****

**XML File Comaprison using Groovy scripts**

**Technical Manual Rev. 1**

**Mastech Infotrellis**

**Apr 2020**

**USER'S MANUAL**

**TABLE OF CONTENT**

|  |
| --- |
| **PURPOSE OF THE DOCUMENT** |
| **DOCUMENT HISTORY**  **ICONS USED** |
| **Chapter 1 Data Driven testing** |
| * 1. Groovy script to perform Data Driven testing   2. Assertions |
| **Chapter 2 Request and Response logging** |
| 2.1 Groovy script to capture request and response |
|  |

PURPOSE OF THE DOCUMENT

The purpose of the document is to demonstrate comparing xml files and listing down the difference using groovy scripts in soapui.

DOCUMENT HISTORY

|  |  |  |  |
| --- | --- | --- | --- |
| **Release No.** | **Date** | **Revision Description** | **Author** |
| **Rev. 1** | **Apr 2020** | **Implementation** |  |
|  |  |  |  |

ICONS USED

|  |  |
| --- | --- |
| icono5 | **Mandatory action** |

XML file comparison in SOAP UI

**Abstract**

The aim is to compare xml files and list down the difference ignoring the dynamic attributes. This is implemented for two scenarios.

1, Feeding the xml files to SOAP UI for comparion

2, Comparing the responses directly in SOAP UI during execution

**Prerequisite** **JARs**

1. For xml comparison 
2. For reading data from xl 
3. Reference 

**Procedure**

**Scenario 1 – Comparing xml files**

**For this sceanrio, the xmls which needs to be compared should be ready.**

**Step 1**

Place the jar files in SOAPUI installation folder\bin\ext

For instance, C:\Program Files (x86)\SmartBear\SoapUI-5.3.0\bin\ext

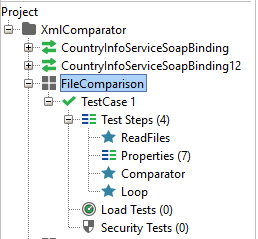
Restart soapui

**Step 2**

We need to have an Excel file with all our data in a local location. For instance, C:\GOALS\ soapUI.xlsx

**Step 3**

Create a test case in SoapUI.

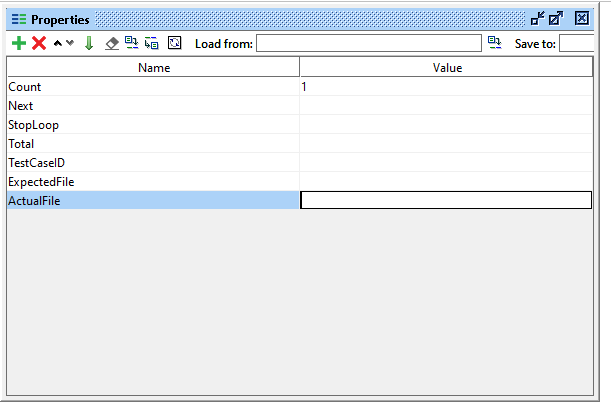


icono5

Test steps – ReadFiles, Properties and Loop are created for Data driven. Kindly refer Data driven document for more information. To refer the code for these steps refer the attached sample project in this document.

**Step 4**

On the Properties step, create the properties of the following picture:



icono5 Make sure Count is initialized to row number (ie..1) of excel sheet from which execution have to be started.

#### Step 5

Inside the ReadFils step, enter the following code:

|  |
| --- |
| import com.eviware.soapui.support.XmlHolder |
| import org.apache.poi.hssf.usermodel.HSSFCell; |
| import org.apache.poi.ss.usermodel.Cell; |
| import org.apache.poi.ss.usermodel.CellStyle; |
| import org.apache.poi.ss.usermodel.DataFormat; |
| import org.apache.poi.ss.usermodel.Row; |
| import org.apache.poi.ss.usermodel.Sheet; |
| import org.apache.poi.ss.usermodel.Workbook; |
| import org.apache.poi.xssf.usermodel.XSSFCell; |
| import org.apache.poi.xssf.usermodel.XSSFRow; |
| import org.apache.poi.xssf.usermodel.XSSFSheet; |
| import org.apache.poi.xssf.usermodel.XSSFWorkbook; |
| try { |
| // Variable Declarations |
| def myTestCase = context.testCase; //myTestCase contains the test case |
| def counter, next, previous, size; //Variables used to handle the loop and to move inside the file |
| XSSFWorkbook workbook1 = new XSSFWorkbook(new FileInputStream("C:/Users/Administrator/Desktop/Utility/XMLComparison/FileComparison/FileNames.xlsx")); //XL file containing the data |
| XSSFSheet sheet1 = workbook1.getSheetAt(0) //getSheet(0) represents Sheet1 in XL file |
| size = sheet1.getPhysicalNumberOfRows()//get the number of rows, each row is a data set |
| propTestStep = myTestCase.getTestStepByName("Properties") // get the Property TestStep object |
| propTestStep.setPropertyValue("Total", size.toString()) |
| counter = propTestStep.getPropertyValue("Count").toString() //counter variable contains iteration number |
| counter = counter.toInteger() |
| next = (counter > size - 2 ? 0 : counter + 1) |
| // Reading data from file |
| XSSFCell TestCaseID = sheet1.getRow(counter).getCell(0) // getCell(column,row). getCell(0,0) represents A1 |
| XSSFCell ExpectedFile = sheet1.getRow(counter).getCell(1) |
| XSSFCell ActualFile = sheet1.getRow(counter).getCell(2) |
|  |
|  |
| String eFilePath = ExpectedFile.getStringCellValue() |
| String eFileContents = new File(eFilePath).getText('UTF-8') |
| String aFilePath = ActualFile.getStringCellValue() |
| String aFileContents = new File(aFilePath).getText('UTF-8') |
|  |
| propTestStep.setPropertyValue("TestCaseID", TestCaseID.toString()) //the value is saved in the property |
| propTestStep.setPropertyValue("ExpectedFile", eFileContents.toString()) //the value is saved in the property |
| propTestStep.setPropertyValue("ActualFile", aFileContents.toString()) //the value is saved in the property |
| propTestStep.setPropertyValue("Count", next.toString()) //increase Count value |
| next++ //increase next value |
| propTestStep.setPropertyValue("Next", next.toString()) //set Next value on the properties step |
| //Decide if the test has to be run again or not |
| if (counter == size - 1) {  propTestStep.setPropertyValue("StopLoop", "T")  propTestStep.setPropertyValue("Count", "1")  log.info "Setting the stoploop property now..."  } else if (counter == 0) {  def runner = new com.eviware.soapui.impl.wsdl.testcase.WsdlTestCaseRunner(testRunner.testCase, null)  propTestStep.setPropertyValue("StopLoop", "F")  assert true  } else {  propTestStep.setPropertyValue("StopLoop", "F")  }  assert true;  }  catch (exc) {  log.error("Exception happened: " + exc.toString());  } |

#### Step 6

On the Loop step enter the following code:

|  |
| --- |
| def myTestCase = context.testCase |
| def runner |
| propTestStep = myTestCase.getTestStepByName(“Properties") // get the Property TestStep |
| endLoop = propTestStep.getPropertyValue("StopLoop").toString() |
| if (endLoop.toString() == "T" || endLoop.toString()=="True" || endLoop.toString()=="true") |
| { |
| log.info ("Exit Groovy Data Source Looper") |
| assert true |
| } |
| else |
| { |
| testRunner.gotoStepByName(“ReadFiles") //setStartStep |
| } |

#### Step 7

On the Comparator step enter the following code:

|  |
| --- |
| import com.eviware.soapui.support.\*;  import org.custommonkey.xmlunit.\*;  import org.xml.sax.SAXException;  import groovy.io.FileType;  import groovy.xml.XmlUtil;  import jxl.\*;  //Get date  def date = new Date();  def dts = date.format("yyyy-MM-dd-HH-mm-ss-ms");  //Create report file  def reportFile = "C:/Users/Administrator/Desktop/Utility/XMLComparison/FileComparison/" + "FC\_ExecutionReport" + ".csv";  def report = new File(reportFile);  if (!report.exists()) {  report.createNewFile();  report.write('"Test Case ID","Total Differences","Result Message","Execution Date"');  }  //Retrieve Test case ID  def myTestCase = context.testCase;  def propTestStep = myTestCase.getTestStepByName("Properties");  def TestCaseID = propTestStep.getPropertyValue("TestCaseID").toString();  //Get documents for comparison  def expFile = context.expand('${Properties#ExpectedFile}');  def actFile = context.expand('${Properties#ActualFile}');  // Creates a list of elements to ignore  Set < String > ignoreList = new HashSet < String > ();  ignoreList.add("Date")  ignoreList.add("Id")  //variable declaration  List allDifferences;  int diffSize = 0;  def msgBuffer = new StringBuffer();  //checking file size  if (expFile == '' || actFile == '') {  msgBuffer.append("Loaded empty file")  } else {  //configuring XMLUnit  XMLUnit.setIgnoreWhitespace(true);  XMLUnit.setIgnoreComments(true)  XMLUnit.setIgnoreDiffBetweenTextAndCDATA(true)  XMLUnit.setNormalizeWhitespace(true)  XMLUnit.setIgnoreAttributeOrder(true);  XMLUnit.setIgnoreDiffBetweenTextAndCDATA(true);  // Create an object with differences between documents  Diff myDiff = new Diff(expFile, actFile)  DetailedDiff diff = new DetailedDiff(myDiff);  // Get list of all differences  allDifferences = diff.getAllDifferences();  diffSize = allDifferences.size();  if (allDifferences.size() == 0) {  msgBuffer.append("Files are Identical")  } else {  int j = 0;  for (int i = 0; i < allDifferences.size(); i++) {  diffNodeName = "";  try {  j++;  diffNodeName = allDifferences.get(i).getControlNodeDetail().getNode().getParentNode().getNodeName()  if (!ignoreList.contains(diffNodeName)) {  msgBuffer.append(j + "." + allDifferences[i])  msgBuffer.append('\n')  } else {  msgBuffer.append(j + "." + "Difference in ignore list tag -" + diffNodeName + '\n')  }  } catch (exc) {  msgBuffer.append(j + "." + allDifferences[i])  msgBuffer.append('\n')  }  }  }  }  //inserting data into report file  report.append('\n');  report.append('"' + TestCaseID + '",');  report.append('"' + diffSize + '",');  report.append('"' + msgBuffer + '",');  report.append('"' + dts + '",'); |

icono5 Edit line number 25 as per your project needs. This is to add the dynamic attributes in the list so that this can be ignored for comparison.

Edit line number 59 by giving path for the report to get saved.

**Step 8**

Save and run the project. Report will generated with list of differences in the given location.

**Scenario 2 – Comparing soap responses directly during execution**

For this sceanrio, data driven approach is followed. Refer Data driven testing use guide for more details.

This projected can also be implemented with assertions for functional testing along with xml comparison.

**Step 1**

Place the jar files in SOAPUI installation folder\bin\ext

For instance, C:\Program Files (x86)\SmartBear\SoapUI-5.3.0\bin\ext

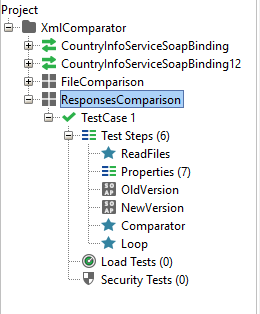
Restart soapui

**Step 2**

Follow Data driven testing steps for preparing excel document and request templates

**Step 3**

Create a test case in SoapUI.



icono5

Test steps – ReadFiles, Properties and Loop are created for Data driven. Kindly refer Data driven document for more information. To refer the code for these steps refer the attached sample project in this document.

**Step 4**

On the Properties step, create the properties as per your project

icono5 Make sure Count is initialized to row number (ie.. 1) of excel sheet from which execution have to be started.

#### Step 5

Inside the ReadFils step, enter the following code:

|  |
| --- |
| import com.eviware.soapui.support.XmlHolder  import org.apache.poi.hssf.usermodel.HSSFCell;  import org.apache.poi.ss.usermodel.Cell;  import org.apache.poi.ss.usermodel.CellStyle;  import org.apache.poi.ss.usermodel.DataFormat;  import org.apache.poi.ss.usermodel.Row;  import org.apache.poi.ss.usermodel.Sheet;  import org.apache.poi.ss.usermodel.Workbook;  import org.apache.poi.xssf.usermodel.XSSFCell;  import org.apache.poi.xssf.usermodel.XSSFRow;  import org.apache.poi.xssf.usermodel.XSSFSheet;  import org.apache.poi.xssf.usermodel.XSSFWorkbook;  try {  // Variable Declarations  def myTestCase = context.testCase; //myTestCase contains the test case  def counter, next, previous, size; //Variables used to handle the loop and to move inside the file  XSSFWorkbook workbook1 = new XSSFWorkbook(new FileInputStream("C:/Users/Administrator/Desktop/Utility/XMLComparison/ResponsesComparison/TestData.xlsx")); //XL file containing the data  XSSFSheet sheet1 = workbook1.getSheetAt(0) //getSheet(0) represents Sheet1 in XL file  size = sheet1.getPhysicalNumberOfRows()//get the number of rows, each row is a data set  propTestStep = myTestCase.getTestStepByName("Properties") // get the Property TestStep object  propTestStep.setPropertyValue("Total", size.toString())  counter = propTestStep.getPropertyValue("Count").toString() //counter variable contains iteration number  counter = counter.toInteger()  next = (counter > size - 2 ? 0 : counter + 1)  // Reading data from file  XSSFCell TestCaseID = sheet1.getRow(counter).getCell(0) // getCell(column,row). getCell(0,0) represents A1  XSSFCell OldVersionInput = sheet1.getRow(counter).getCell(1)  XSSFCell NewVersionInput = sheet1.getRow(counter).getCell(2)    propTestStep.setPropertyValue("TestCaseID", TestCaseID.toString()) //the value is saved in the property  propTestStep.setPropertyValue("OldVersionInput", OldVersionInput.toString()) //the value is saved in the property  propTestStep.setPropertyValue("NewVersionInput", NewVersionInput.toString()) //the value is saved in the property  propTestStep.setPropertyValue("Count", next.toString()) //increase Count value  next++ //increase next value  propTestStep.setPropertyValue("Next", next.toString()) //set Next value on the properties step  //Decide if the test has to be run again or not  if (counter == size - 1) {  propTestStep.setPropertyValue("StopLoop", "T")  propTestStep.setPropertyValue("Count", "1")  log.info "Setting the stoploop property now..."  } else if (counter == 0) {  def runner = new com.eviware.soapui.impl.wsdl.testcase.WsdlTestCaseRunner(testRunner.testCase, null)  propTestStep.setPropertyValue("StopLoop", "F")  assert true  } else {  propTestStep.setPropertyValue("StopLoop", "F")  }  assert true;  }  catch (exc) {  log.error("Exception happened: " + exc.toString());  } |

#### Step 6

On the Loop step enter the following code:

|  |
| --- |
| def myTestCase = context.testCase |
| def runner |
| propTestStep = myTestCase.getTestStepByName(“Properties") // get the Property TestStep |
| endLoop = propTestStep.getPropertyValue("StopLoop").toString() |
| if (endLoop.toString() == "T" || endLoop.toString()=="True" || endLoop.toString()=="true") |
| { |
| log.info ("Exit Groovy Data Source Looper") |
| assert true |
| } |
| else |
| { |
| testRunner.gotoStepByName(“ReadFiles") //setStartStep |
| } |

**Step 7**

Create two soap steps and follow property expansion in the request template.

(Refer data driven testing user guide to know more about property expansion)

You can also assertions to this soap steps if you are planning to do Functional testing as well.

#### Step 8

On the Comparator step enter the following code:

|  |
| --- |
| import com.eviware.soapui.support.\*;  import org.custommonkey.xmlunit.\*;  import org.xml.sax.SAXException;  import groovy.io.FileType;  import groovy.xml.XmlUtil;  import jxl.\*;  //Get date  def date = new Date();  def dts = date.format("yyyy-MM-dd-HH-mm-ss-ms");  //Create report file  def reportFile = "C:/Users/Administrator/Desktop/Utility/XMLComparison/ResponseComparison/" + "RC\_ExecutionReport" + ".csv";  def report = new File(reportFile);  if (!report.exists()) {  report.createNewFile();  report.write('"Test Case ID","Total Differences","Result Message","Execution Date"');  }  //Retrieve Test case ID  def myTestCase = context.testCase;  def propTestStep = myTestCase.getTestStepByName("Properties");  def TestCaseID = propTestStep.getPropertyValue("TestCaseID").toString();  //Get documents for comparison  def expFile = context.expand('${OldVersion#Response}');  def actFile = context.expand('${NewVersion#Response}');  // Creates a list of elements to ignore  Set < String > ignoreList = new HashSet < String > ();  ignoreList.add("m:sCountryFlag")  //variable declaration  List allDifferences;  int diffSize = 0;  def msgBuffer = new StringBuffer();  //checking file size  if (expFile == '' || actFile == '') {  msgBuffer.append("Loaded empty file")  } else {  //configuring XMLUnit  XMLUnit.setIgnoreWhitespace(true);  XMLUnit.setIgnoreComments(true)  XMLUnit.setIgnoreDiffBetweenTextAndCDATA(true)  XMLUnit.setNormalizeWhitespace(true)  XMLUnit.setIgnoreAttributeOrder(true);  XMLUnit.setIgnoreDiffBetweenTextAndCDATA(true);  // Create an object with differences between documents  Diff myDiff = new Diff(expFile, actFile)  DetailedDiff diff = new DetailedDiff(myDiff);  // Get list of all differences  allDifferences = diff.getAllDifferences();  diffSize = allDifferences.size();  if (allDifferences.size() == 0) {  msgBuffer.append("Files are Identical")  } else {  int j = 0;  for (int i = 0; i < allDifferences.size(); i++) {  diffNodeName = "";  try {  j++;  diffNodeName = allDifferences.get(i).getControlNodeDetail().getNode().getParentNode().getNodeName()  if (!ignoreList.contains(diffNodeName)) {  msgBuffer.append(j + "." + allDifferences[i])  msgBuffer.append('\n')  } else {  msgBuffer.append(j + "." + "Difference in ignore list tag -" + diffNodeName + '\n')  }  } catch (exc) {  msgBuffer.append(j + "." + allDifferences[i])  msgBuffer.append('\n')  }  }  }  }  //inserting data into report file  report.append('\n');  report.append('"' + TestCaseID + '",');  report.append('"' + diffSize + '",');  report.append('"' + msgBuffer + '",');  report.append('"' + dts + '",');  //Create request and response files  def myXmlRequestOld = "C:/Users/Administrator/Desktop/Utility/XMLComparison/ResponsesComparison/TestResults/" + TestCaseID + "\_OldVersion\_Request" + ".xml"  def requestOld = context.expand('${OldVersion#Request}')  def reqOld = new File(myXmlRequestOld)  reqOld.write(requestOld, "UTF-8")  def myXmlResponseOld = "C:/Users/Administrator/Desktop/Utility/XMLComparison/ResponsesComparison/TestResults/" + TestCaseID + "\_OldVersion\_Response" + ".xml"  def responseOld = context.expand('${OldVersion#Response}')  def respOld = new File(myXmlResponseOld)  respOld.write(responseOld, "UTF-8")  def myXmlRequestNew = "C:/Users/Administrator/Desktop/Utility/XMLComparison/ResponsesComparison/TestResults/" + TestCaseID + "\_NewVersion\_Request" + ".xml"  def requestNew = context.expand('${NewVersion#Request}')  def reqNew = new File(myXmlRequestNew)  reqNew.write(requestNew, "UTF-8")  def myXmlResponseNew = "C:/Users/Administrator/Desktop/Utility/XMLComparison/ResponsesComparison/TestResults/" + TestCaseID + "\_NewVersion\_Response" + ".xml"  def responseNew = context.expand('${NewVersion#Response}')  def respNew = new File(myXmlResponseNew)  respNew.write(responseNew, "UTF-8") |

icono5 Edit the input parameters as per your project settings

**Step 9**

Save and run the project. Report will generated with list of differences in the given location.

Attachments



Input for scenario 1 

Input for scenario 2 