5 TestNG单元测试框架

本章大纲

5.1 TestNG 介绍

5.2 编写测试用例的步骤

5.3 TestNG的常用注解

5.4 testng.xml

5.5 数据提供者

5.6 断言

5.7 TestNG与 Junit4不同

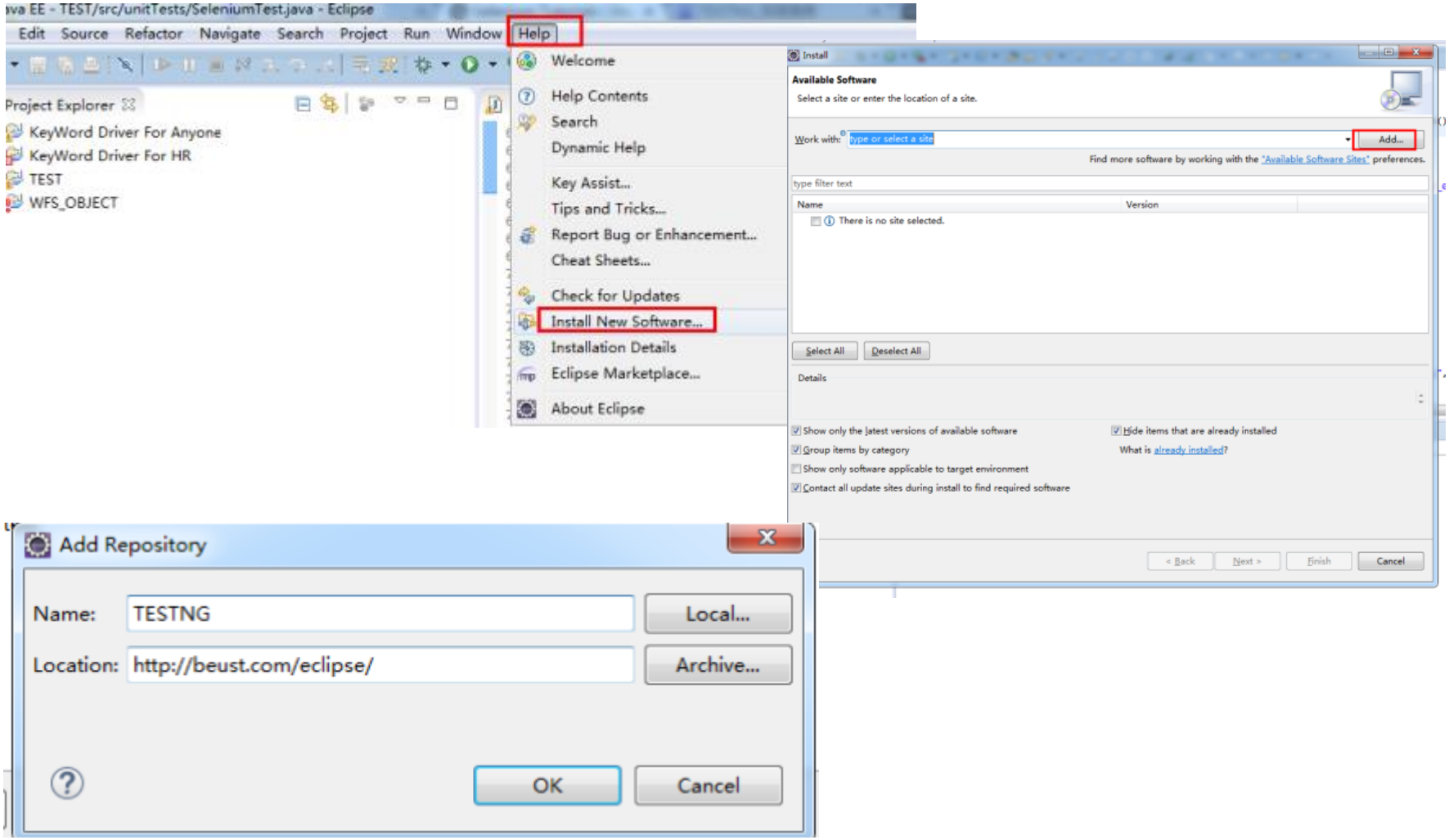
TestNG介绍

- 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


va EE - TEST/src/unitTests/SeleniumTest.java - Eclipse
Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
KeyWord Driver For Anyone
KeyWord Driver For HR
TEST
WFS_OBJECT
Welcome
Help Contents
Search
Dynamic Help
Key Assist...
Tips and Tricks...
Report Bug or Enhancement...
Cheat Sheets...
Check for Updates
Install New Software...
Installation Details
Eclipse Marketplace...
About Eclipse
Install
Available Software
Select a site or enter the location of a site.
Work with:
type or select a site
Add...
Find more software by working with the "Available Software Sites" preferences.
type filter text
Name
Version
There is no site selected.
Select All
Deselect All
Details
Show only the latest versions of available software
Hide items that are already installed
Group items by category
What is already installed?
Show only software applicable to target environment
Contact all update sites during install to find required software
< Back
Next >
Finish
Cancel
Add Repository
Name:
TESTNG
Local...
Location:
http://beust.com/eclipse/
Archive...
OK
Cancel

本章大纲

5.1 TestNG 介绍

5.2 编写测试用例的步骤

5.3 TestNG的常用注解

5.4 testng.xml

5.5 数据提供者

5.6 断言

5.7 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

5.1 TestNG 介绍

5.2 编写测试用例的步骤

5.3 TestNG的常用注解

5.4 testng.xml

5.5 数据提供者

5.6 断言

5.7 TestNG与 Junit4不同

TestNG的常用注解

Annotations	含义
@BeforeMethod	被注释的方法将在每一个测试方法调用前运行
@AfterMethod	被注释的方法将在每一个测试方法调用后运行
@BeforeTest	被注释的方法将在Test中任一测试运行前运行
@AfterTest	被注释的方法将在在Test中任一测试运行后运行
@BeforeClass	被注释的方法将在当前类的第一个测试方法调用前运行
@AfterClass	被注释的方法将在当前类的所有测试方法调用后运行
@BeforeSuite	被注释的方法将在当前测试集合（suite）任一测试运行前运行
@AfterSuite	被注释的方法将在当前测试集合（suite）任一测试运行后运行
@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任一的测试用例前后执行

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");
}
```

特定顺序执行测试用例

- 使用参数priority可以实现按照特定顺序执行测试用例

```java
@Test(priority=0)
public void test0(){
    System.out.println("test0方法");
}

@Test(priority=1)
public void test1(){
    System.out.println("test1方法");
}
@Test(priority=2)
public void test2(){
    System.out.println("test2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

5.1 TestNG 介绍

5.2 编写测试用例的步骤

5.3 TestNG的常用注解

5.4 testng.xml

5.5 数据提供者

5.6 断言

5.7 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

在testng.xml中，可以控制测试用例按顺序执行。 当preserve-order="true"是，可以保证节点下面的方法是按顺序执行的

```xml
<?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>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="testuite">
    <test name="grouping">
        <groups>
            <run>
                <exclude name="动物" />
                <include name="人" />
            </run>
        </groups>

        <classes>
            <class name="com.edu.cal.Grouping" />
            <class name="com.edu.cal.Test1" />
            <class name="com.edu.cal.Test3" />
        </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"

TestNG 会在相同的线程中运行相同的<test>标记下的所有方法，但是每个<test>标签中的所有方法会运行在不同的线程中。这样就 允许你把所有非线程安全的类分组到同一个<test>标签下，并且使其可以利用TestNG多线程的特性的同时，让这些类运行在相同的线程 中。

3.parallel="classes"

TestNG 会在相同线程中相同类中的运行所有的方法，但是每个类都会用不同的线程运行。

配置文件的编写

```
<suite name="Suite1" parallel="tests" thread-count="3">
    <test name="FirefoxTest">
        <parameter name="browser" value="firefox"/>
        <classes>
            <class name="com.baidu.test.MultipleBrowserTest" />
        </classes>
    </test>
    <test name="ieTest">
        <parameter name="browser" value="ie"/>
        <classes>
            <class name="com.baidu.test.MultipleBrowserTest" />
        </classes>
    </test>
    <test name="ChromeTest">
        <parameter name="browser" value="chrome" />
        <classes>
            <class name="com.baidu.test.MultipleBrowserTest" />
        </classes>
```

本章大纲

5.1 TestNG 介绍

5.2 编写测试用例的步骤

5.3 TestNG的常用注解

5.4 testng.xml

5.5 数据提供者

5.6 断言

5.7 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);
}
```

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);
}
```

本章大纲

5.1 TestNG 介绍

5.2 编写测试用例的步骤

5.3 TestNG的常用注解

5.4 testng.xml

5.5 数据提供者

5.6 断言

5.7 TestNG与 Junit4不同

断言

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

本章大纲

5.1 TestNG 介绍

5.2 编写测试用例的步骤

5.3 TestNG的常用注解

5.4 testng.xml

5.5 数据提供者

5.6 断言

5.7 TestNG与 Junit4不同

Junit与TestNG不同

1. Junit 执行每个测试方法之前，都会重新实例化测试类，TestNG不会
2. TestNG提供了比JUnit更多的annotations
3. TestNG使用xml配置文件可以任意组合出需要的各种测试。

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：单元测试规范

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