4 testng.xml

7.5 数据提供者

7.6 断言

7.7 TestNG与 Junit4不同

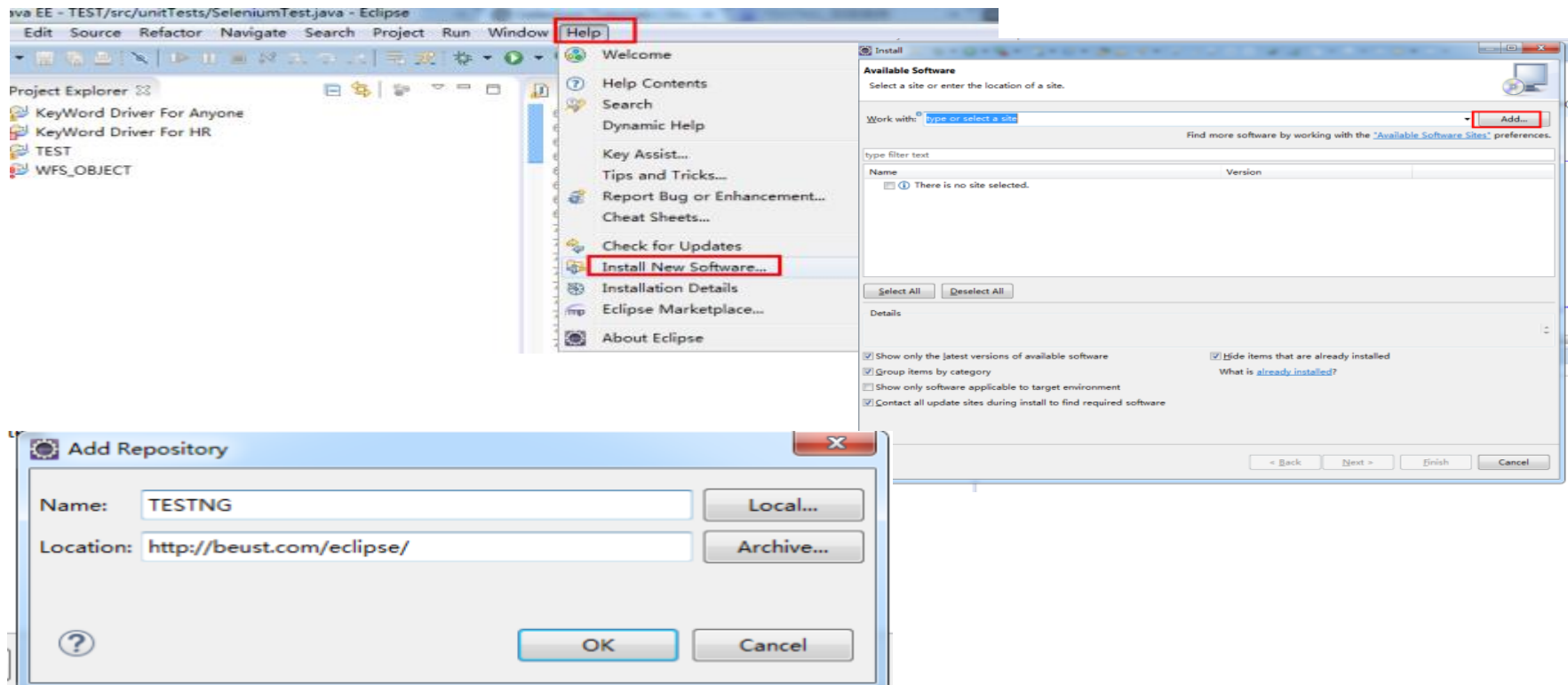
TestNG介绍

- TestNG (Test Next Generation) , 顾名思义, 下一代的测试框架。它借鉴了JUnit和Nuit框架的优秀设计思想, 引入更易用和更强大的功能。它是基于J2SE5.0的注解特性的而构建的轻量级的单元测试框架结构。
- TestNG比JUnit更强大, 提供了更多的扩展功能, 消除了一些老式框架的限制, 让程序员通过注解、分组、序列和参数化等多种方式组织和执行自动化测试脚本。
- 官网: <http://testng.org/doc/index.html>

TestNG的优点

- 支持更多的注解
- 漂亮的HTML格式测试报告
- 参数化测试更简单
- 支持输出日志
- 支持并发测试

安装TestNG



本章大纲

7.1 TestNG 介绍

7.2 编写测试用例的步骤

7.3 TestNG的常用注解

7.4 testng.xml

7.5 数据提供者

7.6 断言

7.7 TestNG与 Junit4不同

编写测试用例的步骤

1. 编写测试代码逻辑
2. 插入TestNG注解标签
3. 配置TestNG.xml文件，设定测试类、测试方法、测试分组的执行信息。
4. 执行TestNG查看测试报告

本章大纲

7.1 TestNG 介绍

7.2 编写测试用例的步骤

7.3 TestNG的常用注解

7.4 testng.xml

7.5 数据提供者

7.6 断言

7.7 TestNG与 Junit4不同

TestNG的常用注解

Annotations	含义
@Test	此注解的方法会被认为是一个测试方法，即一个测试用例
@BeforeMethod	被注释的方法将在每一个测试用例调用前运行
@AfterMethod	被注释的方法将在每一个测试用例调用后运行
@BeforeTest	被注释的方法将在Test中第一测试用例开始运行前运行
@AfterTest	被注释的方法将在在Test中所有测试用例运行后运行
@BeforeClass	被注释的方法将在当前类的第一个测试用例调用前运行
@AfterClass	被注释的方法将在当前类的所有测试用例调用后运行
@BeforeSuite	被注释的方法将在当前测试集合（suite）任一测试用例运行前运行
@AfterSuite	被注释的方法将在当前测试集合（suite）所有测试用例运行后运行
@BeforeGroups	被注释的方法将在分组测试用例的任一的测试用例前执行
@AfterGroups	被注释的方法将在分组测试用例的所有的测试用例后执行

测试方法的执行顺序

一个类里面可能存在多个测试方法(被**@Test**注解的方法)，默认测试方法的执行顺序是按照**方法名的字典序升序**排序执行的。

测试方法的执行顺序

使用priority指定执行顺序(默认值为0)，数值越小，越靠前执行

```
@Test(priority = 0)
```

```
public void testMethod1() { }
```

```
@Test(priority = 1)
```

```
public void testMethod2() { }
```

```
@Test(priority = 2)
```

```
public void testMethod3() { }
```

这样指定以后，将按照testMethod1、testMethod2、testMethod3顺序执行

跳过某个测试方法

- 使用参数enabled=**false**来跳过某测试方法

```
@Test(enabled=false)
public void test4(){
    System.out.println("test4方法");
}
```

依赖测试

某个测试用例被执行之后才执行其他测试用例，此测试场景运行需求称为依赖测试。通过参数`dependsOnMethods`依赖测试，可在不同测试方法间共享数据和程序状态。

```
@Test
public void openBrowser(){
    System.out.println("openBrowser方法");
}

@Test(dependsOnMethods="openBrowser")
public void login(){
    System.out.println("login方法");
}

@Test(dependsOnMethods="login")
public void logOut(){a
    System.out.println("logOut方法");
}
```

测试用例的分组

@Test(groups= "分组名")

例如：

```
public class Grouping {  
    @Test(groups="人")  
    public void student(){  
        System.out.println("student方法被调用");  
    }  
    @Test(groups="人")  
    public void teacher(){  
        System.out.println("teacher方法被调用");  
    }  
    @Test(groups="动物")  
    public void dog(){  
        System.out.println("dog方法被调用");  
    }  
    @Test(groups="动物")  
    public void cat(){  
        System.out.println("cat方法被调用");  
    }  
    @Test(groups={"人","动物"})  
    public void feeder(){  
        System.out.println("feeder方法被调用");  
    }  
}
```

测试报告中的自定义日志

TestNG提供了日志的功能，在测试过程中可以通过自定义的方式记录测试脚本的运行信息

```
@Test
public void openBrowser(){
    System.out.println("openBrowser方法");
    Reporter.Log("打开浏览器");
}
```

com.edu.cal.DependsOnMethodsTest#openBrowser

Messages
打开浏览器

本章大纲

7.1 TestNG 介绍

7.2 编写测试用例的步骤

7.3 TestNG的常用注解

7.4 testng.xml

7.5 数据提供者

7.6 断言

7.7 TestNG与 Junit4不同

testng.xml

TestNG的用例组织结构：

- Test Suite由一个或者多个Test组成
- Test由一个或多个测试class组成
- 一个测试class由一个或多个测试方法组成

testng.xml

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >

<suite name="Suite1" verbose="1" >
  <test name="Nopackage" >
    <classes>
      <class name="NoPackageTest" />
    </classes>
  </test>

  <test name="Regression1">
    <classes>
      <class name="test.sample.ParameterSample"/>
      <class name="test.sample.ParameterTest"/>
    </classes>
  </test>
</suite>
```

testng.xml-执行顺序

当preserve-order= "true" 是，可以保证节点下面的方法是按顺序执行的。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >

<suite name="Suite1" verbose="1">
  <test name="t1">
    <classes>
      <class name="com.edu.CalculatorTest" />
    </classes>
  </test>
  <test name="t2" preserve-order="true">
    <classes>
      <class name="com.edu.AnimalTest" >
        <methods>
          <include name="testDog" />
          <include name="testCat" />
          <include name="testPig" />
        </methods>
      </class>
    </classes>
  </test>
</suite>
```

parallel的取值

1.parallel="methods"

TestNG 会在不同的线程中运行测试方法，除非那些互相依赖的方法。那些相互依赖的方法会运行在同一个线程中，并且遵照其执行顺序。

2.parallel="tests"

不同test 下的用例可以在不同的线程执行，相同test tag下的用例只能在同一个线程中执行。

3.parallel="classes"

不同class 下的用例可以在不同的线程执行，相同class 下的用例只能在同一个线程中执行。

TestNG参数化

- @Parameters("参数1", "参数2")

```
<suite name="testuite">  
<parameter name="sname" value="tom"/>
```

```
@Test  
@Parameters("sname")  
public void testUsername(String sname){  
    System.out.println(sname);  
}
```

测试方法的执行顺序

xml里面使用<include>指定需要执行的方法和顺序

比如:

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Preserve order test runs">
    <test name="Regression 1" preserve-order="true">
        <classes>
            <class name="com.pack.preserve.ClassOne">
                <methods>
                    <include name="B" />
                    <include name="A" />
                </methods>
            </class>
        </classes>
    </test>
</suite>
```

如上配置，ClassOne会执行两个测试方法，先执行B，然后执行A。

本章大纲

7.1 TestNG 介绍

7.2 编写测试用例的步骤

7.3 TestNG的常用注解

7.4 testng.xml

7.5 数据提供者

7.6 断言

7.7 TestNG与 Junit4不同

TestNG数据提供者

```
@DataProvider(name = "data")
public Object[][] createData() {
    return new Object[][] { { 1, 2, 3 }, { 0, 0, 1 }, { -1, 1, 0 },
                             { -1, -2, -3 } };
}
```

```
@Test(dataProvider = "data")
public void addtionTest(int a1, int a2, int result) {
    assertEquals(cal.add(a1, a2), result);
}
```


本章大纲

7.1 TestNG 介绍

7.2 编写测试用例的步骤

7.3 TestNG的常用注解

7.4 testng.xml

7.5 数据提供者

7.6 断言

7.7 TestNG与 Junit4不同

断言

<code>assertEquals(a,b)</code>	测试a是否等于b
<code>assertNotEquals(a, b)</code>	测试a是否不等于b
<code>assertFalse(a)</code>	测试a是否为false
<code>assertTrue(a)</code>	测试a是否为true
<code>assertNull(a)</code>	测试a是否为null
<code>assertNotNull(a)</code>	测试a是否非空
<code>assertSame(a, b)</code>	测试a和b是否都引用同一个对象
<code>assertNotSame(a, b)</code>	测试a和b是否没有都引用同一个对象

本章大纲

7.1 TestNG 介绍

7.2 编写测试用例的步骤

7.3 TestNG的常用注解

7.4 testng.xml

7.5 数据提供者

7.6 断言

7.7 TestNG与 Junit4不同

JUnit与TestNG不同

相同：

JUnit , TestNG执行测试方法之前，都会重新实例化测试类

不同：

- 1.TestNG提供了比JUnit更多的注解
- 2.TestNG使用xml配置文件可以任意组合出需要的各种测试。
- 3.TestNG可以实现并发测试（不同的级别，methods,tests,classes）

TestNG与JUnit4不同---（了解）

功能	JUnit	TestNG
标注为类/方法为测试类和方法	@Test	@Test
标注为在suite中所有测试之前运行	无	@BeforeSuite
标注为在suite中所有测试之后运行	无	@AfterSuite
标注为在测试之前运行（跨越了测试类）	无	@BeforeTest
标注为在测试之后运行	无	@AfterTest
标注为在测试Group中第一个测试方法之前运行	无	@BeforeGroups
标注为在测试Group中最后一个测试方法之后运行	无	@AfterGroups
标注为当前测试类中第一个测试方法之前运行	@BeforeClass	@BeforeClass
标注为当前测试类中最后一个测试方法之后运行	@AfterClass	@AfterClass
标注为在每次测试方法之前运行	@Before	@BeforeMethod
标注为在每次测试方法之后运行	@After	@AfterMethod
忽略某测试，让其不执行	@Ignore	@Test(enable=false)
本方法所依赖的方法列表。	无	@Test(dependsOnMethods="walk")
期待测试抛出什么异常	@Test(expected=XXXException.class)	@Test(exceptedExceptions=XXXException.class)
测试超时，如果测试的执行时间超过了毫秒为单位设置的时间，那么就停止测试并且标记为失败	@Test(timeout=1000)	@Test(timeout=1000)

注意：自动化测试规范

1. 测试类放在test包中，独立存放
2. 测试类名用XXXTest结尾
3. 测试方法用testMethod命名

练习 (selenium+testng)

作业1：实现两个浏览器的并发测试

作业2：实现mymovie前端的登录（三个用户参数化的方式登录）