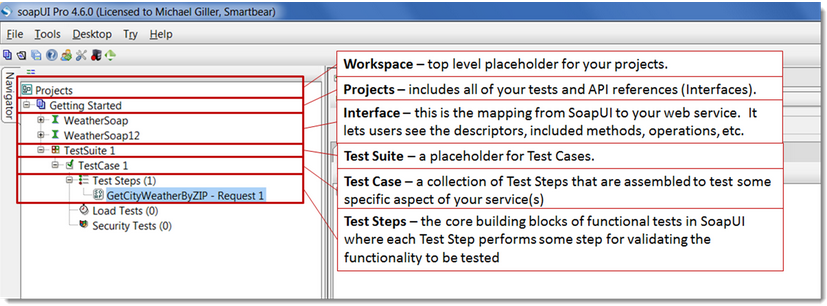
Quick Reference Book SoapUI

# Groovy:

Structure:



Format XML Message: ALT+F

Equality in Validation

Use “.equals ()” for String comparison – compares content not reference.

---------------------------------------------------------------------------------------------

# Test Case Design steps

1. All input Field with Valid data

2. Duplicate data

Ex: Account can be created only once for particular data.

3. All Mandatory field with valid data and null non-mandatory field

4. All Mandatory field as null

5. One mandatory field null/SPACE (for all mandatory field)

Ex: if 4 mandatory field, create 4 test case with one mandatory field null/SPACE at a time.

6. Invalid Data. (Try for all fields or only mandatory field)

Ex: ~{&,/}$%

7. Data type validation

Ex: string in place of number field.

8. Boundary value analysis. (BVA)

Ex: length of some-Id is 4 digit maximum, use 5 digit number for testing.

--------------------------------------------------------------------------------------------

# Xpath

Syntax: //Namespace:NodeName, //\*:NodeName

EX: Let the message be as shown below

<soap:Envelope …">

<soap:Body>

<SendTextToFaxResponse xmlns="http://www.webserviceX.NET">

<SendTextToFaxResult>Success</SendTextToFaxResult>

</SendTextToFaxResponse>

</soap:Body>

</soap:Envelope>

To get “Success” value from above message:

*Declare GroovyUtils (Groovy library)*

Declare holder to hold XML message

Use “getNodeValue” to get content

Script:

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

def holder = groovyUtils.getXmlHolder("SomeServiceName#Response")

holder. getNodeValue("//\*:SendTextToFaxResult")

--------------------------------------------------------------------------------------------

# Xml Holder Declaration:

* Xml holder holds the xml message (Request or Response).
* Holder is used to perform different actions on message, ex: remove node, get node value, set value etc.

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

//response holder

def holder = groovyUtils.getXmlHolder("SomeServiceName#Response")

//request holder

def requestholder = groovyUtils.getXmlHolder("SomeServiceName#Request")

//removeNode**s**

requestholder.removeDomNodes("//\*:NodeName[1]")

//getValue- single value

requestholder. getNodeValue("//\*:NodeName")

//getValues-list of values (if more than one node of same name)

requestholder. getNodeValues("//\*:NodeName")

//setNodeValue

request.setNodeValue("//\*:ToName", value)

//without update no data will be written into request xml.

request.updateProperty()

//setPropertyValue

//runnerComplete points to data sink test step

def runnerComplete = testRunner.testCase.testSteps.DataSink

//set data sink property value

runnerComplete.setPropertyValue("DataSinkProperty",Value)

---------------------------------------------------------------------------------------------

# Clone request

* Copy Template XML request message into SOAP request message (one request to other request)

def requestholder = groovyUtils.getXmlHolder("APR#Request")

def requestTemplate = groovyUtils.getXmlHolder( "Template#Request" )

//cloning to the reference

requestholder.getXmlObject().set(requestTemplate.getXmlObject())

//save

requestholder.updateProperty()

-------------------------------------------------------------------------------------------------------

# Clone Node

* Insert new node in request message

def node1 = request.getDomNode("//\*:FromEmail")

def newNode = node1.cloneNode(true)

node1.getParentNode().insertBefore(newNode, node1.getNextSibling())

-------------------------------------------------------------------------------------------------------

# Remove Node if null

* Remove node if its value is null in test data

def TestData1 = context.expand( '${DataSource# TestData1}' )

if (TestData1 == '') {

request.removeDomNodes("//\*:Node1")

request.updateProperty()

}

-------------------------------------------------------------------------------------------------------

# Write Data To Data Sink

* Write data to Data sink file using groovy script.

def runnerComplete = testRunner.testCase.testSteps.DataSink

runnerComplete.setPropertyValue("DataSinkPropertyName",ResponseNodeValue)

---------------------------------------------------------------------------------------------

# Print Value

log.info ((holder.getNodeValues("//\*:City")).toString())

---------------------------------------------------------------------------------------------

# Select all Values of XML node of name WeatherID

For all value”//\*” -- >holder.getNodeValues("//\*:WeatherID")

---------------------------------------------------------------------------------------------

# For Loop

for (i in 1..5){

result = response\_f;

}

* Same as for(int i=1, i<=5, i++)

---------------------------------------------------------------------------------------------

# Nodes…

def node = holder.getDomNode("//\*:FromEmail")

log.info node.parentNode.toString()

log.info node.childNodes.toString()

---------------------------------------------------------------------------------------------

# Date…

log.info (new Date())

## set date to test request:

${=new java.text.SimpleDateFormat("yyyy-MM-dd").format(new Date())}

<fromDate>${=new java.text.SimpleDateFormat("yyyy-MM-dd").format(new Date())}</fromDate>

<toDate>${=new java.text.SimpleDateFormat("yyyy-MM-dd").format(new Date()+365)}</toDate>

---------------------------------------------------------------------------------------------

**Ip-address:**

${=java.net.InetAddress.getLocalHost().getHostAddress()}

---------------------------------------------------------------------------------------------

# **Random Number** **(Script** ${=(int)(Math.random()\*100)} **should be in single quote)**

def randomNo = context.expand('${=(int)(Math.random()\*100)}')

log.info randomNo

---------------------------------------------------------------------------------------------

# Round Off

ValueBefore = 123.234

def ValueAfter = context.expand('${=(Math.round('+ValueBefore+'))}')

---------------------------------------------------------------------------------------------

# Get and Set Property Value:

def testCaseProperty = testRunner.testCase.getPropertyValue( "UserName" )

log.info testCaseProperty

testRunner.testCase.setPropertyValue( "UserName","Mike")

def testCaseProperty1 = testRunner.testCase.testSteps.GetCityForecastByZIP.getPropertyValue( "Request" )

log.info testCaseProperty1

---------------------------------------------------------------------------------------------

# Set Endpoint Groovy

testRunner.testCase.getTestStepByName(requestName).getHttpRequest().setEndpoint(ENDPOINT)

---------------------------------------------------------------------------------------------

# Date format

def dateformat = new java.text.SimpleDateFormat("yyyy-MM-dd")

log.info dateformat.format(new Date())

DateToday +10: ${=  String.format('%tF', new Date() + 10) }

---------------------------------------------------------------------------------------------

# Split Date

//Get date parameter without timeZone

def creditLineDate = "2014-03-03+05:00";

creditLineDate = creditLineDate.split("\\+");

creditlineDate = creditLineDate[0];

log.info (creditlineDate)

---------------------------------------------------------------------------------------------

# Matching RegEx

Pattern p = Pattern. [compile](http://www.jdocs.com/link/java/util/regex/Pattern.html?m=M-compile(String)) ("a\*b");

Matcher m = p. [matcher](http://www.jdocs.com/link/java/util/regex/Pattern.html?m=M-matcher(CharSequence)) ("aaaaab");

boolean b = m. [matches](http://www.jdocs.com/link/java/util/regex/Matcher.html?m=M-matches()) ();

OR

boolean b = Pattern.matches("a\*b", "aaaaab");

OR

//RegEx://--\d- any digit,--\D-any nonDigit.

def str = responseSTAGE.toString()

matcher = ( str =~ /\d{4}\D\d{2}\D\d{2}/)

log.info matcher

matcher.each {log.info it}

matcher1=Pattern.compile("\\d{4}\\-\\d{2}\\-\\d{2}").matcher(str)

log.info matcher1

matcher1.each {log.info it}

---------------------------------------------------------------------------------------------

# Reverse a string

def node = request.getDomNode("//\*:SendTextToFax")

//log.info node.toString()

def nodeStr = node.toString()

def len = nodeStr.size()

//log.info len

log.info node.childNodes.toString()[-1..-len]

---------------------------------------------------------------------------------------------

# Script assertion:

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

def holder = groovyUtils.getXmlHolder("roleList#Response")

def actualMessage = CommonUtils.getStringValue(holder.getNodeValues("//\*:Message"))

//if System error throw exception, else continue.

if(actualMessage.equals("System error")){

throw new Exception(context.expand( '${Properties#Function}' ) + "-Test Fail: System error in soap response")

}

---------------------------------------------------------------------------------------------

# Xml Slurper

//Xml Slurper: parse the xml response to text

def xmlSlurper = new groovy.util.XmlSlurper()

def responseSTAGE = xmlSlurper.parseText(context.expand('${GetCityForecastByZIP#Response}'));

//fetch value

log.info responseSTAGE.Body.text()

log.info responseSTAGE.Body.GetCityForecastByZIPResponse.GetCityForecastByZIPResult.ForecastResult.Forecast.Date[0].text()

---------------------------------------------------------------------------------------------

Conditional Go-To: if Xpath exists in the response of previous step

Xpath expression

exists(//\*:Success)

OR

matches(//\*:Success, "false")

---------------------------------------------------------------------------------------------

# Property Transfer

get content of Node (using Xpath)

Set content to another Node

---------------------------------------------------------------------------------------------

# Clone parent node and set values (add parent node):

FromEmailarray = fromEmail.split(",")

Subjectarray = Subject.split(",")

def node = holder.getDomNode("//\*:SendTextToFax")

def len = FromEmailarray.size()

if(len != 0)

{

holder.setNodeValue("//\*:FromEmail", FromEmailarray[0])

holder.setNodeValue("//\*:Subject", Subjectarray[0])

//if more than one value add new node and set value

for( int i=1;i<len;i++)

{

log.info("more than one Subject")

def newNode = node.cloneNode(true)

node.getParentNode().insertBefore(newNode, node.getNextSibling())

holder.setNodeValue("//\*:FromEmail", Subjectarray[i])

holder.setNodeValue("//\*:Subject", Subjectarray[i])

}

holder.updateProperty()

}

-------------------------------------------------------------------------------------------------------

# Remove Bracket and Null values

EX:[1,2,3,4] to 1,2,3,4

-------------------------

class CommonUtils {

def static String getStringValue(String[] listValue){

if(null == listValue){

return "";

}

if(0 =

= listValue.size()){

return "";

}

def String str = listValue.toString();

str = str.replace("[", "");

str= str.replace("]", "");

str= str.replace("null, ", "");

str= str.replace(" ,null", "");

str= str.replace("null", "");

return str;

}

}

---------------------------------------------------------------------------------------------

# Add Script library (to save Common Groovy Class file)

* Make new folder (e.g. "C:\GroovyLib"), Add groovy class file as “Callee.groovy”.
* Set File > Preferences > SoapUI Pro tab > Script Library. (I would set that to "C:\GroovyLib" in my example.)
* Then restart SoapUI Pro to pick up the library script.
* Call Groovy as follow(new groovy file)

// Callee.groovy

c = new Callee()

log.info c.hello("Mike")

---------------------------------------------------------------------------------------------

# GetValueOf (method)

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

def holder = groovyUtils.getXmlHolder("APR#Response")

class Getter{

String get(def holder){

def Value = holder.getNodeValues("//\*:Value")

return Value.toString()

}

}

Getter getter = new Getter()

log.info getter.get(holder)

---------------------------------------------------------------------------------------------

# Groovy Looping

for (i in 1..10){

def randomNo = context.expand('${=(int)(Math.random()\*100000)}')

runnerComplete.setPropertyValue("Random Number",randomNo)

log.info randomNo

//request.getDomNode("//\*:ZIP")

request.setNodeValue("//\*:ZIP", randomNo)

request.updateProperty()

//Run TestStep By Name(gotoStepByName)

testRunner.runTestStepByName("GetCityForecastByZIP")

testRunner.runTestStepByName("Validation")

testRunner.runTestStepByName("DataSink")

}

---------------------------------------------------------------------------------------------

# Add properties from external file

* Include properties in a text file (.txt).
* Soap UI Navigation window > Custom Properties > load property value (icon)>
* Browse file, select all check box, OK.

-------------------------------------------------------------------------------------------------------

# Attach file in request initial settings

* Change ‘Enable MTOM’ property of Attachments to true.
* Save file in project directory or provide complete path to file.
* File type can be changed using prefix
  + ‘cid:’ for content type, Ex: cid:sample.docx
  + ‘file:’ for text type, Ex: file:c:\soapui\file.txt

-------------------------------------------------------------------------------------------------------

# Key Store file

<http://docs.oracle.com/cd/E19118-01/n1.sprovsys51/819-1655/fapsf/index.html>

* Open cmd
* Change directory to your JAVA\_HOME/bin folder
* Execute cmd:

keytool -genkey -alias tomcat -keyalg RSA -keystore E:\keystore1.jks -storepass password

-------------------------------------------------------------------------------------

Advanced

# Read Excel

import jxl.Workbook

import jxl.read.biff.BiffException

class ReadExcel {

String inputFile = "D:/SoapUI/SoapUI\_Loan\_prj/NewMicrosoftExcelWorksheet.xls"

String[] read() throws IOException {

def strArray = []

File inputWorkbook = new File(inputFile);

Workbook w;

try {

w = Workbook.getWorkbook(inputWorkbook);

// Get the first sheet

Sheet sheet = w.getSheet(0)

// Loop over columns and lines

for (int j = 0; j < sheet.getColumns(); j++) {

for (int i = 0; i < sheet.getRows(); i++) {

Cell cell = sheet.getCell(i, j)

strArray.add(cell.getContents())

}

}

}

catch (BiffException e) {

e.printStackTrace();

report "Data file not found."

}

//return list

return strArray;

}

}

ReadExcel readExcel = new ReadExcel()

String[] some = readExcel.read()

log.info some.toString()

--------------------------------------------------

# (Groovy) Read excel and write in new excel

import jxl.\*;

import jxl.write.\*

Workbook workbook = Workbook.getWorkbook(new File("D:\\NewMicrosoftExcelWorksheet.xls"));

WritableWorkbook newExcel = Workbook.createWorkbook(new File("d:\\NewExcelSoapUI.xls"),workbook);

Sheet sheet1 = newExcel.getSheet("Sheet1");

WritableSheet sheet2 = newExcel.getSheet("Sheet2");

//copy some column from sheet1 to sheet2 in New Excel

columnToCopy =[0,1,2,3,4,5] ;

for( col in columnToCopy){

def rows = sheet1.getRows();

for( tempRow in 1..rows-1){

WritableCell tmpCell = sheet1.getCell(col,tempRow);

String s1 = tmpCell.getContents();

log.info("$s1")

Label label = new Label(col,tempRow,s1);

sheet2.addCell(label);

}

}

newExcel.write();

newExcel.close();

workbook.close();

---------------------------------------------------------------------------------------------

# Read .xlsx File in Groovy and Run Test Steps:

Add necessory jar files(Apache POI Files) to lib file.

import org.apache.poi.ss.usermodel.\*;

import org.apache.poi.hssf.usermodel.\*;

import org.apache.poi.xssf.usermodel.\*;

import org.apache.poi.ss.util.\*;

import java.io.\*;

class ExcelReader {

def xlsFilePath = null;

ExcelReader(String XlsFilePath){

xlsFilePath = XlsFilePath

}

def readData(String Sheet, context) {

InputStream inputStream = new FileInputStream(xlsFilePath);

Workbook workbook = WorkbookFactory.create(inputStream);

Sheet sheet = workbook.getSheet(Sheet);

Iterator rowIterator = sheet.rowIterator();

rowIterator.next()

Row row;

def rowsData = []

while(rowIterator.hasNext()) {

row = rowIterator.next()

def rowIndex = row.getRowNum()

def colIndex;

def rowData = []

for (Cell cell : row) {

colIndex = cell.getColumnIndex()

def type = cell.getCellType();

switch (type) {

case 0:

def numValue = cell.getNumericCellValue();

rowData[colIndex] = context.expand('${=(Math.round('+numValue+'))}')

break;

case 1:

rowData[colIndex] = cell.getStringCellValue();

break;

case 2:

rowData[colIndex] = cell.getCellFormula();

break;

case 3:

rowData[colIndex] = "";

break;

case 4:

rowData[colIndex] = cell.getBooleanCellValue();

break;

case 5:

rowData[colIndex] = cell.getErrorCellValue();

break;

default:

throw new RuntimeException(

"There are no support for this type of cell");

}

}

rowsData << rowData

}

rowsData

}

}

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

def myTestCase = context.testCase

ExcelReader excelReader = new ExcelReader("D:\\SoapUI\\SoapUI\_Loan\_prj\\Loanxlsx.xlsx");

def requestholder = groovyUtils.getXmlHolder("LeaseMonthlyPayment#Request")

log.info requestholder;

List rows = excelReader.readData("Sheet2", context);

log.info rows

def d = []

Iterator i = rows.iterator();

while( i.hasNext()){

d = i.next();

log.info d[0];

requestholder.setNodeValue("//\*:LoanAmount", d[1])

requestholder.setNodeValue("//\*:ResidualValue", d[2])

requestholder.setNodeValue("//\*:InterestRate", d[3])

requestholder.setNodeValue("//\*:Months", d[4])

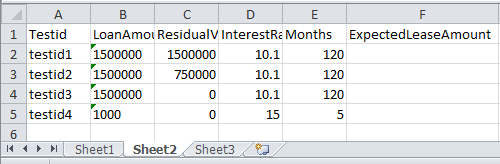
requestholder.updateProperty()

testRunner.runTestStepByName("LeaseMonthlyPayment")

testRunner.runTestStepByName("DataSink")

}

Snapshot:



---------------------------------------------------------------------------------------------

OR

import org.apache.poi.ss.usermodel.\*;

import org.apache.poi.hssf.usermodel.\*;

import org.apache.poi.xssf.usermodel.\*;

import org.apache.poi.ss.util.\*;

import java.io.\*;

class ReadExcelSheetPOI {

def XlsFilePath

ReadExcelSheetPOI(String xlFilePath) {

XlsFilePath = xlFilePath;

}

def getExcelData(def Sheet, context) throws Exception {

File excel = new File(XlsFilePath);

FileInputStream fis = new FileInputStream(excel);

def data = [];

Workbook wb = null;

if (XlsFilePath.contains(".xlsx")) {

wb = new XSSFWorkbook(fis);

} else if (XlsFilePath.contains(".xls")) {

wb = new HSSFWorkbook(fis);

}

try {

Sheet ws = wb.getSheet(Sheet);

int rowNum = ws.getLastRowNum() + 1;

int colNum = ws.getRow(1).getLastCellNum();

// rowNum-1 is used to remove 1st row count

//data = new String[rowNum-1][colNum];

// i=1 represent that it reads from second row.

for (int i = 1; i < rowNum; i++) {

def Data = []

Row row = ws.getRow(i)

for (int j = 0; j < colNum; j++) {

Cell cell = row.getCell(j)

Data[j] = cellToString(cell, context);

// System.out.println("The value is" + value);

}

data << Data

}

}catch (Exception e) {

System.out.println("error in getExcelData()");

}

return data;

}

static String cellToString(Cell cell, context) {

int type;

Object result;

// getCellType will return integer value 0 to 5, depends on cell Type,

// which is used to get cell Value in Switch Case.

type = cell.getCellType();

switch (type) {

case 0:

def numValue = cell.getNumericCellValue();

result = context.expand('${=(Math.round('+numValue+'))}')

//result = cell.getNumericCellValue();

break;

case 1:

result = cell.getStringCellValue();

break;

case 2:

result = cell.getCellFormula();

break;

case 3:

result = "";

break;

case 4:

result = cell.getBooleanCellValue();

break;

case 5:

result = cell.getErrorCellValue();

break;

default:

throw new RuntimeException(

"There are no support for this type of cell");

}

return result.toString();

}

}

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

ReadExcelSheetPOI read = new ReadExcelSheetPOI("D://ExcelFile//GoogleSignUp.xlsx");

List ExcelData = read.getExcelData("Sheet1",context);

def d = []

Iterator i = ExcelData.iterator();

log.info(ExcelData.toString());

while( (i.hasNext())){

d = i.next();

//map is used to identify the colum names of excel file

def map = [loanAmount:d[0], ReidualValue:d[1], interestRate:d[2], months:d[3]]

def requestholder = groovyUtils.getXmlHolder("LeaseMonthlyPayment#Request")

requestholder.setNodeValue("//\*:LoanAmount", map.get("loanAmount"))

requestholder.setNodeValue("//\*:ResidualValue", map.get("ReidualValue"))

requestholder.setNodeValue("//\*:InterestRate", map.get("interestRate"))

requestholder.setNodeValue("//\*:Months", map.get("months"))

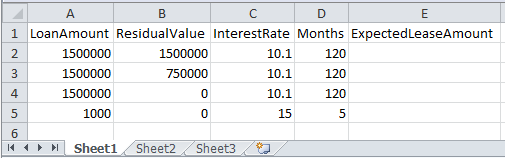
requestholder.updateProperty()

testRunner.runTestStepByName("LeaseMonthlyPayment")

testRunner.runTestStepByName("DataSink")

}

Snapshot:



---------------------------------------------------------------------------------------------

Split input data and set Last value into node

//If null do something

else{

def node = holder.getDomNode("//\*:FromEmail")

def i=0

FromEmailarray = fromEmail.split(",")

def len = FromEmailarray.size()

log.info (len)

for( code in fromEmail.split(","))

{

holder.setNodeValue("//\*:FromEmail", FromEmailarray[len-1])

holder.updateProperty()

}

}

---------------------------------------------------------------------------------------------

~~gu = new com.eviware.soapui.support.GroovyUtils( context )~~

~~def xmlSlurper = new groovy.util.XmlSlurper()~~

~~// Setting up the response parameters~~

~~def responseSTAGE = xmlSlurper.parseText(context.expand('${GET Invoices - STAGE#Response}'));~~

~~def responseSTAGE2 = xmlSlurper.parseText(context.expand('${GET Invoices - STAGE2#Response}'));~~

~~responseInvoicesSTAGE = responseSTAGE.Invoices~~

~~responseInvoicesSTAGE2 = responseSTAGE2.Invoices~~

~~def arrayOfInvoicesSTAGE = []~~

~~def arrayOfInvoicesSTAGE2 = []~~

~~def counter = 0~~

~~for (invoice in responseInvoicesSTAGE.Invoice) {~~

~~arrayOfInvoicesSTAGE[counter] = responseInvoicesSTAGE.Invoice[counter].InvoiceID~~

~~//log.info counter+" STAGE"+arrayOfInvoicesSTAGE[counter]~~

~~arrayOfInvoicesSTAGE2[counter] = responseInvoicesSTAGE2.Invoice[counter].InvoiceID~~

~~//log.info counter+" STAGE2"+arrayOfInvoicesSTAGE2[counter]~~

~~counter++~~

~~}~~

~~log.info arrayOfInvoicesSTAGE~~

~~log.info arrayOfInvoicesSTAGE2~~

~~def sortedSTAGE = arrayOfInvoicesSTAGE.sort()~~

~~def sortedSTAGE2 = arrayOfInvoicesSTAGE2.sort()~~

~~log.info sortedSTAGE~~

-----------------------------------------------------------------------------

# **POM.**x**ml: (Maven)**

<project

Schema definition…>

POM Prerequisites…

<pluginRepositories>

<pluginRepository>

<id>SmartBearPluginRepository</id>

<url>http://www.soapui.org/repository/maven2/</url>

</pluginRepository>

</pluginRepositories>

<build>

<plugins>

<plugin>

<groupId>com.smartbear.soapui</groupId>

<artifactId>soapui-pro-maven-plugin</artifactId>

<version>4.6.1</version>

<executions>

<execution>

<phase>test</phase>

<goals>

<goal>test</goal>

</goals>

<configuration> <projectFile>${project.basedir}/src/soapui-project.xml</projectFile>

<testSuite>WeatherSoap12 TestSuite</testSuite>

<testCase>GetCityForecastByZIP TestCase</testCase>

<outputFolder>${project.basedir}/target/site</outputFolder>

<junitReport>true</junitReport> <printReport>true</printReport>

<exportAll>false</exportAll>

<reportName>IHG-B2BRegression-ProjectReport</reportName>

<reportFormat>PDF,HTML</reportFormat>

<projectProperties>

<value>message=Hello World!</value>

</projectProperties>

<soapuiProperties>

<property> <name>soapui.scripting.library</name> <value>${project.basedir}/src/test/scripts</value>

</property>

</soapuiProperties>

</configuration>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

---------------------------------------------------------------------------------------------

# JDBC connection

oracle.jdbc.driver.OracleDriver

jdbc:oracle:thin:sys as sysdba/PASS\_VALUE@127.0.0.1:1521:xe

\*\*\*\*\*\*\*\*

Select \*

From USER\_DETAIL  
---------------------------------------------------------------------------------------------