# **Rest Assured+TestNg实现数据驱动的接口测试**

### 参考、摘录自 https://www.jianshu.com/p/20f82c961632

### **引言**

笔者之前一直使用Jmeter做接口测试，也围绕Jmeter做了一些功能集成，比如：生成excle结果文件、数据库断言、自动提交缺陷、自动更新案例执行结果至Testlink等。虽说Jmeter简单易上手，但大批量执行测试案例时，响应时间较长，这对向来追求测试效率的笔者而言，无疑是心头之痛。  
 很早就听说过Rest Assured、TestNg两大框架，也看过一些相关的文章，但苦于各种原因，一直都是浅尝辄止。这两天心血来潮，尝试使用Rest Assured+TestNg来实现数据驱动的接口测试，谁知不“尝（试）”则已，一“尝”惊人，实在是接口测试人员的福音。

### **框架介绍**

* Rest Assured

REST Assured是一个可以简化HTTP Builder顶层，基于REST服务的测试过程的Java DSL（针对某一领域，具有受限表达性的一种计算机程序设计语言）。它支持发起POST,GET,PUT,DELETE,OPTIONS,PATCH和HEAD请求，并且可以用来验证和校对这些请求的响应信息。

* TestNg

TestNG is a testing framework designed to simplify a broad range of testing needs, from unit testing (testing a class in isolation of the others) to integration testing (testing entire systems made of several classes, several packages and even several external frameworks, such as application servers).

* ReportNg

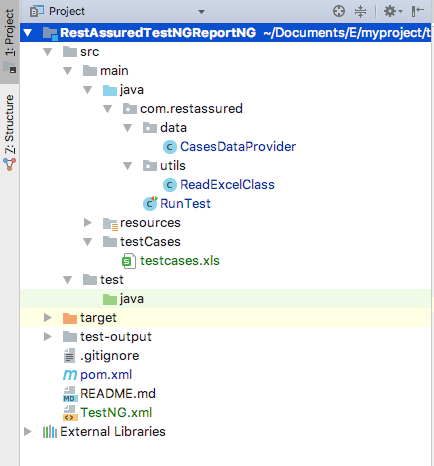
ReportNG is a simple HTML reporting plug-in for the TestNG unit-testing framework.

### **实现功能**

* 读取excel测试案例数据。
* 发送请求。
* 断言。
* 生成测试报告。

### **实现步骤**

#### **1、代码结构及案例模板**



代码结构

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| caseNo | testPoint | preResult | YorN | tableCheck | api | version | acctType |
| case001 | 接口 | true | Y | N | this/is/interface | 1 | 1 |
| case002 | 参数错误 | true | N | Y |  | 1 | 1 |
| case003 |  | true | Y |  |  |  |  |

案例模板（部分字段预留后续使用）

#### **2、新建maven项目并配置pom.xml**

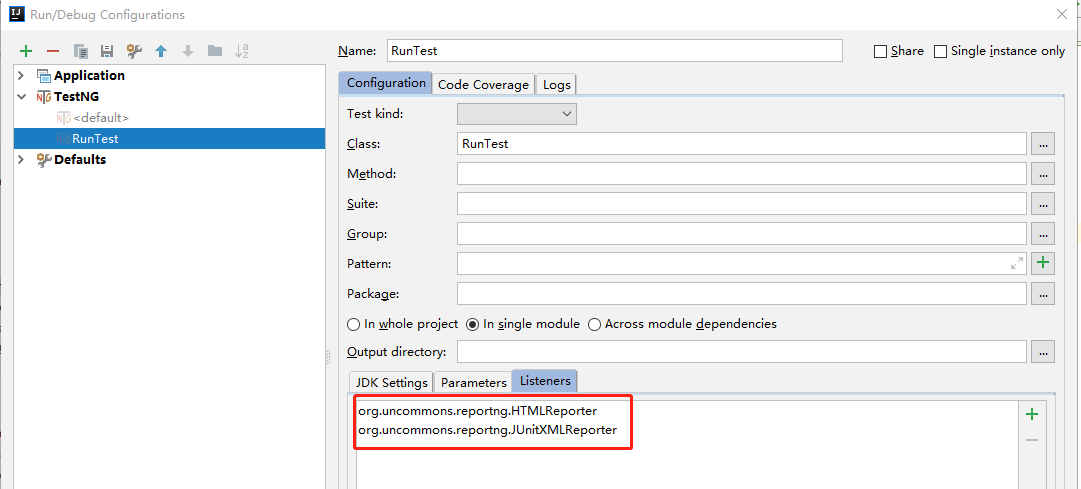
|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**modelVersion**>4.0.0</**modelVersion**>   <**groupId**>com.wjxRestAssuredTestNGReportNG</**groupId**>  <**artifactId**>restassuredtestngreportng</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>   <**dependencies**>  <**dependency**>  <**groupId**>io.rest-assured</**groupId**>  <**artifactId**>rest-assured</**artifactId**>  <**version**>3.3.0</**version**>  *<!--<scope>test</scope>-->* </**dependency**>   *<!-- https://mvnrepository.com/artifact/org.testng/testng -->* <**dependency**>  <**groupId**>org.testng</**groupId**>  <**artifactId**>testng</**artifactId**>  <**version**>6.14.3</**version**>  *<!--<scope>test</scope>-->* </**dependency**>   *<!--JExcelApi is a java library which provides the ability to read, write, and modify Microsoft Excel spreadsheets.-->  <!-- https://mvnrepository.com/artifact/net.sourceforge.jexcelapi/jxl -->* <**dependency**>  <**groupId**>net.sourceforge.jexcelapi</**groupId**>  <**artifactId**>jxl</**artifactId**>  <**version**>2.6.12</**version**>  </**dependency**>   *<!-- 依赖reportNg 关联testNg-->  <!--ReportNG is a simple HTML reporting plug-in for the TestNG unit-testing framework-->  <!-- https://mvnrepository.com/artifact/org.uncommons/reportng -->* <**dependency**>  <**groupId**>org.uncommons</**groupId**>  <**artifactId**>reportng</**artifactId**>  <**version**>1.1.4</**version**>  *<!--<scope>test</scope>-->* <**exclusions**>  <**exclusion**>  <**groupId**>org.testng</**groupId**>  <**artifactId**>testng</**artifactId**>  </**exclusion**>  </**exclusions**>  </**dependency**>   *<!--Google Guice Core Library Google Guice 一个轻量级的依赖注入框架 -->  <!-- https://mvnrepository.com/artifact/com.google.inject/guice -->* <**dependency**>  <**groupId**>com.google.inject</**groupId**>  <**artifactId**>guice</**artifactId**>  <**version**>4.2.3</**version**>  </**dependency**>   </**dependencies**>   <**build**>  <**plugins**>  *<!-- 添加插件 关联testNg.xml 什么作用？？？？？？？？ -->* <**plugin**>  <**groupId**>org.apache.maven.plugins</**groupId**>  <**artifactId**>maven-surefire-plugin</**artifactId**>  <**version**>2.17</**version**>  <**configuration**>  <**suiteXmlFiles**>  <**suiteXmlFile**>xmlfile/TESTNG.xml</**suiteXmlFile**>  </**suiteXmlFiles**>  </**configuration**>  </**plugin**>  *<!-- 添加插件,添加ReportNg的监听器，修改最后的TestNg的报告 -->  <!--Maven Surefire MOJO in maven-surefire-plugin 通过该插件来运行maven项目中的测试用例-->* <**plugin**>  <**groupId**>org.apache.maven.plugins</**groupId**>  <**artifactId**>maven-surefire-plugin</**artifactId**>  <**version**>2.5</**version**>  <**configuration**>  <**properties**>  <**property**>  <**name**>usedefaultlisteners</**name**>  <**value**>false</**value**>  </**property**>  <**property**>  <**name**>listener</**name**>  <**value**>org.uncommons.reportng.HTMLReporter,org.uncommons.reportng.JUnitXMLReporter</**value**>  </**property**>  </**properties**>  <**workingDirectory**>target/</**workingDirectory**> *<!--什么作用？？？？？？？？-->  <!--<workingDirectory>test-output/html/</workingDirectory>-->* <**forkMode**>always</**forkMode**>  </**configuration**>  </**plugin**>  </**plugins**>  </**build**> </**project**> |

**3、配置TestNG.xml文件**

配置TestNG.xml文件，设置监听信息

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* **<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd"*>*** <**suite name="Default Suite" allow-return-values="true"**>   <**listeners**>  <**listener class-name="org.uncommons.reportng.HTMLReporter"**/>  <**listener class-name="org.uncommons.reportng.JUnitXMLReporter"**/>  </**listeners**>   <**test verbose="2" preserve-order="true" name="Test"**>  <**classes**>  <**class name="com.restassured.RunTest"**/>  <**class name="com.restassured.data.CasesDataProvider"**/>  <**class name="com.restassured.utils.ReadExcelClass"**/>  </**classes**>  </**test**> </**suite**> |

#### **4、配置ReportNg监听**



reportNg监听

#### **5、读取案例数据**

由于TestNg的@DataProvider注释返回的是二维数组，所以需读取excel案例数据保存到一个二维数组。

|  |
| --- |
| **package** com.restassured.utils;  **import** jxl.Sheet; **import** jxl.Workbook; **import** jxl.read.biff.BiffException;  **import** java.io.FileInputStream; **import** java.io.IOException; **import** java.io.InputStream;  **public class** ReadExcelClass {   **public static** Object[][] readCases(String filePath) **throws** IOException, BiffException {   InputStream inputStream = **new** FileInputStream(filePath);  Workbook rwb = Workbook.*getWorkbook*(inputStream);   Sheet sheet = rwb.getSheet(0);  **int** rsRows = sheet.getRows(); *// 获取总行数* **int** rsColums = sheet.getColumns();*//获取总列数   // 统计需要执行的案例数* **int** countY = 0;  **for** (**int** i = 1; i < rsRows; i++) {  **if**(sheet.getCell(3, i).getContents().equals(**"Y"**)) *//统计需要执行的案例数* countY++;  }  System.***out***.println(**"需要执行的案例数 : "**+countY);   Object[][] cases = **new** Object[countY][rsColums];   **int** x =0;  **for** (**int** i = 1; i < rsRows; i++) {  **if**(sheet.getCell(3, i).getContents().equals(**"Y"**)){ *//执行标识为“Y”才记录数组* **for** (**int** j = 0; j < rsColums; j++) {  cases[x][j] = sheet.getCell(j, i).getContents();  }  x++;  }  }  **return** cases;  } } |

TestNg的@Test传参有多种方法，具体可百度，本例子使用@DataProvider来传参。

|  |
| --- |
| **package** com.restassured.data;  **import** com.restassured.utils.ReadExcelClass; **import** jxl.read.biff.BiffException; **import** org.testng.annotations.DataProvider; **import** org.testng.annotations.Test;  **import** java.io.File; **import** java.io.IOException; **import** java.util.Date;  **public class** CasesDataProvider {   @DataProvider(name = **"casesProvider"**)  **public static** Object[][] caseProvider(){  *// 获取项目路径* String projectDirectory = System.*getProperty*(**"user.dir"**) ;  System.***out***.println(**"projectDirectory : "**+projectDirectory);  String filePath = projectDirectory + File.***separator*** + **"src"** + File.***separator*** + **"main"** + File.***separator*** + **"testCases"** + File.***separator*** + **"testcases.xls"**;  *// 获取用例路径* System.***out***.println(**"filePath : "**+filePath);  *// 获取对象数组* Object[][] cases = **null**;  **try** {  cases = ReadExcelClass.*readCases*(filePath);  } **catch** (IOException e) {  e.printStackTrace();  } **catch** (BiffException e) {  e.printStackTrace();  }  *// 输出对象数组的内容* **int** rowNum = cases.**length**;  **int** colNum=0;  **if**(rowNum>0)  colNum = cases[0].**length**;  **for**(**int** i=0;i<rowNum;i++){  **for**(**int** j=0;j<colNum;j++){ *// if(cases[i][j]!=null)* System.***out***.print(**"["**+i+**"]["**+j+**"]"**+**" = "**+cases[i][j]+**" "**); *// else System.out.print("["+i+"]["+j+"]"+" ");* }  }   *// 为了与下一个模块隔开，输出下面的空行* System.***out***.println();   **return** cases;  }  } |

#### **6、执行案例**

|  |
| --- |
| **package** com.restassured;  **import** com.restassured.data.CasesDataProvider; **import** io.restassured.RestAssured; **import** io.restassured.parsing.Parser; **import** io.restassured.path.json.JsonPath; **import** io.restassured.response.Response; **import** org.testng.Assert; **import** org.testng.annotations.BeforeClass; **import** org.testng.annotations.Test;   **public class** RunTest {   @BeforeClass  **public void** setUp() {  *// 请求IP* RestAssured.*baseURI* = **"https://\*\*\*\*\*\*"**;  RestAssured.*basePath* = **"/"**; *// RestAssured.port = 8080;  //在多数场景下，SSL能顺利运转，这多亏于HTTP Builder和HTTP Client。如果服务端使用了无效的证书，然而有些例子下还是会出错。最简单的方法是使用"relaxed HTTPs validation",也可以为所有的请求静态定义这个配置：* RestAssured.*useRelaxedHTTPSValidation*();  *// 通过注册预置的解析器来解析现在不支持的内容类型* RestAssured.*registerParser*(**"text/plain"**, Parser.***JSON***);  }   @Test(dataProvider = **"casesProvider"**, dataProviderClass = CasesDataProvider.**class**)  **public void** runCases(String caseNo, String testPoit, String preResult, String YorN, String tableCheck, String api, String version,String acctType) { *// public void runCases(String caseNo, String testPoit, String preResult, String YorN, String tableCheck, String appId, String merchantId, String api, String version, String phone, String bizTransaction, String acctType) { // // String bodyString = "{\n" + // "\t\"appId\":\"" + appId + "\",\n" + // "\t\"api\":\"" + api + "\",\n" + // "\t\"data\":{\n" + // "\t\t\"merchantId\":\"" + merchantId + "\",\n" + // "\t\t\"bizTransaction\":\"" + bizTransaction + "\",\n" + // "\t\t\"phone\":\"" + phone + "\",\n" + // "\t\t\"acctType\":\"" + acctType + "\"\n" + // "\t\t},\n" + // "\t\"version\":\"" + version + "\"\n" + // "}\n"; // System.out.println("bodyString : "+bodyString); // Response response = RestAssured.given() // .contentType("application/json;charset=UTF-8") // .request() // .body(bodyString) // .post();* System.***out***.println(**"当前测试用例编号是 : "**+caseNo);  System.***out***.println(**"测试点是 : "**+testPoit);  Response response = RestAssured.*given*() *// .contentType("application/json;charset=UTF-8")* .when()  .get(api);  System.***out***.println(**"下面是格式化后的响应报文 : "**);  response.prettyPrint(); *//格式化响应报文   //断言* String json = response.asString();  JsonPath jp = **new** JsonPath(json);  *// if(response.statusCode() == 200){ //请求成功* **if**(jp.getInt(**"code"**)==1){  System.***out***.println(**"请求成功，下面进行判断"**);  Assert.*assertEquals*(jp.get(**"success"**).toString(),preResult);  }**else**{  System.***out***.println(**"请求失败，下面进行判断"**);  Assert.*assertEquals*(jp.get(**"success"**).toString(),preResult);  }  *// 为了与下一个用例隔开，输出下面的空行* System.***out***.println();  } } |

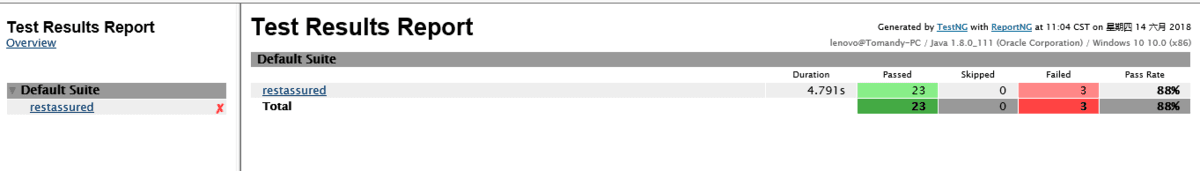
**7、执行测试用例**

先在类RunTest方法setUp中设置RestAssured.baseURI的值，然后在 项目下的 src/main/java/testCases/testcases.xls文件中添加对应的接口地址。  
 执行时，直接右键执行 RunTest类文件，或者右键执行TestNG.xml文件即可。

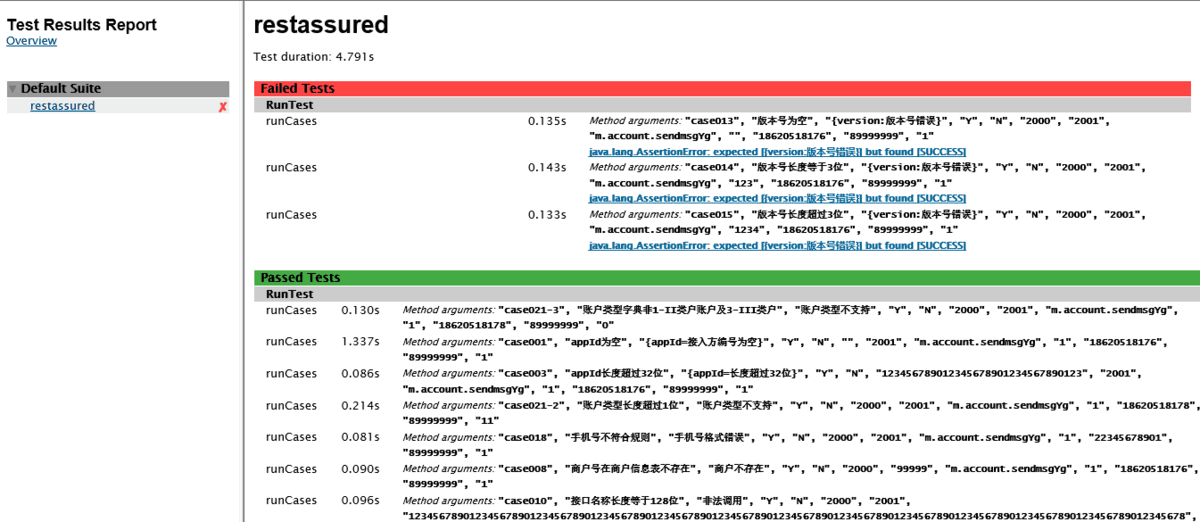
#### **8、查看测试报告**

ReportNg测试报告支持自定义，后续再做探究。

在项目的根目录下，测试报告为 项目名/test-output/html/index.html



测试报告



测试报告

### **展望**

以上只是Rest Assured+TestNg强大功能的冰山一角，后续笔者再慢慢摸索。另外，既然迈出了这一步，那怎么也得展望一下未来，笔者打算后续搞个接口测试平台玩玩。

### **参考资料**

[Rest Assured使用指南](https://testerhome.com/topics/7060" \t "/Users/finup/Documents\\x/_blank)  
[TestNg报告优化](https://blog.csdn.net/a542551042/article/details/46729585" \t "/Users/finup/Documents\\x/_blank)