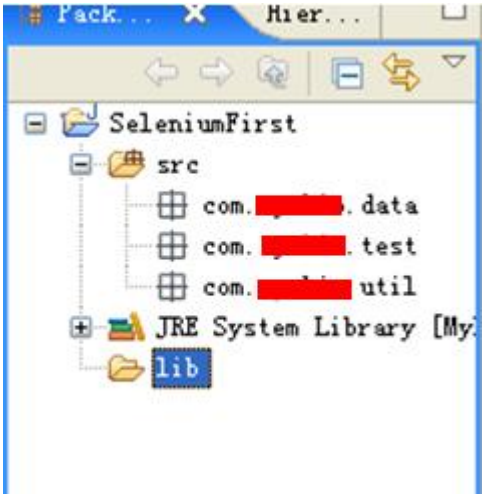
Junit 与Selenium结合
运行多测试类多个用例时程序设计方案

目录

1 环境配置

1） 新建 Java Project，命名为 Selenium

2） 创建目录结构，如下所示



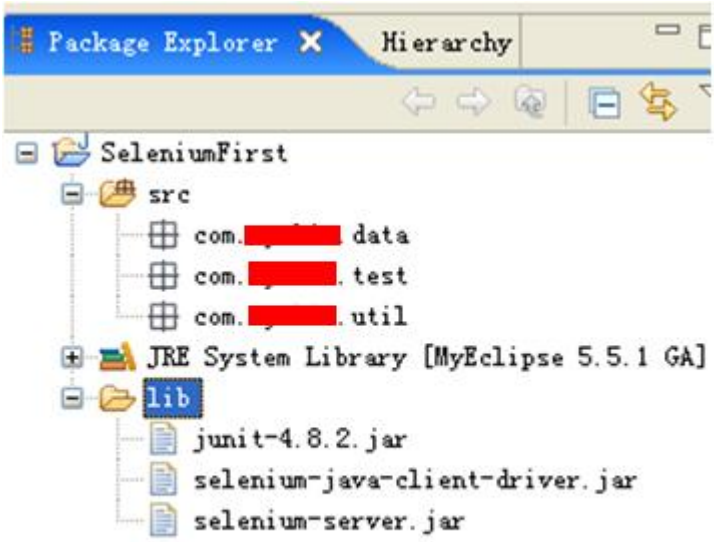
3） 解压 Selenium RC 插件，

拷贝 \selenium-remote-control-1.0.3\selenium-server-1.0.3 路径下的 jar 包 selenium-server.jar 到 lib 下

拷贝 selenium-remote-control-1.0.3\selenium-java-client-driver-1.0.1 下面的 selenium-java-client-driver.jar 拷贝到 lib 下

将下载到的 junit-4.8.2.jar 拷贝到 lib 下，

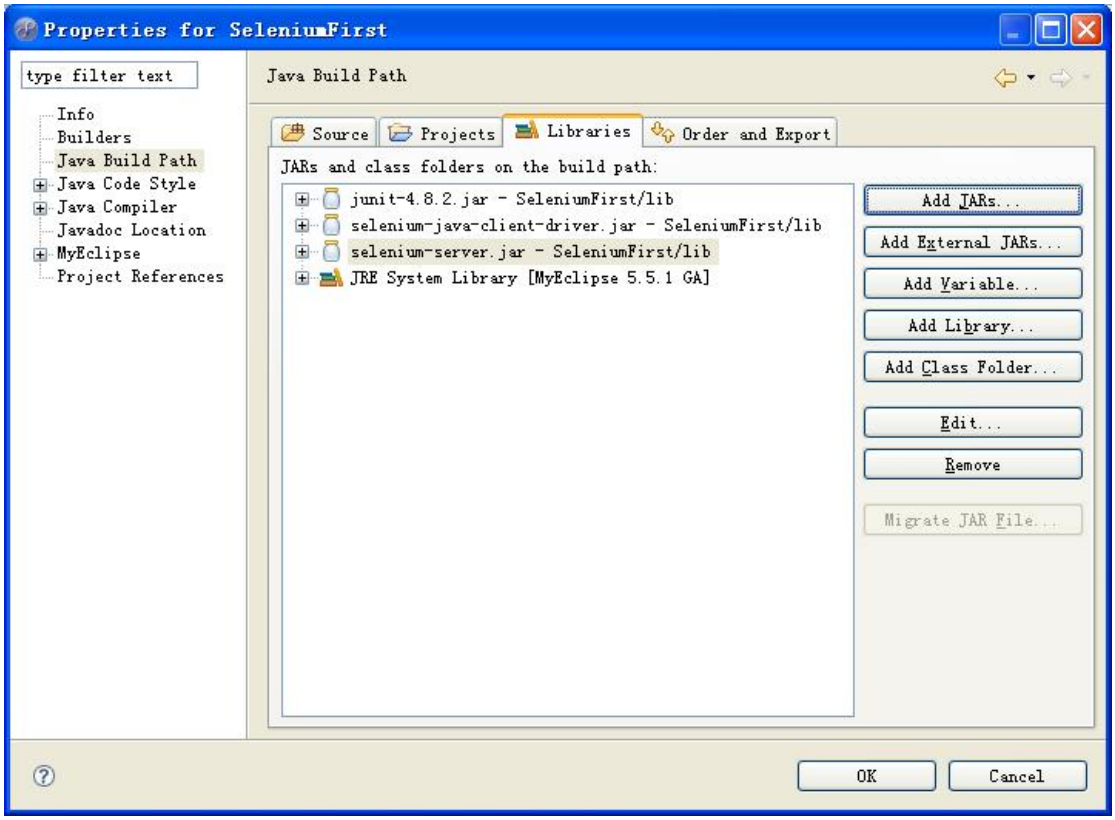
拷贝后的目录结构如图：



4） 配置 Java Bulid Path

在工程名 SeleniumFirst 上右键选择 Properties，打开对话框，

切换至 Libraries,点击 Add JARs...，将工程 lib 文件夹下的 jar 包导入，如下图所示。



5） 拷贝基类

打开路径 \selenium-remote-control-1.0.3\selenium-java-client-driver-1.0.1，找到 selenium-java-client-driver-sources.jar 包，解压后，找到 SeleneseTestBase.java 和 SeleneseTestCase.java 文件，将两文件拷贝至 com.abc.util 下，如图所示：

如果有错误，可以修改包名，引入新的 jar 包。

6） 启动 Selenium 服务

在 Dos 命令行中切换至\selenium-remote-control-1.0.3\selenium-server-1.0.3 下面，运行 *java -jar selenium-server.jar* 命令，启动 Selenium 服务。

2 只有 Junit 的多 case 测试环境

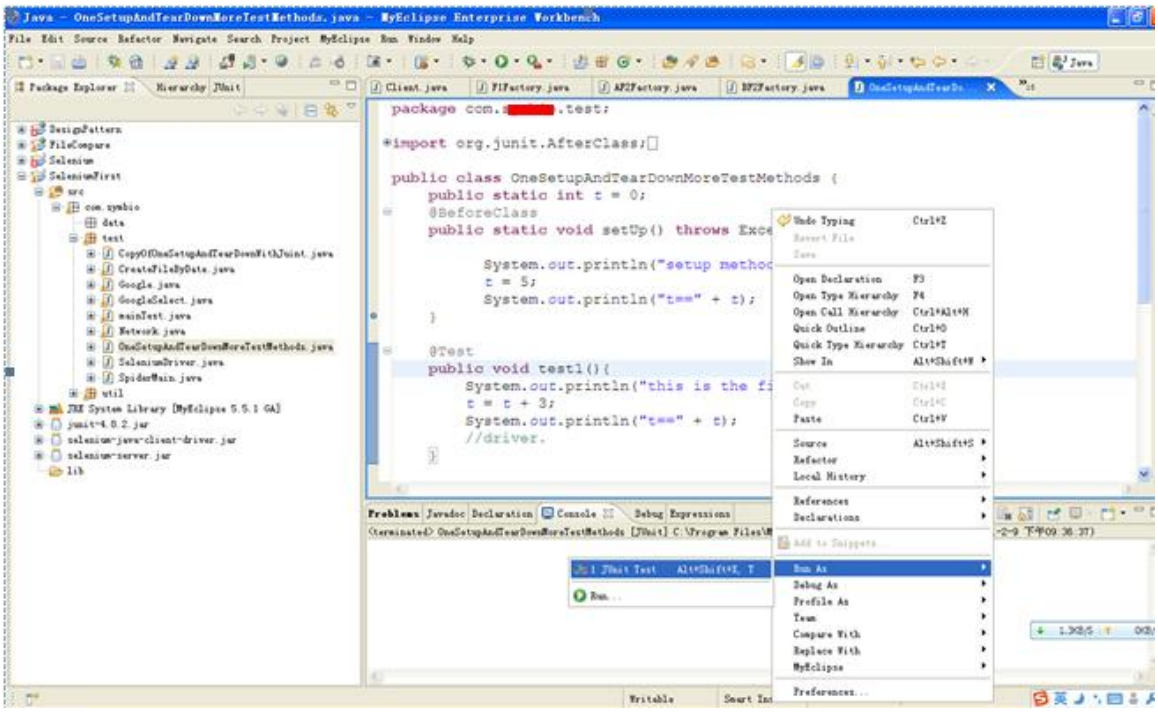
2.1 多个 case 随机执行

在test目录下创建类OneSetupAndTearDownMoreTestMethods.java,该类文件的功能是实现多个case依次执行，而Setup()和TearDown()方法均只执行一次。

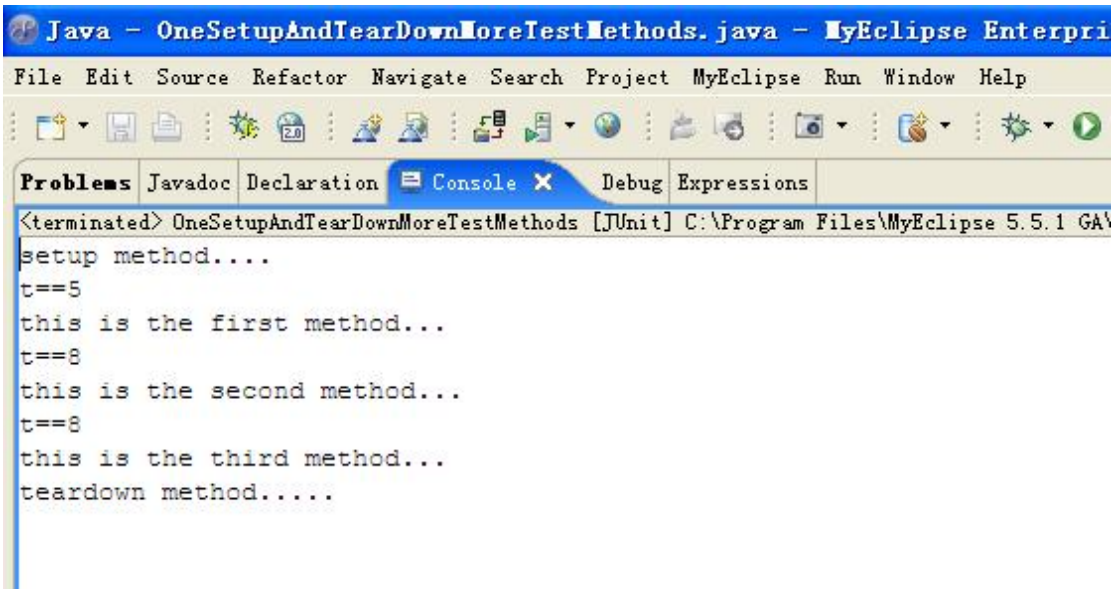
类文件的java源码如下：

```
package com.abc.test;
import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.Test;

public class OneSetupAndTearDownMoreTestMethods {
	public static int t = 0;
	@BeforeClass
	public static void setUp() throws Exception {
		System.out.println("setup method....");
		t = 5;
		System.out.println("t==" + t);
	}

	@Test
	public void test1(){
		System.out.println("this is the first method...");
		t = t + 3;
		System.out.println("t==" + t);
		//driver.
	}

	@Test
	public void test2(){
		System.out.println("this is the second method...");
		System.out.println("t==" + t);
	}

	@Test
	public void test3(){
		System.out.println("this is the third method...");
	}

	@AfterClass
	public static void tearDown() throws Exception {
		System.out.println("teardown method.....");
	}
}
```

在Java视图中，右键**à** Run As **à** Junit Test



执行结果如下：



注意：1、此处使用的是Junit-4.8.2.jar，实现此功能需要使用4.x版本

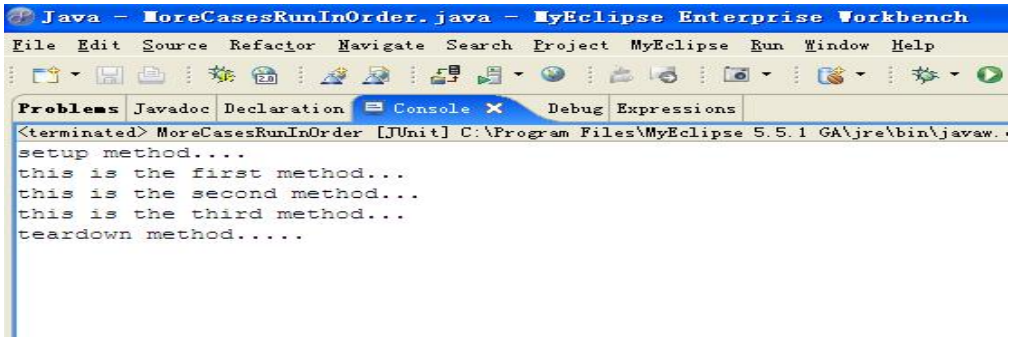
2、上面类文件只保证了可以执行多个case，并在case之前或之后只允许一次setup()和teardown(),但没有保证多个case顺序执行。该问题可以通过以下办法来解决。

2.2 多个 case 顺序执行

创建类文件`MoreCasesRunInOrder.java`，继承`TestCase`类，利用`Test suite`定义各个`case`的执行顺序。源码如下：

```
package com.abc.test;
import junit.framework.Test;
import junit.framework.TestCase;
import junit.framework.TestSuite;
public class MoreCasesRunInOrder extends TestCase {
    public MoreCasesRunInOrder (String name){
         super(name);
    }
  public static void setUpBeforeClass() throws Exception {
      System.out.println("setup method....");
   }
   public void test1(){
      System.out.println("this is the first method...");
   }

   public void test2(){
      System.out.println("this is the second method...");
   }

   public void test3(){
      System.out.println("this is the third method...");
   }
     public static Test suite() {
        TestSuite suite = new TestSuite(
                    "Test for
com.loglogic.logapp.webtest.testcase.users");
     //$JUnit-BEGIN$
     suite.addTest( new MoreCasesRunInOrder( "setUpBeforeClass" ));
     suite.addTest( new MoreCasesRunInOrder( "test1" ));
     suite.addTest( new MoreCasesRunInOrder( "test2" ));
     suite.addTest( new MoreCasesRunInOrder( "test3" ));
     suite.addTest( new
MoreCasesRunInOrder( "tearDownAfterClass" ));
     //$JUnit-END$
           return suite;
     }
   public static void tearDownAfterClass()
     throws Exception {
       System.out.println("teardown method.....");
   }
}
```

执行结果如下：



采用上述方式，用户可以自己根据需要定义方法的执行顺序或有选择的执行一个或多个case。

注意：此处不启动Selenium服务也可以实现上述功能，该类文件主要是依赖Junit执行单元测试

3 基于 Selenium 与 Junit 的多 case 程序设计

3.1 无序的多 case 测试解决方案

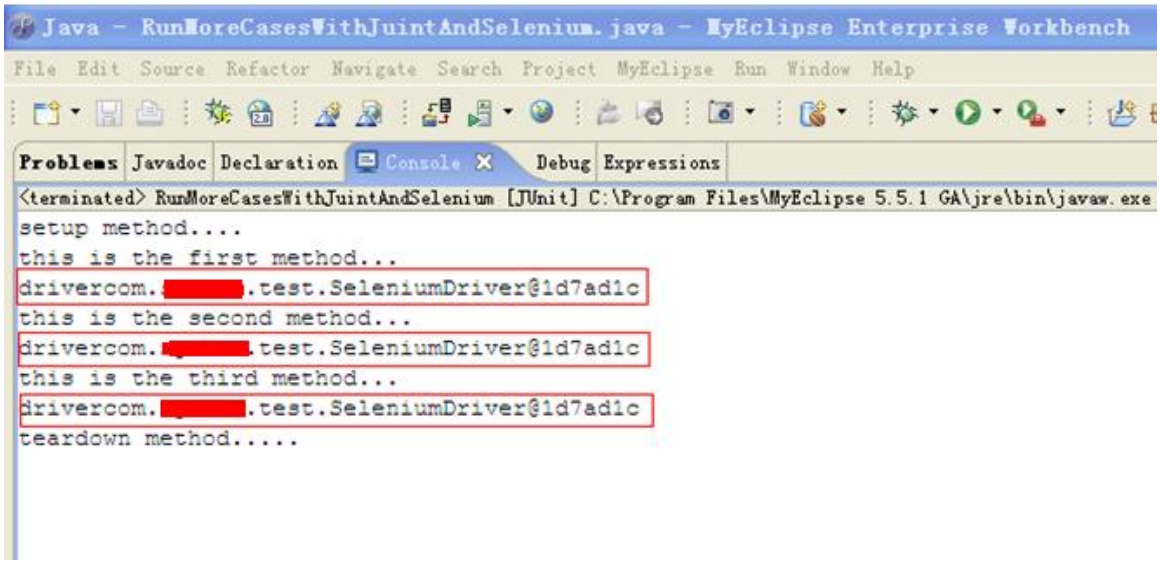
根据自动化测试的需要，对于常用的初始化操作均放在 setup 方法中完成，如打开浏览器。为确保所有的 case 执行前只执行一次 setup 方法，通常需要定义单例模式。示例文件如下：

单例模式 SeleniumDriver.java

```
package com.abc.test;
import com.thoughtworks.selenium.DefaultSelenium;
public class SeleniumDriver {
private DefaultSelenium sel;
private static SeleniumDriver instance;
private SeleniumDriver( String serverIp, int serverPort, String
browserType, String URL ) {
       sel = new DefaultSelenium( serverIp, serverPort,
browserType, URL );
   }
   public static SeleniumDriver getInstance( String serverIp, int
serverPort, String browserType, String URL ) {
       if ( instance == null ) {
          instance = new SeleniumDriver( serverIp, serverPort,
browserType, URL );
       }
       return instance;
   }
}
}
```

```
package com.abc.test;
import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.Test;

public class RunMoreCasesWithJuintAndSelenium {

	public static SeleniumDriver driver;
	@BeforeClass
	public static void setUp() throws Exception {
		System.out.println("setup method....");
		driver = SeleniumDriver.getInstance("127.0.0.1", 44, "*firefox",
"http://www.baidu.com");
	}
	@Test
	public void test1(){
		System.out.println("this is the first method...");
		driver = SeleniumDriver.getInstance("127.0.0.1", 44, "*firefox",
"http://www.baidu.com");
		System.out.println("driver" + driver);
	}
	@Test
	public void test2(){
		System.out.println("this is the second method...");
		System.out.println("driver" + driver);
	}
	@Test
	public void test3(){
		System.out.println("this is the third method...");
		driver = SeleniumDriver.getInstance("127.0.0.1", 44, "*firefox",
"http://www.baidu.com");
		System.out.println("driver" + driver);
	}

	@AfterClass
	public static void tearDown() throws Exception {
		System.out.println("teardown method.....");
	}
}
```

执行结果如下：



3.2 有序的多 case 测试解决方案

可以看出，利用单例模式，每次获取的都是同一个 selenium 对象。这就保证了在运行所有 case 之前只执行一次 setup 操作，在结束后执行一次 teardown 操作，并利用 Selenium 实现多个自动化测试 case 的运行。但这种做法并不能保证所有的 case 均顺序执行，其解决思路同 2.2，详细解决方法如下。

创建类文件 `RunMoreCasesWithJuintAndSeleniumInOrder.java`

```
package com.abc.test;
import junit.framework.Test;
import junit.framework.TestCase;
import junit.framework.TestSuite;
public class RunMoreCasesWithJuintAndSeleniumInOrder extends
TestCase {
    public static SeleniumDriver driver;
    public RunMoreCasesWithJuintAndSeleniumInOrder (String name){
        super(name);
    }
    public static void setUpBeforeClass() throws Exception {
        System.out.println("setup method....");
        driver =  SeleniumDriver.getInstance("127.0.0.1", 44,
"*firefox", "http://www.baidu.com");
    }
    public void test1(){
        System.out.println("this is the first method...");
        driver =  SeleniumDriver.getInstance("127.0.0.1", 44,
"*firefox", "http://www.baidu.com");
        System.out.println("driver" + driver);
    }
```

```
    public void test2(){
        System.out.println("this is the second method...");
        System.out.println("driver" + driver);
    }
    public void test3(){
        System.out.println("this is the third method...");
        driver =  SeleniumDriver.getInstance("127.0.0.1", 44,
"*firefox", "http://www.baidu.com");
        System.out.println("driver" + driver);
    }
     public static Test suite() {
         TestSuite suite = new TestSuite(
                     "Test for
com.loglogic.logapp.webtest.testcase.users");
         //$JUnit-BEGIN$
         suite.addTest( new
RunMoreCasesWithJuintAndSeleniumInOrder( "setUpBeforeClass" ));
         suite.addTest( new
RunMoreCasesWithJuintAndSeleniumInOrder( "test1" ));
         suite.addTest( new
RunMoreCasesWithJuintAndSeleniumInOrder( "test2" ));
         suite.addTest( new
RunMoreCasesWithJuintAndSeleniumInOrder( "test3" ));
         suite.addTest( new
RunMoreCasesWithJuintAndSeleniumInOrder( "tearDownAfterClass" )
);
         //$JUnit-END$
         return suite;
     }
    public static void tearDownAfterClass() throws Exception {
          System.out.println("teardown method.....");
    }
}
```

该类文件调用到的单例模式以及执行结果同 3.1

4 多测试类多用例串联顺序执行

在实际测试项目中，需要将测试类串起来，然后通过一次运行操作，完成所有的测试用例，这将大大提高自动化测试效率，给测试工作带来方便。其详细解决方案如下：

在工程中创建测试类 TestSuite.java,其源码如下：

```
/*********************************************
  说明：本示例用于实现利用 Junit 一次性调用并顺序执行多个 Java 测试类
        其中要调用的类写在 Suitclasses 中即可,但需要创建 TestSuite
（名称可任意定义），即使该类中没有任何内容都可以成功运行，但必须存在该类。
*********************************************/

package com.abc.test;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(Suite.class) // 标记已经指明了这个类是junit，故做为

junit来运行

@Suite.SuiteClasses({

   Test1.class,  // 将要测试的多个类放到这里即可

   Test2.class
})

public class TestSuite { // 类不用写东西即可

}
```

Test1 和 Test2 的具体格式可参考 3.2 的 `RunMoreCasesWithJuintAndSeleniumInOrder.java` 源码

补充说明

不要依赖或假定测试运行的顺序，因为 JUnit 利用 Vector 保存测试方法。所以不同的平台会按不同的顺序从 Vector 中取出测试方法，因此需要采用相应的机制以保证执行顺序，对于常用的情况，本文均做了详细说明，所列源码均在 Eclipse 环境下测试通过。