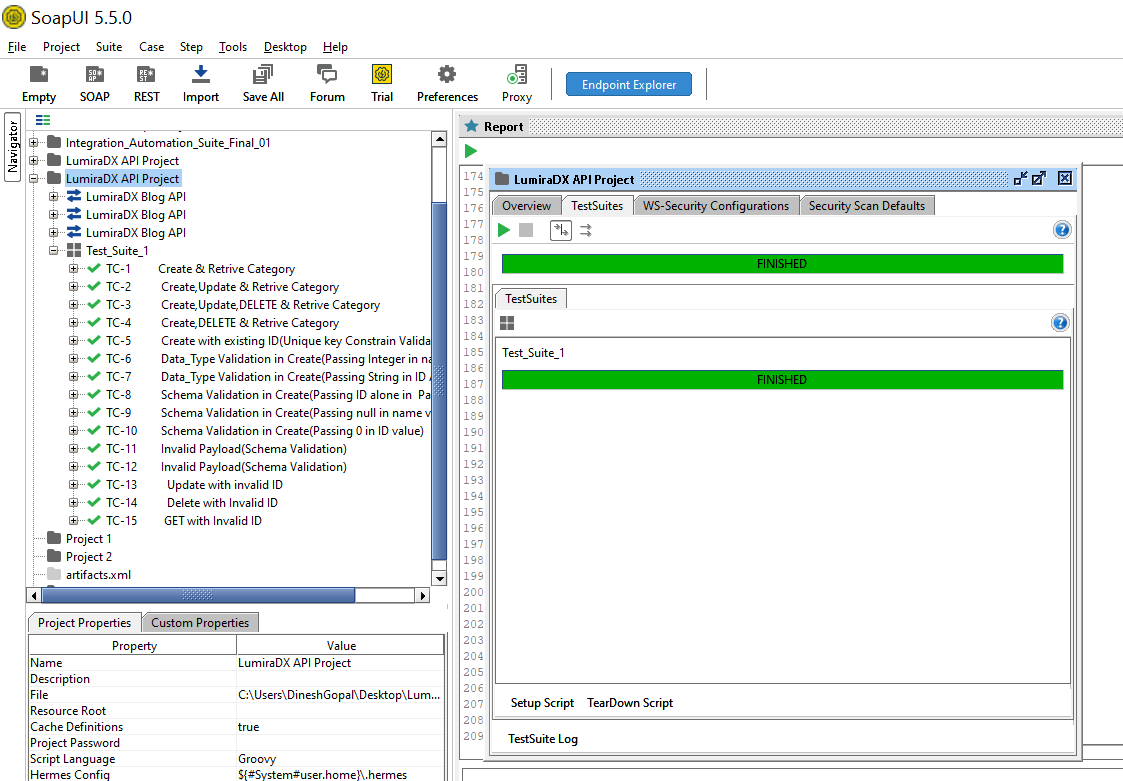
**Automated Test Script**

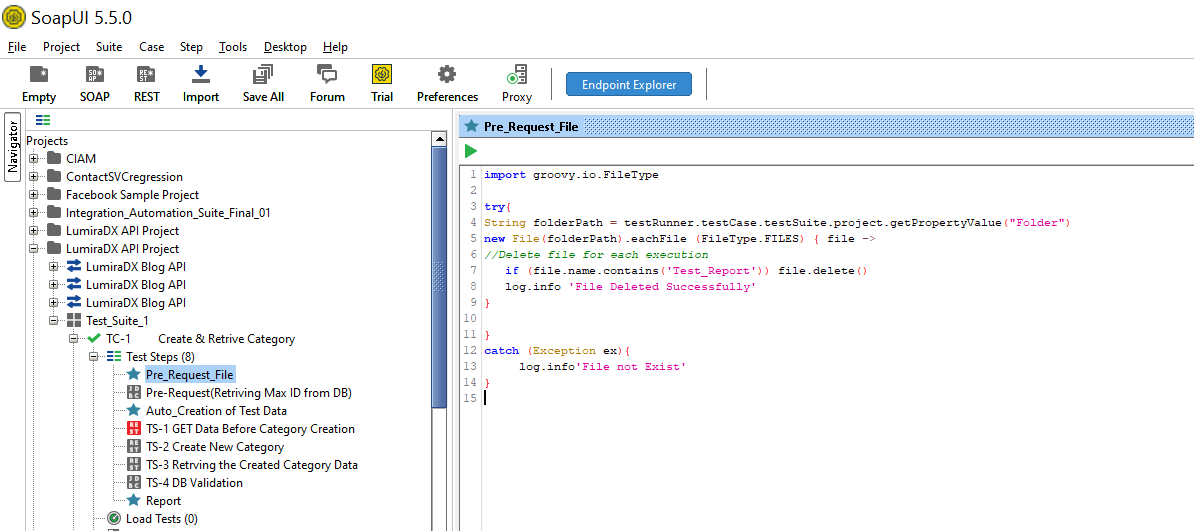
**For each Iteration of test execution, Data is auto created and passed into attributes via data parameterization using groovy Script**

**Note: All the automation test scripts are available in test automation file (LumiraDX API Project)**

**Test Suite Execution:**



**Pre-Request, old Report File Deletion:**



**import groovy.io.FileType**

**try{**

**String folderPath = testRunner.testCase.testSuite.project.getPropertyValue("Folder")**

**new File(folderPath).eachFile (FileType.FILES) { file ->**

**//Delete file for each execution**

**if (file.name.contains('Test\_Report')) file.delete()**

**log.info 'File Deleted Successfully'**

**}**

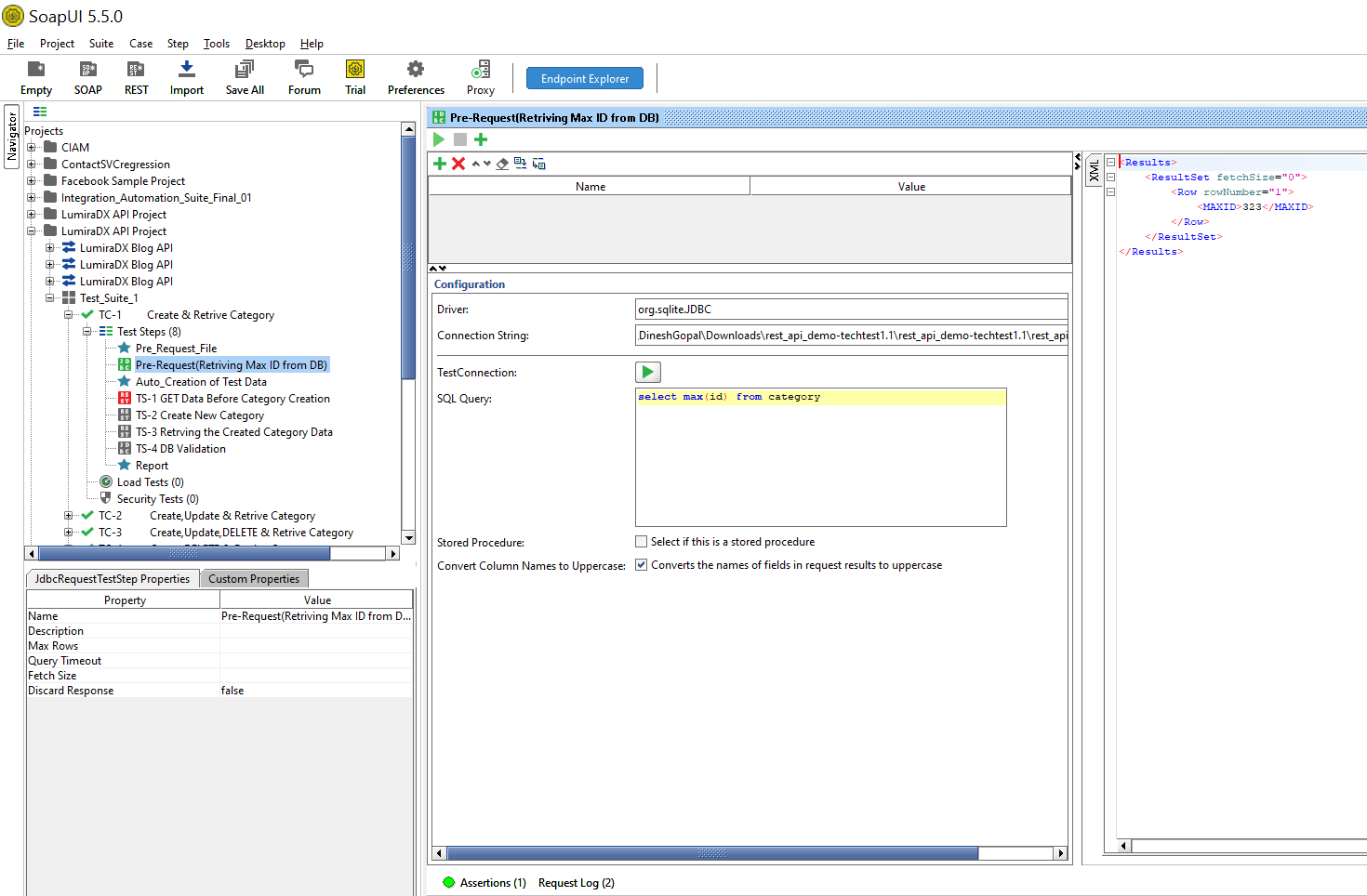
**}**

**catch (Exception ex){**

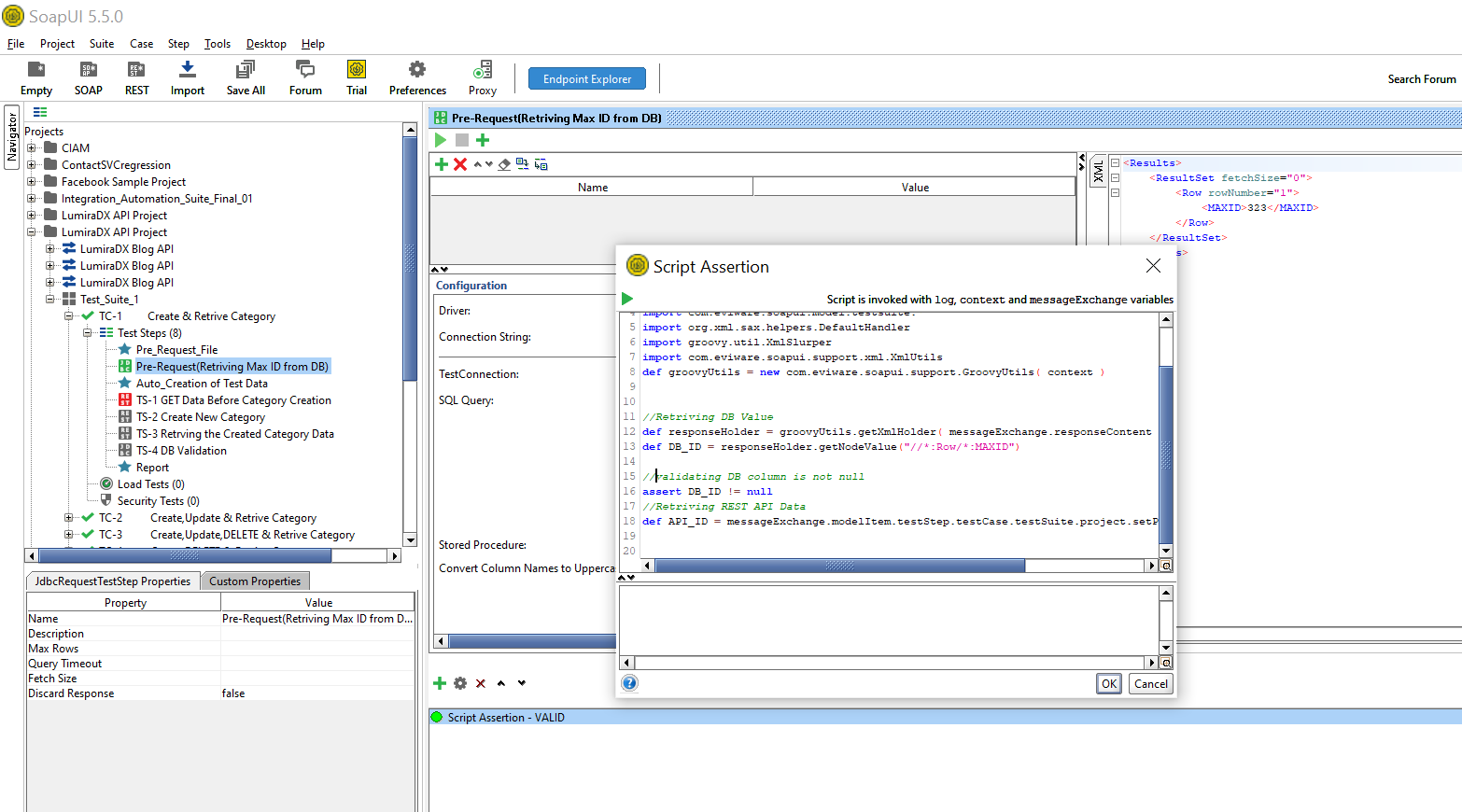
**log.info'File not Exist'**

**}**

**2. Retrieve max ID from category table using JDBC connection with DB, for incremental auto ID attribute data creation**



**Validating DB data using groovy Script:**



**import groovy.lang.\***

**import groovy.util.\***

**import com.eviware.soapui.model.testsuite.TestRunner;**

**import com.eviware.soapui.model.testsuite.\*;import org.xml.sax.helpers.DefaultHandler; import groovy.util.XmlSlurper import com.eviware.soapui.support.xml.XmlUtils**

**def groovyUtils = new com.eviware.soapui.support.GroovyUtils( context )**

**//Retriving DB Value**

**def responseHolder = groovyUtils.getXmlHolder( messageExchange.responseContent )**

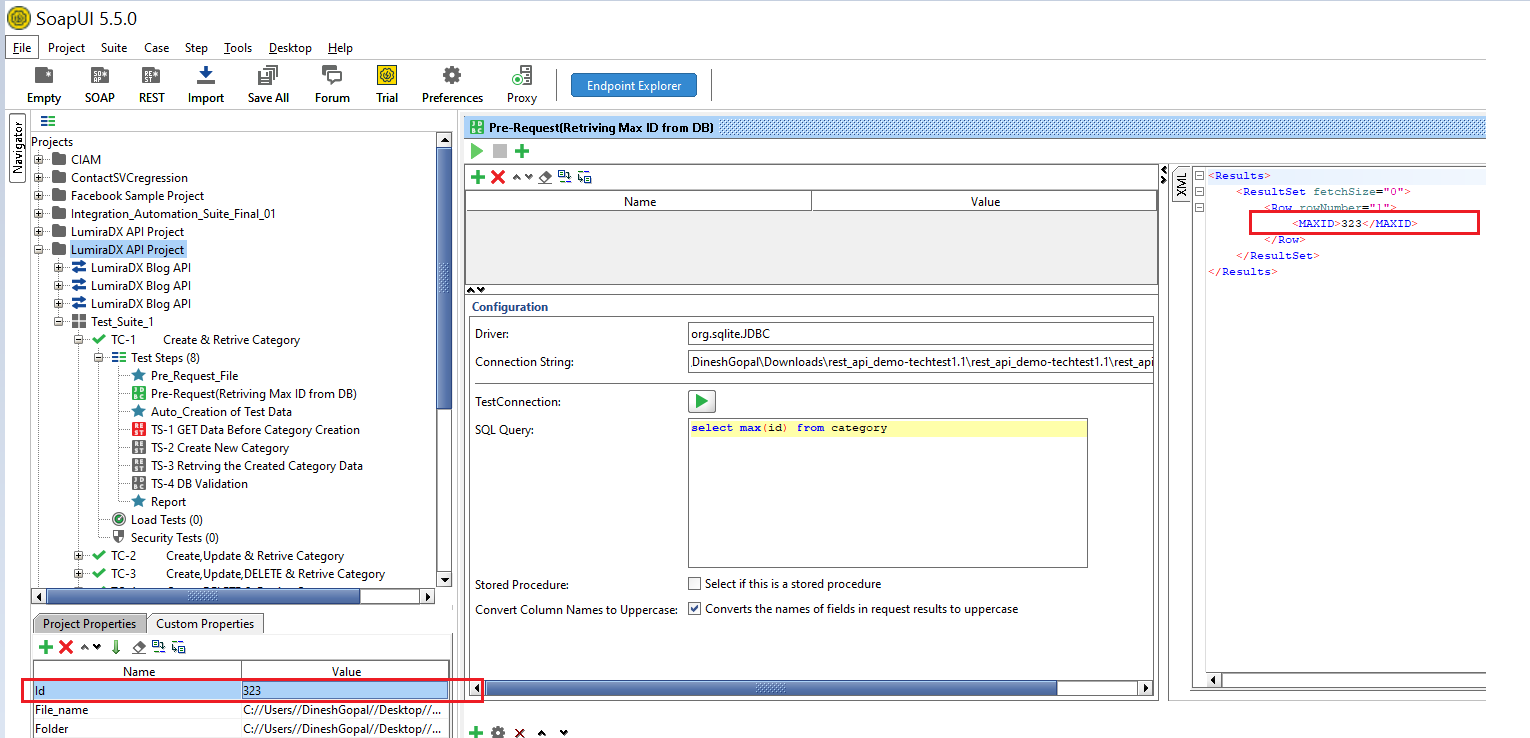
**def DB\_ID = responseHolder.getNodeValue("//\*:Row/\*:MAXID")**

**//validating DB column is not null**

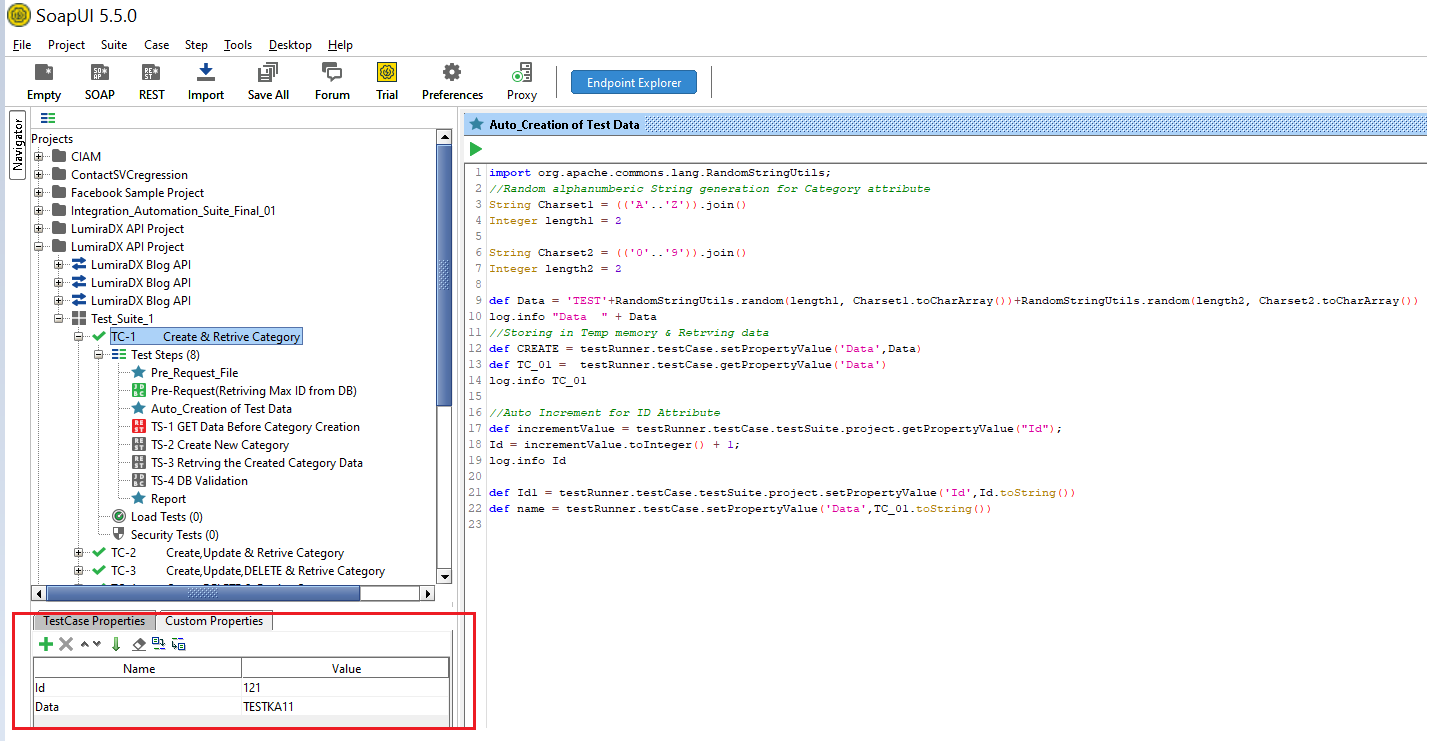
**assert DB\_ID != null**

**//Retriving REST API Data**

**def API\_ID = messageExchange.modelItem.testStep.testCase.testSuite.project.setPropertyValue('Id',DB\_ID)**



**Auto Creation of Test Data:**



**import org.apache.commons.lang.RandomStringUtils;**

**//Random alphanumberic String generation for Category attribute**

**String Charset1 = (('A'..'Z')).join()**

**Integer length1 = 2**

**String Charset2 = (('0'..'9')).join()**

**Integer length2 = 2**

**def Data = 'TEST'+RandomStringUtils.random(length1, Charset1.toCharArray())+RandomStringUtils.random(length2, Charset2.toCharArray())**

**log.info "Data " + Data**

**//Storing in Temp memory & Retrving data**

**def CREATE = testRunner.testCase.setPropertyValue('Data',Data)**

**def TC\_01 = testRunner.testCase.getPropertyValue('Data')**

**log.info TC\_01**

**//Auto Increment for ID Attribute**

**def incrementValue = testRunner.testCase.testSuite.project.getPropertyValue("Id");**

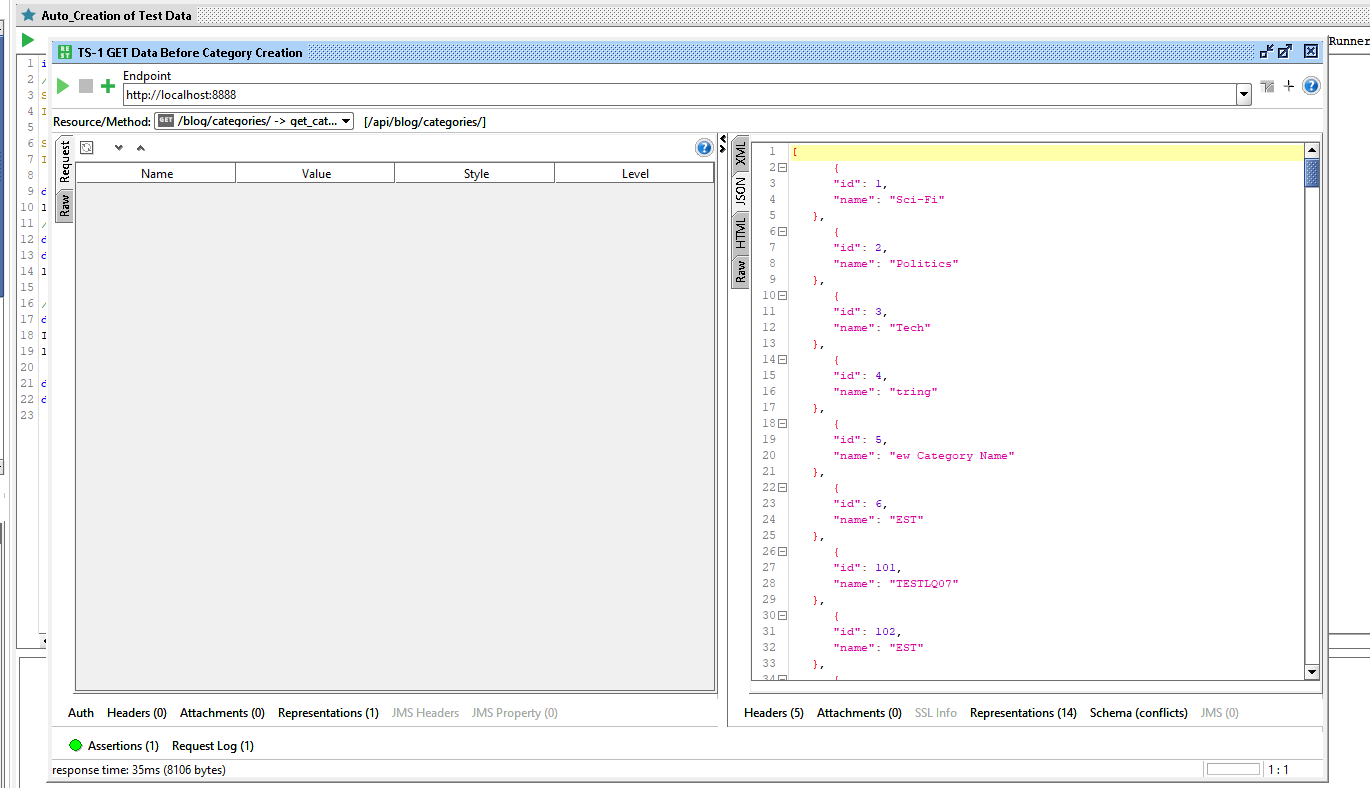
**Id = incrementValue.toInteger() + 1;**

**log.info Id**

**def Id1 = testRunner.testCase.testSuite.project.setPropertyValue('Id',Id.toString())**

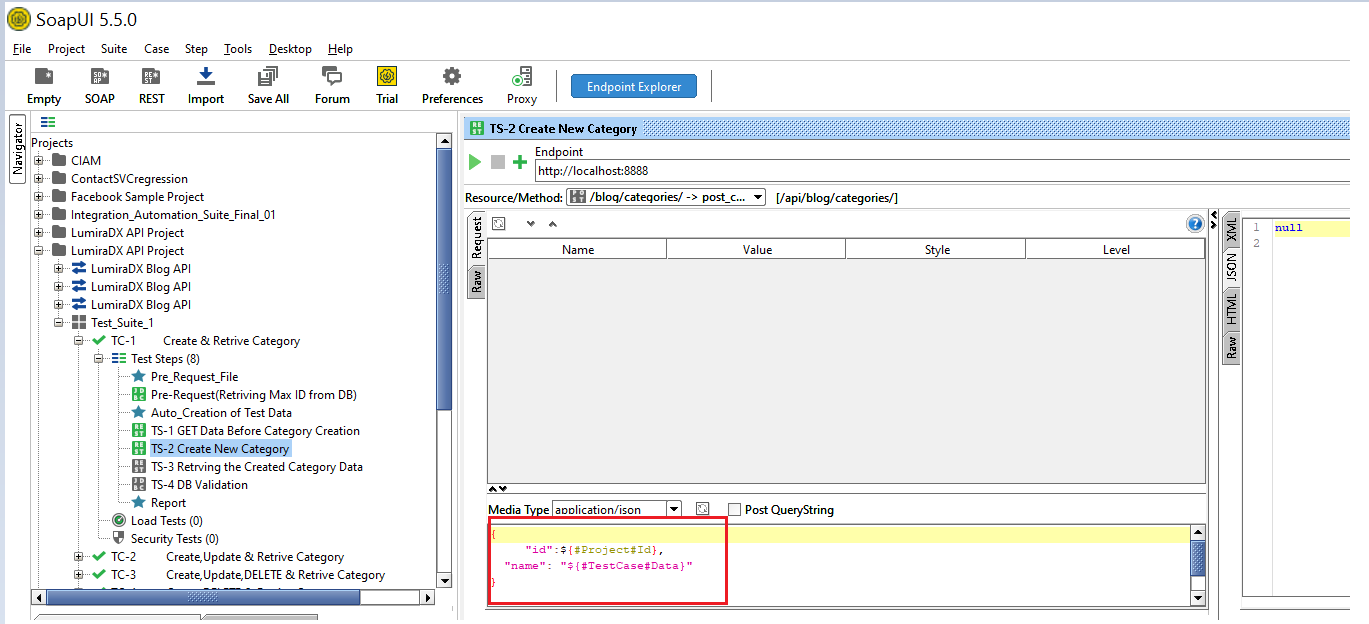
**def name = testRunner.testCase.setPropertyValue('Data',TC\_01.toString())**

**Retrieval of GET service Data**

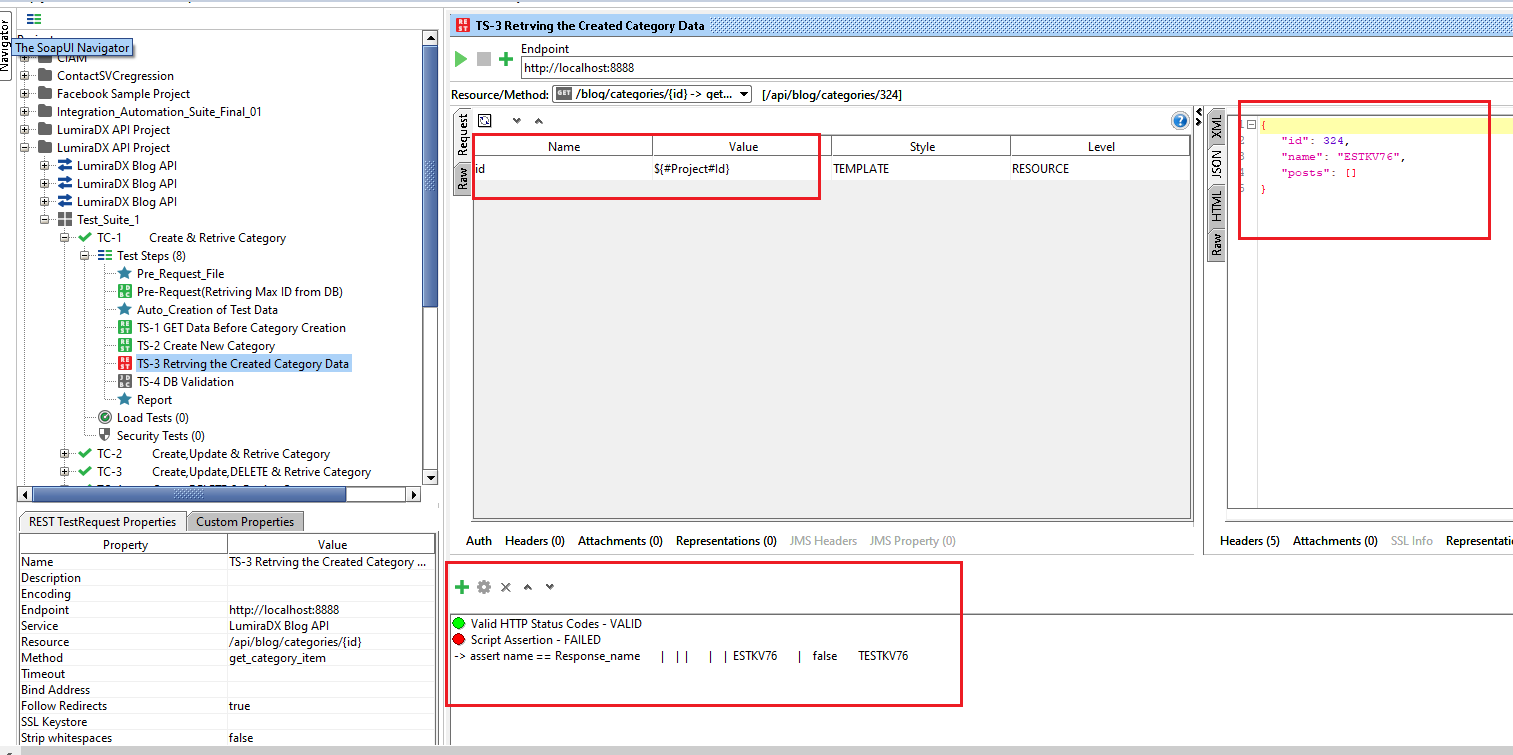


**Creating new Category name using auto created data on POST Service**

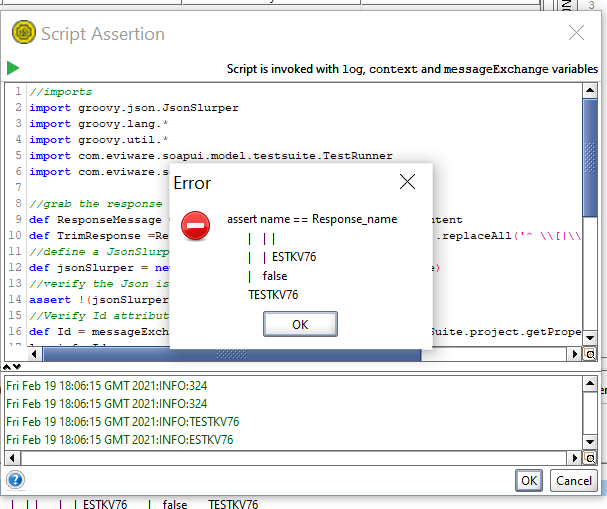
**On each Iteration Data is auto created and passed into attributes using data parameterization**



**Retrieve Data using GET Service using Data parameterization:**



**Script validation of asserting Data:**



**Script:**

**//imports**

**import groovy.json.JsonSlurper**

**import groovy.lang.\***

**import groovy.util.\***

**import com.eviware.soapui.model.testsuite.TestRunner**

**import com.eviware.soapui.model.testsuite.\***

**//grab the response**

**def ResponseMessage = messageExchange.response.responseContent**

**def TrimResponse =ResponseMessage.replaceAll('^\"|\"$','').replaceAll('^ \\[|\\]$','').replaceAll('\\\\','')**

**//define a JsonSlurper**

**def jsonSlurper = new JsonSlurper().parseText(TrimResponse)**

**//verify the Json isn't empty**

**assert !(jsonSlurper.isEmpty())**

**//Verify Id attribute**

**def Id = messageExchange.modelItem.testStep.testCase.testSuite.project.getPropertyValue('Id')**

**log.info Id**

**def A =(jsonSlurper.id).toString()**

**Response\_Id = A.replace("","")**

**log.info Response\_Id**

**assert Id == Response\_Id**

**//Verify Name attribute**

**def name = messageExchange.modelItem.testStep.testCase.getPropertyValue('Data')**

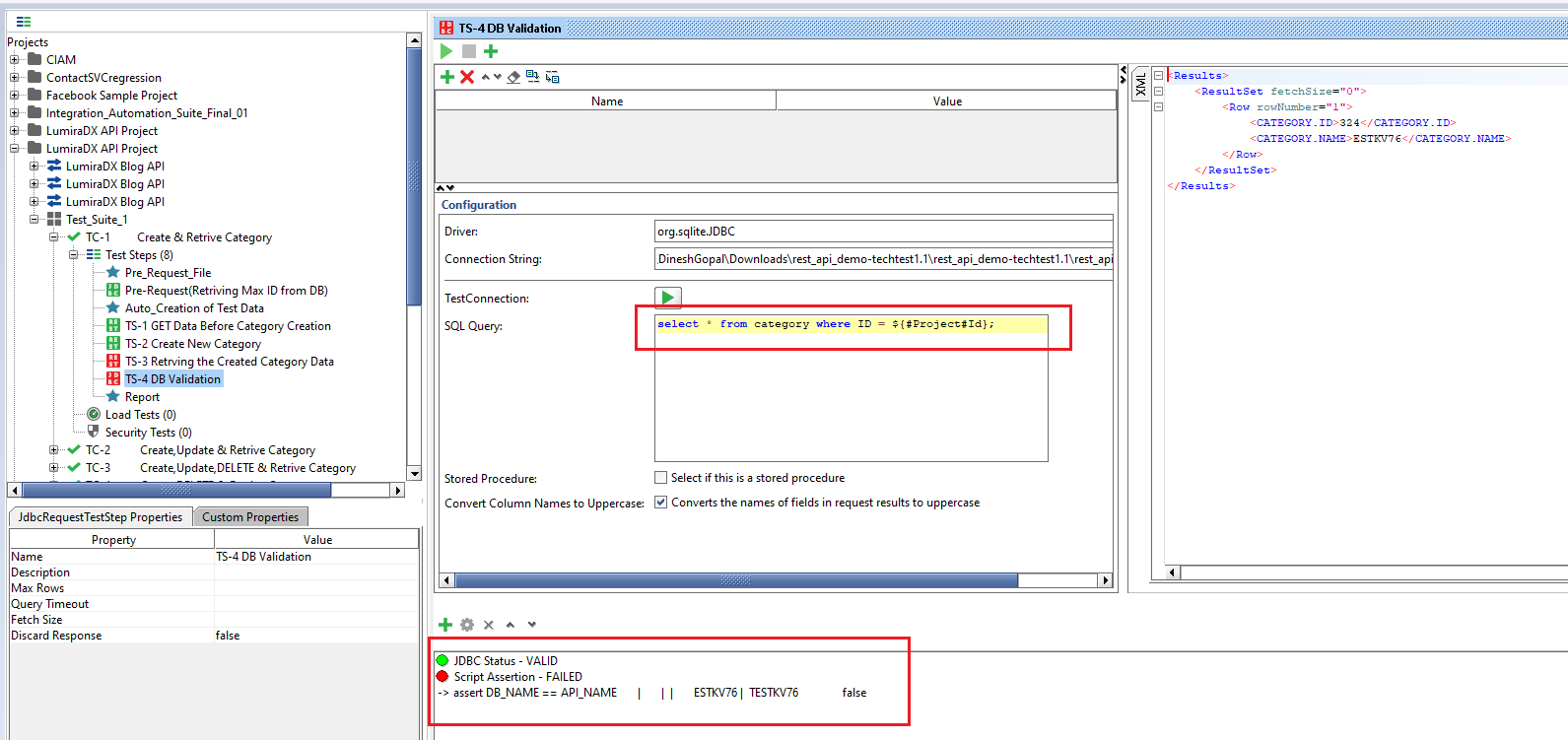
**log.info name**

**def Response\_name = jsonSlurper.name**

**log.info Response\_name**

**assert name == Response\_name**

DB Validation with Groovy Script:



**import groovy.lang.\***

**import groovy.util.\***

**import com.eviware.soapui.model.testsuite.TestRunner**

**import com.eviware.soapui.model.testsuite.\***

**import org.xml.sax.helpers.DefaultHandler**

**import groovy.util.XmlSlurper**

**import com.eviware.soapui.support.xml.XmlUtils**

**def groovyUtils = new com.eviware.soapui.support.GroovyUtils( context )**

**//Retriving DB Value**

**def responseHolder = groovyUtils.getXmlHolder( messageExchange.responseContent )**

**def DB\_ID = responseHolder.getNodeValue("//\*:Row/\*:CATEGORY.ID")**

**def DB\_NAME = responseHolder.getNodeValue("//\*:Row/\*:CATEGORY.NAME")**

**//validating DB column is not null**

**assert DB\_ID != null**

**//Retriving REST API Data**

**def API\_ID = messageExchange.modelItem.testStep.testCase.testSuite.project.getPropertyValue('Id')**

**log.info API\_ID**

**//ASSERTION**

**assert DB\_ID == API\_ID**

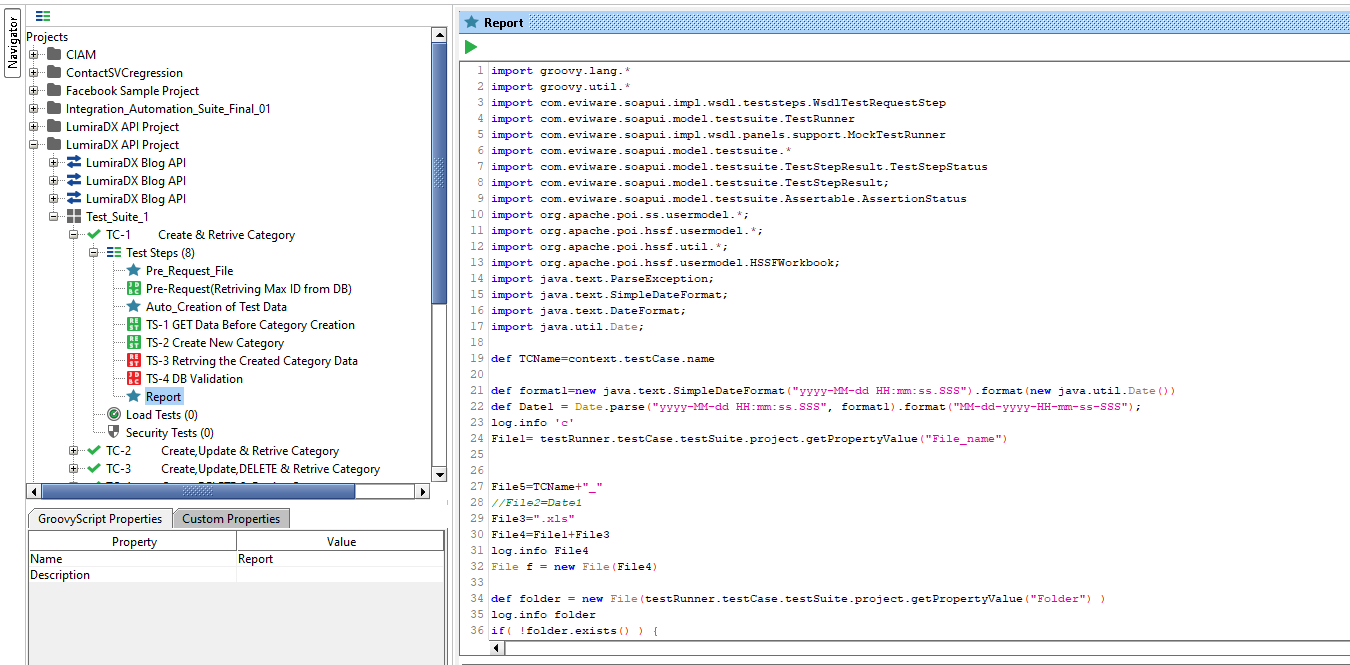
**//Name attribute Validation**

**def API\_NAME = messageExchange.modelItem.testStep.testCase.getPropertyValue('Data')**

**log.info API\_NAME**

**assert DB\_NAME == API\_NAME**

**Report Script:**



**def TCName=context.testCase.name**

**def format1=new java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS").format(new java.util.Date())**

**def Date1 = Date.parse("yyyy-MM-dd HH:mm:ss.SSS", format1).format("MM-dd-yyyy-HH-mm-ss-SSS");**

**log.info 'c'**

**File1= testRunner.testCase.testSuite.project.getPropertyValue("File\_name")**

**File5=TCName+"\_"**

**//File2=Date1**

**File3=".xls"**

**File4=File1+File3**

**log.info File4**

**File f = new File(File4)**

**def folder = new File(testRunner.testCase.testSuite.project.getPropertyValue("Folder") )**

**log.info folder**

**if( !folder.exists() ) {**

**// Create all folders up-to and including B**

**folder.mkdirs()**

**}**

**if(f.exists()!= true )**

**{**

**// Directory path where the xls file will be created**

**String destinationFilePath = f;**

**log.info f**

**// Create object of FileOutputStream**

**FileOutputStream fout = new FileOutputStream(destinationFilePath);**

**try{**

**// Build the Excel File**

**ByteArrayOutputStream outputStream = new ByteArrayOutputStream();**

**HSSFWorkbook workBook = new HSSFWorkbook();**

**// Create the spreadsheet**

**HSSFSheet spreadSheet = workBook.createSheet("RegressionTestResults");**

**// Create the first row**

**HSSFRow row = spreadSheet.createRow((short) 0);**

**// Create the cells and write to the file**

**HSSFCell cell;**

**//Style**

**HSSFCellStyle backgroundStyle = workBook.createCellStyle();**

**backgroundStyle.setFillForegroundColor(IndexedColors.YELLOW.getIndex());**

**backgroundStyle.setFillPattern(CellStyle.SOLID\_FOREGROUND);**

**backgroundStyle.setBorderBottom(CellStyle.BORDER\_THIN);**

**backgroundStyle.setBottomBorderColor(IndexedColors.BLACK.getIndex());**

**backgroundStyle.setBorderLeft(CellStyle.BORDER\_THIN);**

**backgroundStyle.setLeftBorderColor(IndexedColors.BLACK.getIndex());**

**backgroundStyle.setBorderRight(CellStyle.BORDER\_THIN);**

**backgroundStyle.setRightBorderColor(IndexedColors.BLACK.getIndex());**

**backgroundStyle.setBorderTop(CellStyle.BORDER\_THIN);**

**backgroundStyle.setTopBorderColor(IndexedColors.BLACK.getIndex());**

**// Write the headers**

**cell = row.createCell(0);**

**cell.setCellValue(new HSSFRichTextString("Project\_Name"));**

**cell.setCellStyle(backgroundStyle);**

**cell = row.createCell(1);**

**cell.setCellValue(new HSSFRichTextString("Test\_Suite\_Name"));**

**cell.setCellStyle(backgroundStyle);**

**cell = row.createCell(2);**

**cell.setCellValue(new HSSFRichTextString("Test\_Case\_Name"));**

**cell.setCellStyle(backgroundStyle);**

**cell = row.createCell(3);**

**cell.setCellValue(new HSSFRichTextString("Test\_Case\_ID"));**

**cell.setCellStyle(backgroundStyle);**

**cell = row.createCell(4);**

**cell.setCellValue(new HSSFRichTextString("Status"));**

**cell.setCellStyle(backgroundStyle);**

**cell = row.createCell(5);**

**cell.setCellValue(new HSSFRichTextString("Date"));**

**cell.setCellStyle(backgroundStyle);**

**workBook.write(outputStream);**

**outputStream.writeTo(fout);**

**outputStream.close();**

**fout.close();**

**}**

**catch ( IOException ex){**

**FileInputStream file = new FileInputStream(f1);**

**HSSFWorkbook workbook = new HSSFWorkbook (file);**

**HSSFSheet sheet = workbook.getSheetAt(0);**

**rowNum=sheet.getLastRowNum()**

**def h;**

**for(int i=0;i<8;i++)**

**{**

**if(i==0)**

**h= "Project\_Name";**

**else if(i==1)**

**h= "Test\_Suite\_Name";**

**else if(i==2)**

**h= "Test\_Case\_Name";**

**else if(i==3)**

**h= "Test\_Case\_ID";**

**else if(i==4)**

**h="Status";**

**else if(i==5)**

**h= "Date";**

**row.createCell(i).setCellValue(h)**

**}**

**}**

**}**

**def TestCase = testRunner.getTestCase()**

**def StepList = TestCase.getTestStepList()**

**StepList.each{**

**if(it.metaClass.hasProperty(it,'assertionStatus')){**

**Project\_Name= testRunner.testCase.testSuite.project.name**

**Test\_Suite\_Name= testRunner.testCase.testSuite.name**

**Test\_Case\_Name=context.testCase.name**

**Test\_Case\_ID=it.name**

**r=it.assertionStatus**

**if(it.assertionStatus == AssertionStatus.FAILED)**

**{**

**Status="Fail"**

**}**

**else**

**{**

**Status="Pass"**

**}**

**//def a= testRunner.testCase.testSteps[Test\_Case\_ID].testRequest.response.getRequestContent()**

**//def b= testRunner.testCase.testSteps[Test\_Case\_ID].testRequest.response.getResponseContent()**

**def date1=new java.text.SimpleDateFormat("yyyy-MM-dd").format(new java.util.Date())**

**def f1 = f**

**if(f.exists()== true )**

**{**

**FileInputStream file = new FileInputStream(f1);**

**HSSFWorkbook workbook = new HSSFWorkbook (file);**

**HSSFSheet sheet = workbook.getSheetAt(0);**

**rowNum=sheet.getLastRowNum()**

**count= rowNum+1**

**FileOutputStream out = new FileOutputStream(f1);**

**Row row = sheet.createRow(count);**

**row.createCell(0).setCellValue(Project\_Name)**

**sheet.autoSizeColumn(0);**

**row.createCell(1).setCellValue(Test\_Suite\_Name)**

**sheet.autoSizeColumn(1);**

**row.createCell(2).setCellValue(Test\_Case\_Name)**

**sheet.autoSizeColumn(2);**

**row.createCell(3).setCellValue(Test\_Case\_ID)**

**sheet.setColumnWidth(3, 10000)**

**row.createCell(4).setCellValue(Status)**

**sheet.autoSizeColumn(4);**

**row.createCell(5).setCellValue(date1)**

**sheet.autoSizeColumn(5);**

**workbook.write(out);**

**out.close();**

**}**

**else {**

**log.info("Excel Failed")**

**}**

**}**

**}**

**def dontCare = null;**

**Auto Generated Excel Report:**

