

网站自动化测试

操作手册

|  |  |  |  |
| --- | --- | --- | --- |
| 文档状态 | [√] 初稿 | 当前版本 |  |
| [ ] 评审通过 |
| [ ] 修改 |
| [ ] 发布 |
| [ ] 作废 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **日期** | **版本** | **修订内容描述** | **作者** | **审核人** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**修 订 记 录**

目录

[1 概述 4](#_Toc491094333)

[**1.1目的** 4](#_Toc491094334)

[**1.2适用范围** 4](#_Toc491094335)

[2 Window下环境搭建(Java + Maven + Selenium + TestNG) 4](#_Toc491094336)

[**2.1 Java环境** 4](#_Toc491094337)

[2.1.1 安装JDK 4](#_Toc491094338)

[2.1.2 配置Java环境变量 4](#_Toc491094339)

[2.1.3 确认环境配置是否正确 5](#_Toc491094340)

[**2.2 Java IDE** 6](#_Toc491094341)

[**2.3 Maven环境** 7](#_Toc491094342)

[2.3.1 安装Maven 8](#_Toc491094343)

[2.3.2 配置Maven环境变量 8](#_Toc491094344)

[2.3.3 确认Maven是否安装成功 8](#_Toc491094345)

[2.3.4 配置Maven本地仓库 8](#_Toc491094346)

[2.3.5 MyEclipse中集成Maven 9](#_Toc491094347)

[**2.4 TestNG** 9](#_Toc491094348)

[2.4.1 在线安装TestNG 9](#_Toc491094349)

[2.4.2 离线安装TestNG 10](#_Toc491094350)

[**2.5 Git** 10](#_Toc491094351)

[2.5.1 安装Git和Git 客户端 10](#_Toc491094352)

[**2.6安装Chrome浏览器** 11](#_Toc491094353)

[3 编写自动化测试用例 11](#_Toc491094354)

[**3.1导入工程** 11](#_Toc491094355)

[**3.2新建测试用例** 13](#_Toc491094356)

[3.2.1 新建Package和Class 13](#_Toc491094357)

[3.2.2 新建TestNG的xml 18](#_Toc491094358)

[**3.3 运行用例和结果显示** 19](#_Toc491094359)

[3.3.1 运行测试用例 19](#_Toc491094360)

[3.3.2 输出结果和html报告 19](#_Toc491094361)

[4 关于pom.xml设置 21](#_Toc491094362)

[**4.1相关插件介绍** 21](#_Toc491094363)

[4.1.1 插件build-helper-maven-plugin 21](#_Toc491094364)

[4.1.2 插件maven-compiler-plugin 22](#_Toc491094365)

[4.1.3 插件maven-surefire-plugin 22](#_Toc491094366)

[**4.2依赖Jar包** 23](#_Toc491094367)

# 1 概述

**1.1目的**

使开发测试人员能够依据步骤搭建开发环境，编写清算所网站UI自动化测试用例（可以从手工用例转化）。

**1.2适用范围**

本文档适用于从事清算所网站自动化测试的所有用户。

# 2 Window下环境搭建(Java + Maven + Selenium + TestNG)

安装JDK、MyEclipse、Maven、TestNG、Git。

**2.1 Java环境**

### 2.1.1 安装JDK

目前网站自动化使用的JDK版本为: jdk1.7.0\_15。

安装jdk-7u51-windows-i586.exe(64位)，使用默认设置即可。

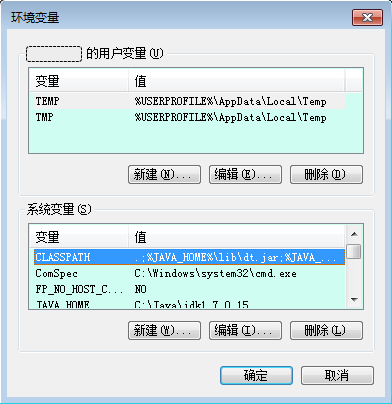
### 2.1.2 配置Java环境变量

1) 计算机→属性→高级系统设置，单击“环境变量”。在“系统变量”栏下单击“新建”，创建新的系统环境变量。

2) 新建->变量名“JAVA\_HOME”，变量值“C:\Program Files\Java\jdk1.7.0\_15”（JDK安装路径，根据自己电脑实际情况设置路径）。

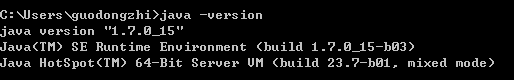
3) 编辑->变量名“Path”，在原变量值的最后面加上“;%JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin”。

4) 新建->变量名“CLASSPATH”,变量值“.;%JAVA\_HOME%\lib;%JAVA\_HOME%\lib\dt.jar;%JAVA\_HOME%\lib\tools.jar”。



### 2.1.3 确认环境配置是否正确

按Windows键+R,输入cmd运行DOS窗口,在控制台分别输入java -version命令和javac命令，出现如下结果表示java环境配置完成。

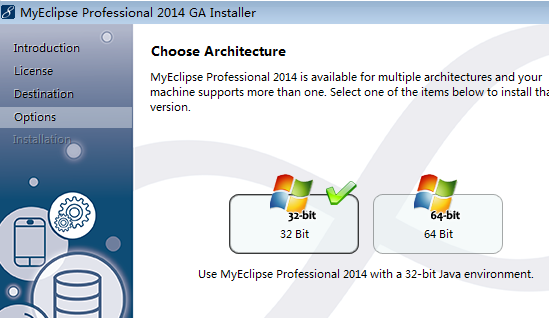




**2.2 Java IDE**

目前网站自动化主要使用MyEclipse，版本为MyEclipse Professional 2014。 双击安装程序, 选择完全安装和32位（或64位）即可。





**2.3 Maven环境**

### 2.3.1 安装Maven

目前网站自动化使用的Maven版本为: Maven 3.3.9。

用Maven安装包或者从群共享目录复制文件和Maven本地仓库。

### 2.3.2 配置Maven环境变量

1) 计算机→属性→高级系统设置，单击“环境变量”。在“系统变量”栏下单击“新建”，创建新的系统环境变量。

2)新建->变量名“MAVEN\_HOME”，变量值“D:\guodz\Maven\3.3.9”（Maven安装路径，根据自己电脑实际情况设置路径）。

3)编辑->变量名“Path”，在原变量值的最后面加上“;%MAVEN\_HOME%\bin”。

### 2.3.3 确认Maven是否安装成功

按Windows键+R,输入cmd运行DOS窗口,在控制台输入mvn -v，如果有maven 版本信息输出则证明配置成功。



### 2.3.4 配置Maven本地仓库

D:\guodz\Maven\3.3.9\conf（我的本地路径）打开maven的配置文件settings.xml，搜索local，修改为本地仓库路径，如

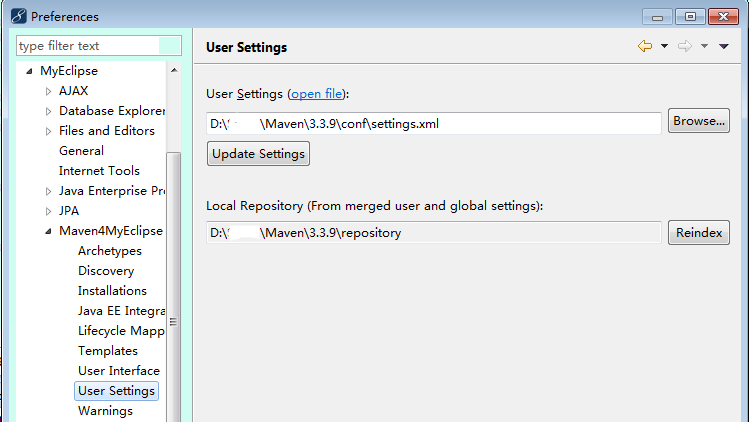
<localRepository>D:\guodz\Maven\3.3.9\repository</localRepository>

### 2.3.5 MyEclipse中集成Maven

1）打开MyEclipse ，点击菜单Window->Preference,

MyEclipse->Maven4MyEclipse->Installations, 点击add，输入安装路径D:\guodz\Maven\3.3.9。

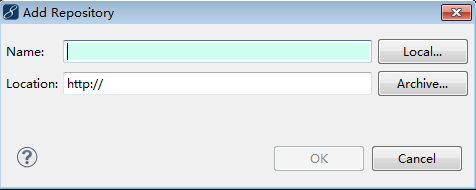
2）在User settings,输入settings.xml的路径

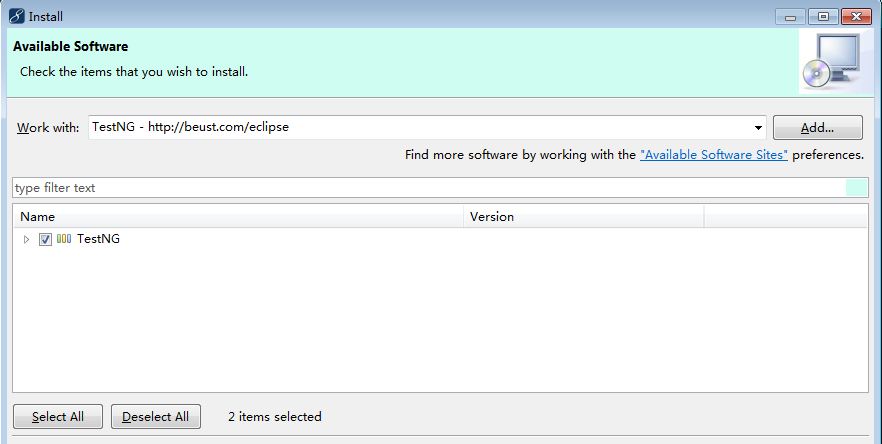


**2.4 TestNG**

### 2.4.1 在线安装TestNG

MyEclipse->Help\_>Install from Site…, 点击add，Name输入TestNG, Location输入<http://beust.com/eclipse>。(需联网), 点击OK，全选并安装，如图所示。





### 2.4.2 离线安装TestNG

可以复制TestNG的离线包，把对应文件夹分别放到MyEclipse的features和plugins目录，重启MyEclipse即可使用。

**2.5 Git**

### 2.5.1 安装Git和Git 客户端

参考“Git简易教程.docx”

项目Git SSH地址：git@10.30.202.61:git/autoTest.git

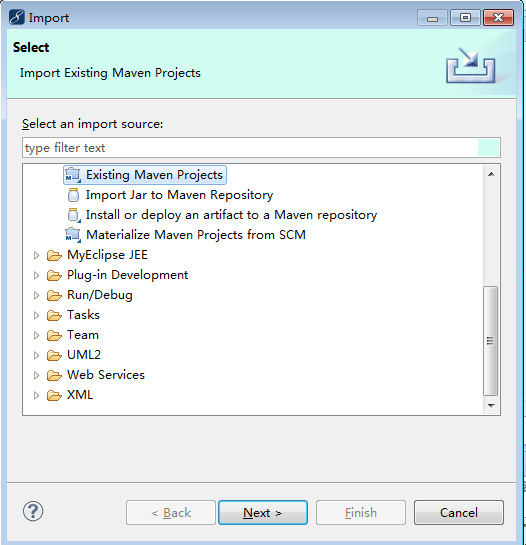
**2.6安装Chrome浏览器**

目前自动化框架使用Chrome进行测试，使用的chromedriver.exe 版本为2.22，可以安装Chrome v49.0.2623.110或v51.0.2704.106。

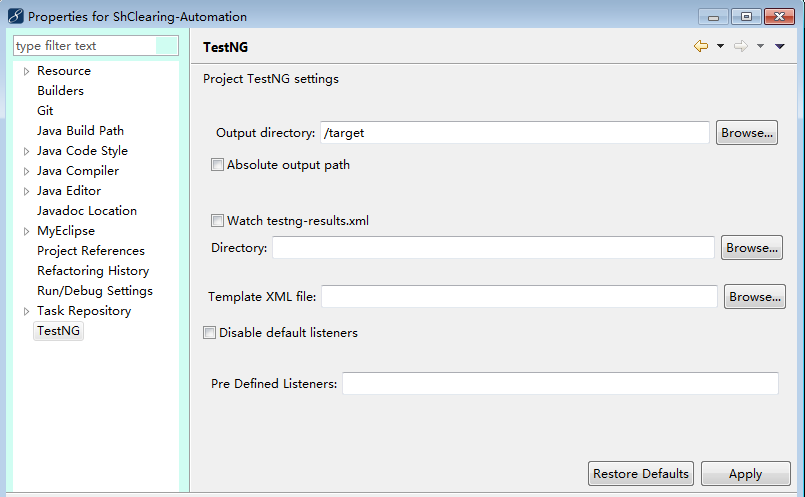
# 3 编写自动化测试用例

**3.1导入工程**

从Git上获取网站自动化测试项目代码，MyEclipse导入该Maven项目。



设置项目TestNG输出目录为/target。

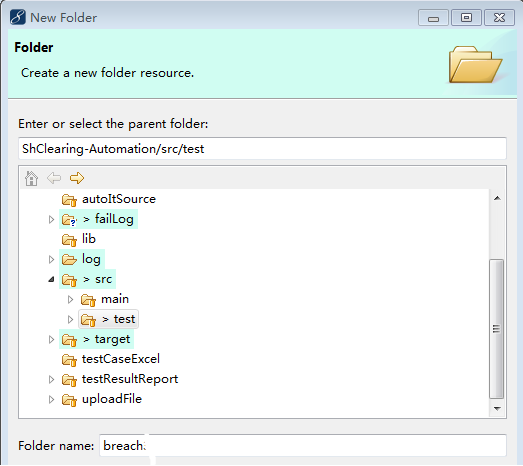


**3.2新建测试用例**

页面元素和一些方法放在页面文件，用例放在脚本文件，做到对象和用例分离。模块使用的公共方法可以放在com.shclearing.automation.common包下的相应模块文件。

### 3.2.1 新建Package和Class

1）在src/test下新建目录，文件夹命名为所测子系统英文， 如breach（违约）。右键，Build Path->Use as source folder，如下所示。





2）新建package，前台命名为com.shclearing.\*\*\*.frontpage， 如com.shclearing.breach.frontpage。后台命名为com.shclearing.\*\*\*.backpage，如果子系统模块比较多，也可前后台各分成几个package。

3）新建页面Class，命名为\*\*\*Page, 如RegisterPage,不需要Main函数，继承ShClearingCommonPage，如：

**public** **class** RegisterPage **extends** ShClearingCommonPage{

**public** RegisterPage(IWebHandler handler){

**this**.handler = handler ;

}

需要引入必须的package，如

**import** org.openqa.selenium.By;

**import** com.shclearing.automation.common.ShClearingCommonPage;

**import** com.shclearing.automation.framework.action.IWebHandler;

4）定义页面元素对象，可以使用id，name，xpath，cssSelector。对于元素，每一行必须有注释，方便以后阅读和修改，如：

**public** By userRegisterBtn = By.*xpath*(".//\*[@id='tb\_user']/div/form/table/tbody/tr[4]/td[2]/a[1]");//注册链接

**public** By labels = By.*xpath*(".//\*[@id='form\_regist']/ul/li/label[1]"); //注册页面的label

**public** By submitBtn = By.*id*("btn\_regist");//注册页面提交按钮

**public** By resetBtn = By.*id*("btn\_reset");//注册页面重置按钮

5）定义testIsStandBy 方法,用于进入所测页面，testIsStandBy调用goToTestPage()，该方法为实际的页面导航和进入页面的点击操作，一般私有，必要时可以返回值。

如下所示：

/\*\*

\* 进入测试页面

\*/

**public** **void** testIsStandBy() **throws** Throwable {

**try**{

goToTestPage();

}**catch**(Exception ex){

exceptionErrorHandle(ex);

}

}

/\*\*

\* this method is used to go to the test page

\* **@throws** Throwable

\*/

**private** String goToTestPage() **throws** Throwable{

String newWin=**null**;

**try** {

navigateToUrl(*breachUrl*);

oriHandle = **this**.handler.getWindowHandle();

eleClickBy(userRegisterBtn);//点击注册

waitPageLoad(10);

newWin=**this**.handler.switchToNewWindowFromDefault(oriHandle);

} **catch** (Exception e) {

exceptionErrorHandle(e);

}

**return** newWin;

}

6）新建脚本package，前台命名为com.shclearing.\*\*\*.front

script， 如com.shclearing.breach. frontscript。后台命名为com.shclearing.\*\*\*.backscript，如com.shclearing.

breach.backscript。

7）新建脚本Class，命名为TestCaseOf\*\*\*,如TestCaseOfAccount，并继承ShClearingTestCaseBase。如下：

**public class** TestCaseOfAccount **extends** ShClearingTestCaseBase {

}

需要引入必要的package，如

**import** org.testng.annotations.Test;

**import** com.shclearing.automation.framework.base.ShClearingTestCaseBase;

**import** com.shclearing.automation.framework.exception.FrameworkException;

**import** com.shclearing.automation.framework.verification.ValidationUtil;

**import** com.shclearing.breach.frontpage.\*;

8）参考用例步骤，编写用例代码，必要时可以调整用例各场景的先后顺序以减少重复操作。如果同样的操作重复出现，可以封装成方法，放在前面的页面文件。验证主要放在用例脚本中。

在用例方法前加注释，添加@Test，组名和延时，如

/\*\*

\* case：用户注册-界面要素检查 \*

\* **@throws** FrameworkException

\* **@author** \*\*\*\*\*\*

\*/

@Test(groups = { "shclearing.breach.front.register" }, timeOut = *MAX\_EXCUTE\_TIME*)

用例结尾有验证，如

ValidationUtil.verifyTrue(result, caseName, **this**.handler);

用例中对元素的常用操作，如单击eleClick，发送文本txtBoxSendValue，获取文本eleTxtGet，可在AbstractWebBaseAction.java中找到。下面是一个元素检查的用例，不需要每行注释，但在比较重要、复杂逻辑的地方(包括Tips)则要添加必要的注释，为以后维护和更新带来便利。再次强调添加注释的重要性，不但为自己也是为了别人阅读方便，尤其是产品更新后导致元素变化，用例变更。

示例如下图所示：



### 3.2.2 新建TestNG的xml

在src/test/xml目录新建xml，复制以下内容（除高亮部分），在<groups>节点下添加相应组名，在classes下添加脚本类名。

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

<suite name=*"shclearing-reports"* parallel=*"false"*>

<parameter name =*"handlerMode"* value=*"Selenium"* />

<parameter name =*"dbType"* value=*"Oracle"* />

<parameter name =*"connectPath"* value=*"dummyconnectstring;username;password"* />

<parameter name =*"browserType"* value=*"chrome"*/>

<parameter name =*"browserPath"* value=*""* />

<test name=*"ShclearingTest"*>

<groups>

<define name=*"shclearing.breach.front"*>

<include name=*"shclearing.breach.front.register"*/>

</define>

<define name=*"all"*>

<include name=*"shclearing.breach.\*.\*"*/>

</define>

<run>

<include name=*"all"*/>

</run>

</groups>

<classes>

<!-- 违约处理 -->

<class name=*"com.shclearing.breach.frontscript.TestCaseOfAccount"* />

</classes>

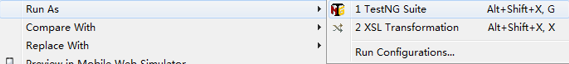
</test>

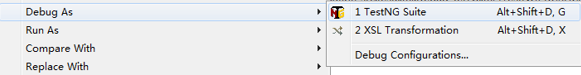
</suite>

**3.3 运行用例和结果显示**

### 3.3.1 运行测试用例

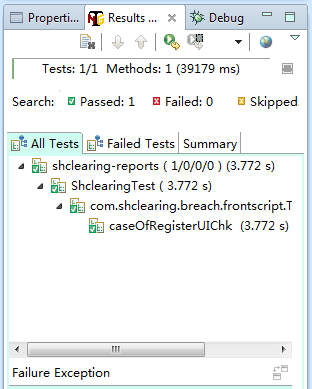
右键上述xml，选择Run As(或者Debug As)->TestNG Suite，可以运行或调试用例。



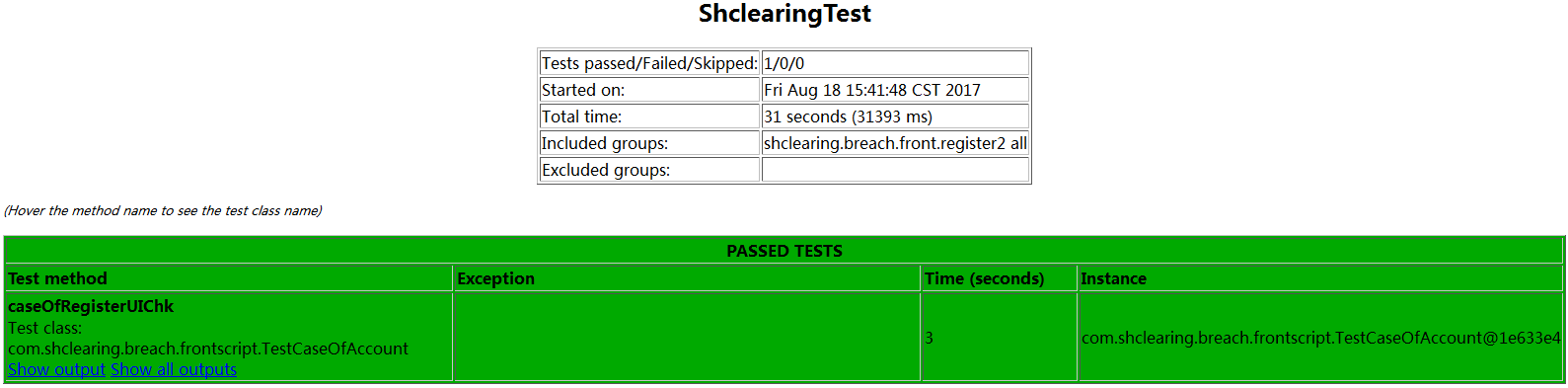


### 3.3.2 输出结果和html报告

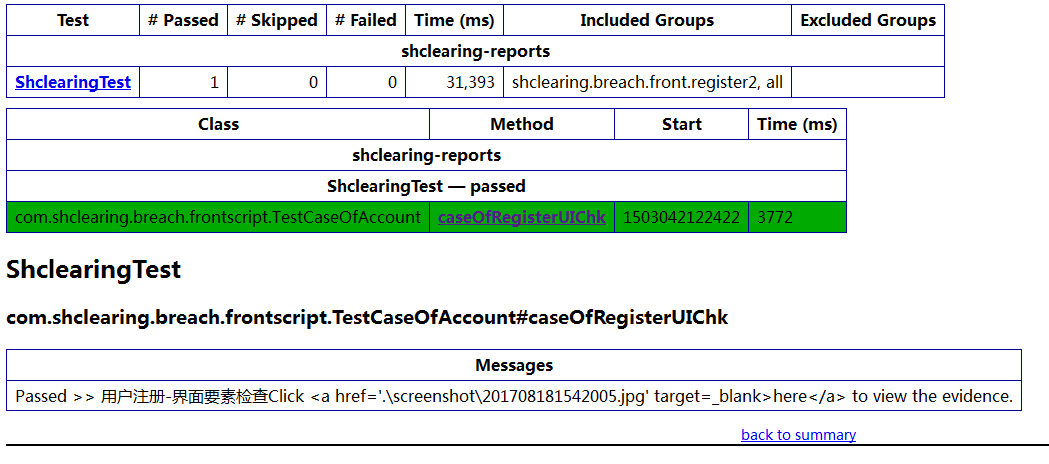
运行完毕，MyEclipse右下角有结果显示，Console窗口也有相应输出。



项目设置了运行完自动打开target/shclearing-reports/ShclearingTest.html。



也可手工打开target/emailable-report.html，查看结果。



\target\shclearing-reports\screenshot下面有验证通过与否时截取的浏览器界面图。

# 4 关于pom.xml设置

目前Jenkins 调用Maven的命令行(mvn clean test)执行，Maven的配置文件pom.xml(实际为模块名.xml),又与TestNG的xml建立了关联。

**4.1相关插件介绍**

### 4.1.1 插件build-helper-maven-plugin

build-helper-maven-plugin插件用于设置多个代码源文件夹。需要把模块的路径放在source节点下，以违约处理为例，如下所示：

<!-- build-helper-maven-plugin, 设置多个源文件夹 -->

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>build-helper-maven-plugin</artifactId>

<version>3.0.0</version>

<executions>

<execution>

<id>add-test-source</id>

<phase>generate-test-sources</phase>

<!-- <phase>initialize</phase> -->

<goals>

<goal>add-test-source</goal>

</goals>

<configuration>

<sources>

<source>src/test/java</source>

<source>src/test/breach</source>

<source>src/test/xml</source>

</sources>

</configuration>

</execution>

</executions>

</plugin>

### 4.1.2 插件maven-compiler-plugin

maven-compiler-plugin插件用来编译源代码，将Java编译器的版本设定为某个版本，如1.7。

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.7</source>

<target>1.7</target>

</configuration>

</plugin>

### 4.1.3 插件maven-surefire-plugin

maven-surefire-plugin插件用来关联TestNG.xml，suiteXmlFile节点指定相应xml的路径（统一命名为ShClearing-AutoTest.xml）。

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId> <version>2.19.1</version>

<configuration>

<suiteXmlFiles>

<suiteXmlFile> ${basedir}/src/test/xml/breach/ShClearing-AutoTest.xml</suiteXmlFile>

</suiteXmlFiles>

</configuration>

</plugin>

**4.2依赖Jar包**

项目依赖的Jar包定义在pom.xml中， dependency节点下，可以根据用例需要添加，以下是处理csv文件的Jar包。

<dependencies>

<dependency>

<groupId>net.sourceforge.javacsv</groupId>

<artifactId>javacsv</artifactId>

<version>2.0</version>

<scope>system</scope>

<systemPath>${basedir}/lib/javacsv.jar</systemPath>

</dependency>

……

</dependencies>