**android app自动化测试框架Appium资料整理**

**1 自动化测试**

1.1 自动化测试简介

自动化测试是把以人为驱动的测试行为转化为机器执行的一种过程。从广义上来讲，一切通过工具（程序）的方式来代替或辅助手工测试的行为都可以看做自动化测试。从狭义上来讲，自动化测试就是通过工具记录或编写脚本的方式模拟手工测试的过程，通过回放或运行脚本来执行测试用例，从而代替人工对系统的功能进行验证。

1.2 自动化测试的优点

1）提高效率：通过运行自动化测试脚本，就可以在夜间、午休等时间进行测试用例回归，实现无人值守测试，大大提高了测试效率。

2）避免重复工作：每发布一个新的版本，其中大部分功能和界面都和上一个版本相似或完全相同，这部分功能就可以用自动化测试，避免人工重复劳动。

3）避免人为出错：例如忘了执行某些用例，用例执行步骤出错，输入数据出错等。

4）保证每次测试的一致性和可重复性：由于每次自动化测试运行的脚本是相同的，所以每次执行的测试具有一致性，这一点手工测试是很难做到的；而且，由于自动化测试的一致性，很容易发现被测软件的任何改变。

5）节省人力资源：减少人工，降低成本。

6）克服手工测试的局限性：许多死锁、资源冲突、多线程等有关的问题 ，通过手工测试很难捕捉到；系统压力、性能测试，以及需要模拟大数据或大并发用户等各种测试场景，很难通过手工测试执行；系统可靠性测试，需要模拟系统长时间运行，以验证系统能否稳定运行，难以通过手工测试执行。

**1.3 哪些测试需要用自动化测试解决**

并不是所有项目都适合自动化测试，适合自动化测试的项目一般具有如下特点：首先项目周期要长，需求不会频繁变更；其次系统中多数对象要可以被识别。

你不能指望自动化测试去帮你发现新的bug，自动化测试本身是不具备想象力的（相对于手工测试）。它的优势在于反复迭代，它的价值产出在于长期的回归测试，以保证被测产品在版本更新时的稳定性。

一般来说，具备如下特点的测试可以考虑使用自动化测试：

1.大量机械的、重复性的回归测试；

2.结果的正确性不依赖主观判断的测试；

3.需要模拟大量数据、大量并发量的测试；

4.需要不间断执行的测试；

5.需要短时间内完成的大量测试用例执行（比如完整的功能回归测试）；

测试趋势应该就是往自动化测试去发展，这个方向是对的，但现实中往往会因为自动化框架的限制等原因，造成问题的漏测，因此适当的进行手工测试也是很有必要的。

**2 自动化测试工具**

**2.1 移动端自动化测试工具的比较**

比较流行的移动应用自动化测试工具的比较：



Native apps（原生应用）：仅使用Android或iOS的标准SDK编写的应用；

Web apps（移动浏览器应用）：用移动平台的浏览器访问的应用；

Hybrid apps（混合应用）：把一个基于webview实现的功能进行包装的应用。

**3 Appium**

**3.1 Appium简介**

Appium是一个开源的自动化测试工具，其支持iOS和Android平台上的原生应用、基于移动浏览器的应用以及混合应用的测试。

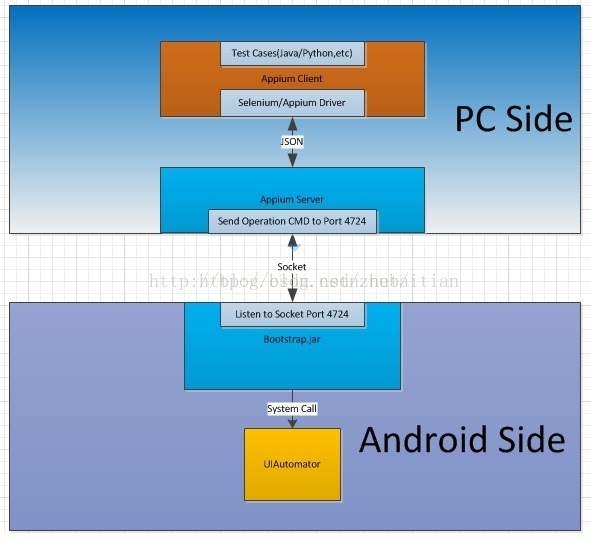
1）Appium的设计理念

1. Appium使用的是移动平台供应商（iOS的UIAutomation和Android的Instrumentation及UIAutomator)提供的标准自动化测试框架，这就使得你不需要重新编译或者修改你的app；

2. Appium将不同的移动平台供应商的自动化测试框架进行一次更高层次的封装，做成一套统一的API（WebDriverAPI）提供出来，使得你不需要使用一门特定的语言和一个特定的框架去实现和运行你的测试；

3. Appium是开源的；

2）Appium原理框架图（网上图片）



从上图可以看出：

Appium的核心是基于C/S架构的网络服务器。它接受客户端（测试

用例实现端）过来的连接，监听（客户端发送过来的）命令，在移动设备上运行命令，然后把包含命令运行结果的HTTP响应包发送回客户端。

Appium使用C/S的架构事实上为我们打开了很多可能性：我们可以在任何支持http 客户端API的语言上面实现我们的测试代码；我们可以把服务器端放在跟我们的测试运行机器完全不一样机器上（例如实现云端测试）。

3）相关组件解释

1.Appium client：

使用WebDriver库或者基于WebDriver的扩展库（统称Appium客户端库）编写的测试用例。

2.Appium server：

它是个http服务器，它专门接收从Appium client发送过来的命令，同时，它也是bootstrap客户端。它接收到客户端的命令后，需要把这些命令发送给目标机器的bootstrap，bootrap接收到命令并转换为UIAtuomator可以执行的命令。

3.Selenium driver：

即WebDriver，WebDriver是一个用来进行web自动化测试的工具，但是它没有和任何测试框架进行绑定。它提供了一系列的API对web应用的中界面元素进行定位、模拟用户操作的API，底层利用JOSN通过HTTP与服务器进行交互。Appium扩展了WebDriver API，使它更适用于移动应用程序的测试。

4.Bootstrap：

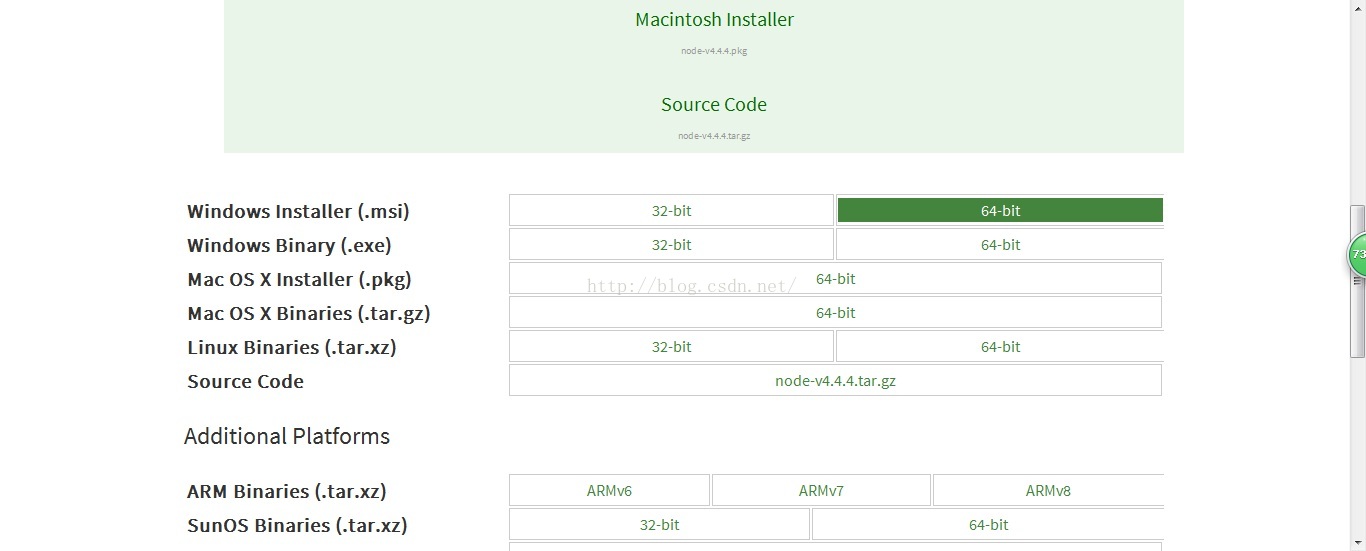
Bootstrap是Appium运行在安卓目标测试机器上的一个UiAutomator测试脚本，该脚本所做的事情是在目标机器开启一个socket服务器来把Appium从PC端过来的命令发送给UiAutomator来执行处理。

**3.2 Appium测试环境搭建（Windows）**

Appium测试环境的搭建步骤：

1）下载并安装nodejs

Nodejs的地址：https://nodejs.org/en/download/



2）安装JDK

安装过程略，但请注意如下事项。

1.如果选择64bits的Eclipse，则必须安装64bits的Java，否则运行Eclipse会出现如下问题：



2.如果导入Android工程，注意JDK版本的选择，有可能出现如下错误：Android requires compiler compliance level5.0 or 6.0. Found '1.8' instead. Please use Android Tools > Fix ProjectProperties

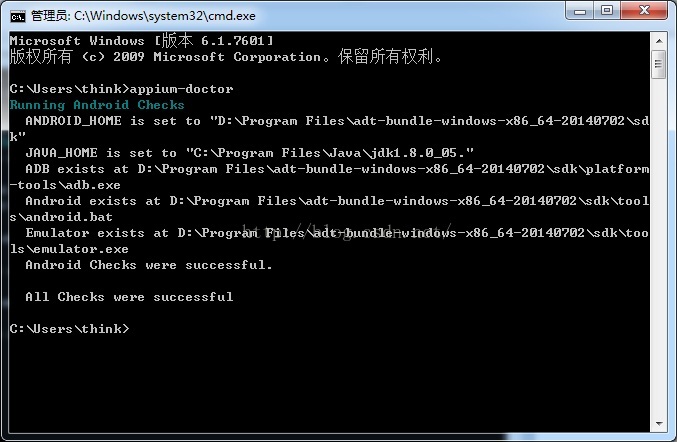
3）安装Android SDK（可选）

4）安装Maven（可选）

5）下载并安装Appium

Appium的下载地址：http://appium.io/，但是一般不会成功。推荐国内下载地址：https://testerhome.com/topics/680

安装完成之后，将Appium安装路径\node\_modules\.bin添加到path中，然后在cmd中运行appium-doctor，如果所有的依赖检查可以通过，表示Appium安装成功。



**4 基于Java的测试框架的选择**

考虑到大家使用的开发语言主要以Java为主，因此测试框架考虑基于Java的JUnit4和TestNG。

二者之间的比较请参考：

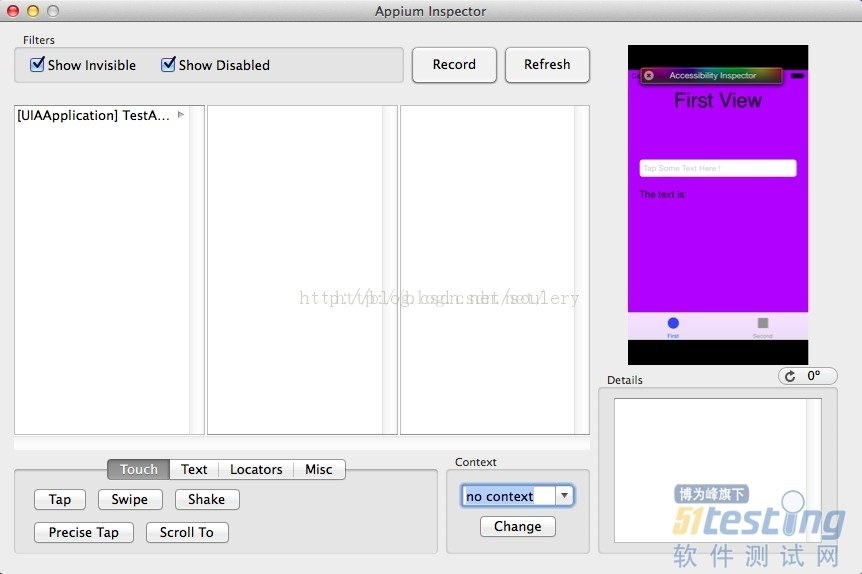
这里引用该文章的总结性言论作为我们选择TestNG这一测试框架的理由：“当我们做完所有特性的对比以后，我建议使用 TestNG 作为 Java 项目的主要单元测试框架，因为 TestNG 在参数化测试、依赖测试以及套件测试（组）方面功能更加强大。TestNG 意味着高级的测试和复杂的集成测试。它更加的灵活，特别是对大的套件测试。另外，TestNG 也涵盖了 JUnit4 的全部功能。那就没有任何理由使用 JUnit了。”

另外，由于TestNG和JUnit存在非常多的相似性，二者的代码常常不需要过多转换就可以移植，熟悉JUnit的开发人员也不必纠结需要重新学习TestNG；而且，TestNG支持XML文件的配置，非常灵活。

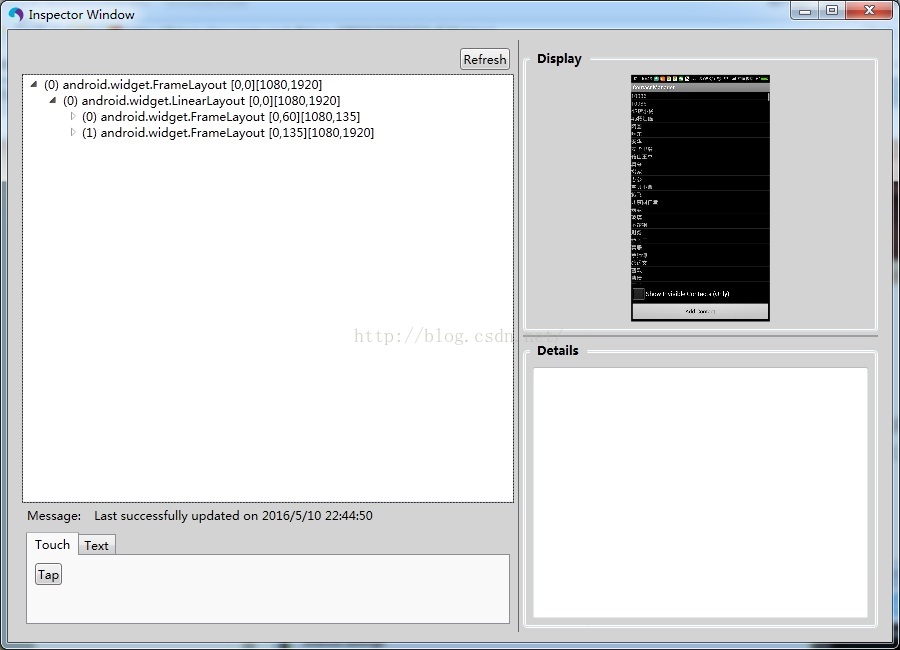
**5 脚本录制**

**5.1 Appium的脚本录制**

打开Appium.exe，点击Inspector按钮进行录制，Mac版本是具备录制功能的（网上图片）：



但是Window版本没有录制工具，如下图：



**5.2 如何自己实现脚本录制**

可以以考虑使用AccessibilityService，监听用户输入，在onAccessibilityEvent(AccessibilityEvent)方法中解析Event，例如eventtype，className，text，resourceId，index等信息，根据Event类型和其携带的控件属性生成测试代码，然后在修改成Appium或者其它自动化测试框架需要的脚本。

**6 TestNG的编译和使用**

**6.1 编译**

TestNG的官网上没有现成的jar包供下载，但是提供了在Eclipse上使用的插件（更新的地址：http://beust.com/eclipse）。如果需要获取jar包，可以获取源代码并编译。

关于编译的步骤，官网上有说明：

|  |
| --- |
| TestNGis also hosted on GitHub, where you can download the source and build thedistribution yourself:  $ gitclone git://github.com/cbeust/testng.git  $ cdtestng  $./build-with-gradle  You willthen find the jar file in the target directory |

其中，build-with-gradle步骤也会去下载gradle工具。

**6.2 基本使用**

新建Java project，导入TestNG包，编写测试用例类，在该类中编写测试用例，编写方法与使用Junit相似。

运行测试TestNG测试用例：右击project->run as->TestNG Test

详细的使用过程请参考：

**6.3 TestNG测试用例的命令行运行**

TestNG的官方文档有命令行运行的描述。

在项目中增加testng.xml文件，编译后，进入工程目录，输入命令：

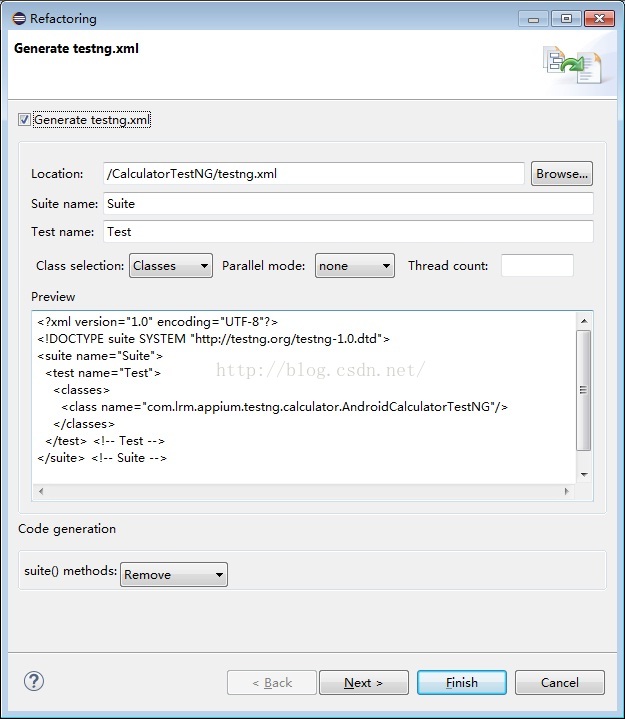
|  |
| --- |
| javaorg.testng.TestNG testng.xml |

注：使用前需将testNG的jar包（以及selenium和selenium server需要的jar包）设置在classpath中，上面命令中的org.testng.TestNG是testNG的jar包中的一个类。

**6.4 使用testNG.xml文件管理测试用例**

1.生成testng.xml文件

右击Project->TestNG->Convert to TestNG，进入如下界面：



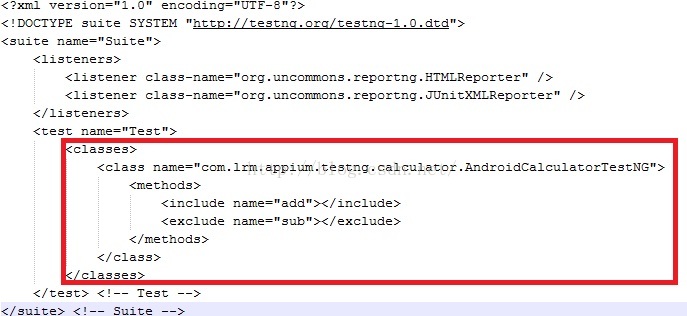
2.测试用例的管理

1. testng.xml文件结构

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE suite SYSTEM"http://testng.org/testng-1.0.dtd">  <suite name="suitename" junit="false"verbose="3" parallel="false" thread-count="5"  configfailurepolicy="skip"annotations="javadoc"  time-out="10000"skipfailedinvocationcounts="true" data-provider-thread-count="5"  object-factory="classname"allow-return-values="true"> <!-- name参数为必须 -->  <suite-files>  <suite-filepath="/path/to/suitefile1"></suite-file> <!-- path参数为必须 -->  <suite-filepath="/path/to/suitefile2"></suite-file>  </suite-files>  <parametername="par1" value="value1"></parameter> <!-- name, value参数为必须 -->  <parametername="par2" value="value2"></parameter>  <method-selectors>  <method-selector>  <selector-classname="classname" priority="1"></selector-class><!-- name参数为必须 -->  <scriptlanguage="java"></script> <!-- language参数为必须 -->  </method-selector>  </method-selectors>  <testname="testename" junit="false" verbose="3"parallel="false" thread-count="5"annotations="javadoc"  time-out="10000"enabled="true" skipfailedinvocationcounts="true"  preserve-order="true"allow-return-values="true"> <!-- name参数为必须 -->  <parametername="par1" value="value1"></parameter> <!-- name, value参数为必须 -->  <parametername="par2" value="value2"></parameter>  <!-- groups:前提是需要存在classes的组，否则所有方法不被运行 -->  <groups>  <!--define：定义新的组名 -->  <definename="group\_name"> <!-- name参数为必须 -->  <!--group\_name1为测试类中方法的Annotation定义的组名 -->  <includename="group\_name1" description=""invocation-numbers="" /> <!-- name参数为必须 -->  <includename="group\_name2" description=""invocation-numbers="" />  </define>  <run>  <!-- group\_name为测试类中定义的组名或者xml中定义的组名 -->  <includename="group\_name" /> <!-- name参数为必须 -->  <excludename="" /> <!-- name参数为必须 -->  </run>  <dependencies>  <groupname="" depends-on=""></group> <!--name,depends-on均为参数为必须 -->  <groupname="" depends-on=""></group>  </dependencies>  </groups>  <!-- classes的用法，classes中包含类名，类名底下可以包含方法名或排除方法名 -->  <classes>  <classname="classname"> <!-- name参数为必须 -->  <methods>  <parametername="par3" value="value3"></parameter>  <includename="method\_name" description=""invocation-numbers=""></include>  <excludename="method\_name"></exclude>  </methods>  <methods></methods>  </class>  </classes>  <packages>  <packagename="" /> <!-- name参数为必须 -->  <packagename="">  <includename="" description=""invocation-numbers=""></include>  <excludename=""></exclude>  </package>  </packages>  <listeners>  <listenerclass-name="classname1" /> <!-- name参数为必须 -->  <listenerclass-name="classname2" />  </listeners>  </test>  <test>  </test>  </suite> |

2. 使用classes、class标签进行用例管理

在class标签中通过name属性声明实现了测试方法的类，一个classes标签可以包含多个class标签。在class标签中，还可以通过include和exclude声明该类中的被包含或者被排除在测试用例中的方法。

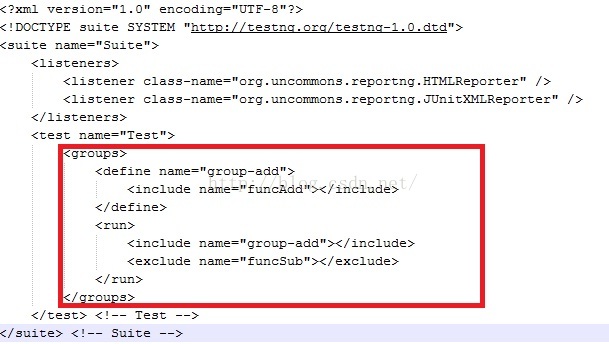


3. 使用groups进行用例管理

可以指定test方法属于哪些group，一个方法可以属于多个分组。方法属于哪个分组可以在Java代码中通过Annotation指定，例如：

|  |
| --- |
| @Test(groups = {"group\_name1","group\_name2"})  Public void func()  {  } |

指定test方法所属的分组之后，就可以在testng.xml文件中指定要运行的分组，testng.xml文件修改如下：



其中，define标签表示在testng.xml文件中根据存在的分组重新定义分组名称；run标签定义要运行的分组。

注：

TestNG的group标签导致的@BeforeMethod方法无效，这样，如果group里面的方法对@BeforeMethod方法有依赖，可能会导致错误。

解决方法：

将@BeforeMethod替换为@BeforeMethod(alwaysRun = true)，或者将@BeforeMethod方法也放在group中。

**7 测试报告**

**7.1 TestNG的测试报告**

参考：

**7.2 ReportNG的编译**

ReportNG的官网上已经申明不再进行维护，当前稳定版本为1.1.4，源代码已经托管到Github，因此可以根据需要进行定制。

该项目为Maven工程，没有尝试进行编译。但是Eclipse IDE环境已经自带了Maven插件，可以进行项目的create和build。

注：Maven项目中很多jar包是需要从网络下载的。

**7.3 ReportNG的使用**

使用步骤：

1. 作为jar包加入工程的build路径；

2. 在testng.xml中的suite标签中添加listener，如下：

|  |
| --- |
| <listeners>  <listener class-name="org.uncommons.reportng.HTMLReporter"/>  <listener class-name="org.uncommons.reportng.JUnitXMLReporter"/>  </listeners> |

注：

1. reportNG需要guice的支持，下载地址：http://google-guice.googlecode.com/files/guice-3.0.zip

2. reportNG显示中文时会有乱码，解决方法参考：

3. 测试用例跑完之后，可以在test-output目录的html文件夹找到index.html测试报告。

ReportNG 目前有很多缺点，但是由于提供了源代码，可以根据需要自己定制。例如，在测试报告中，出错的用例的高亮显示没法做，需要自己去修改ReportNG源码才能实现；另外，你可以通过代码修改静态的html模板去实现报告DIY定制。

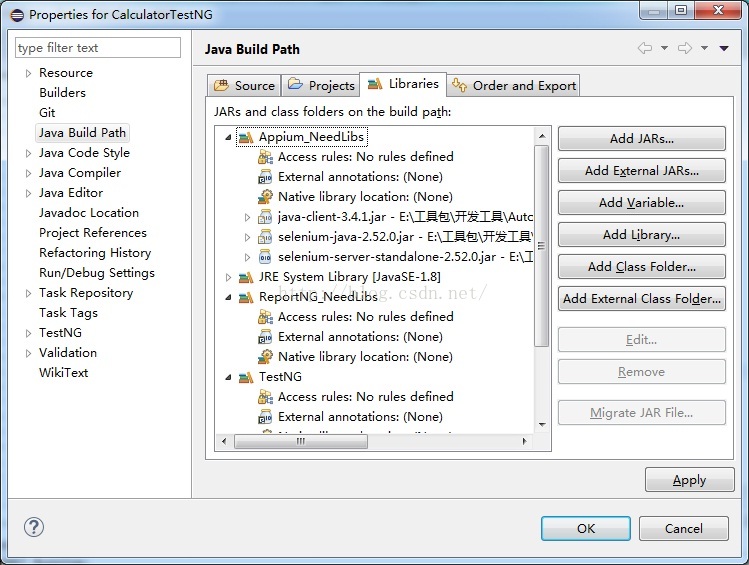
8 示例

了解了Appium、TestNG之后，就可以尝试编写测试程序了。文档中测试代码在GitHub的地址：git@github.com:GitHub-inthewind/CalculatorTestNG.git，是一个基于S2的Calculator编写的测试用例程序。

8.1 编写测试用例

1）创建Java工程

2）导入Libs，包括Selenium client库，SeleniumServer库，Appium Client库（可以在Appium的官网上下载，地址：http://appium.io/），testNG，reportNG库；



3）在Java工程目录下新建apps目录，将要测试的APK放在该目录下；

4）创建class，编写测试用例

|  |
| --- |
| public class AndroidCalculatorTestNG {  privateAppiumDriver<AndroidElement> driver;  @BeforeSuite(alwaysRun = true)  public void setUp() throws Exception {  System.out.println("setUp entry");  // setup appium  File classpathRoot = new File(System.getProperty("user.dir"));  File appDir = new File(classpathRoot, "apps");  File app = new File(appDir, "Calculator.apk");  DesiredCapabilities capabilities= new DesiredCapabilities();  //project can run,even if the settings of this following statement is wrong  capabilities.setCapability("deviceName","0a733f0b");  capabilities.setCapability("platformVersion", "4.4.4");  // setthe APK's path  capabilities.setCapability("app", app.getAbsolutePath());  capabilities.setCapability("appPackage", "com.android.calculator2");  capabilities.setCapability("appActivity", "com.android.calculator2.Calculator");  driver = new AndroidDriver<>(new URL("http://127.0.0.1:4723/wd/hub"), capabilities);  System.out.println("setUp end");  }  @AfterSuite(alwaysRun = true)  public void tearDown() throws Exception {  System.out.println("tearDown entry");  driver.quit();  System.out.println("tearDown end");  }  @Test(groups = {"funcAdd"}, invocationCount = 1)  public void add(){  WebElement num1 = driver.findElementById("digit\_1");  num1.click();  WebElement add = driver.findElementById("op\_add");  add.click();  WebElement num2 = driver.findElementById("digit\_2");  num2.click();  WebElement eq = driver.findElementById("eq");  eq.click();  }  @Test(groups = {"funcSub"})  public void sub(){  WebElement num1 = driver.findElementById("digit\_1");  num1.click();  WebElement add = driver.findElementById("op\_sub");  add.click();  WebElement num2 = driver.findElementById("digit\_2");  num2.click();  WebElement eq = driver.findElementById("eq");  eq.click();  }  } |

5）创建testng.xml文件

testng.xml文件的创建步骤请参考TestNG的编译和使用章节。

如果需要生成reportNG格式的测试报告，需要对testng.xml文件进行修改，如下：

|  |
| --- |
| <?xml version="1.0"encoding="UTF-8"?>  <!DOCTYPE suite SYSTEM "http://testng.org  /testng-1.0.dtd">  <suite name="Suite">  <!—生成reportNG格式的测试报告-->  <listeners>  <listener class-name="org.uncommons.reportng  .HTMLReporter"/>  <listener class-name="org.uncommons.reportng.  JUnitXMLReporter"/>  </listeners>  <test name="Test">  <!—对测试用例进行管理-->  <groups>  <define name="group-add">  <include name="funcAdd"></include>  </define>  <run>  <include name="group-add"></include>  <exclude name="funcSub"></exclude>  </run>  </groups>  <classes>  <class name="com.lrm.appium.testng.calculator.AndroidCalculatorTestNG">  </class>  </classes>  </test> <!-- Test -->  </suite> <!-- Suite --> |

6）创建Ant的构建文件build.xml

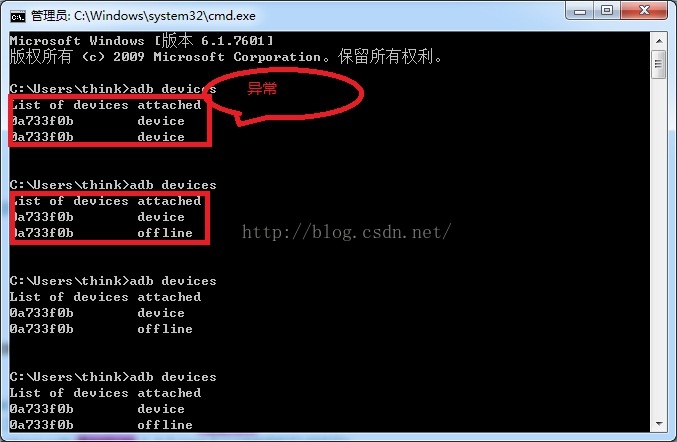
Build文件的创建请参考Ant的使用章节。

8.2 运行测试用例

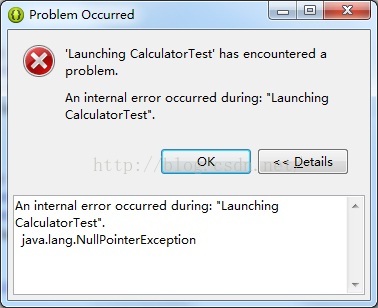
将手机与PC通过USB连接，在cmd中，输入appium，启动Appium；然后选择测试项目，右键->Run As->TestNG Test。

8.3 异常解决

1）有时候会出现Appium测试用例程序不能运行的情况，提示执行安装appium settings程序失败，此时可以看看连接的devices是否在线



2）运行出现NullPointException



问题排查：

查看WorkSpace下的.metadata文件夹下的.log文件，发现有org.eclipse.jdt.junit相关的错误，继续在Eclipse下查找这个jdt.junit相关的插件， Eclipse下缺少org.eclipse.jdt.junit相关的插件，造成用JUnit运行测试用例的时候，报空指针的错误。但是Eclipse自带的JUnit组件确实是安装了的，有可能是下载的Android开发套件删除了某些东西导致，于是从Eclipse官网上下载Eclipse版本自己配置环境，编译运行OK。