**Groovy script for reading data from an Excel sheet**

**Following is the Groovy script for reading data:**

import com.eviware.soapui.model.\*

import com.eviware.soapui.model.testsuite.Assertable

import com.eviware.soapui.support.XmlHolder

import java.io.File;

import java.util.\*;

import jxl.write.\*

import jxl.\*

def regLogger =

org.apache.log4j.Logger.getLogger("RegressionTestLoger");

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

context )

def properties = new java.util.Properties();

//context.expand('${Properties#propertyname}')

def s2

def s3=(testRunner.testCase.getPropertyValue("RUN"))

regLogger.info(s3);

if (s3 != '1' && s3 != '2' && s3 != '3')

{

testRunner.testCase.setPropertyValue("RUN", '1' );

s3=(testRunner.testCase.getPropertyValue("RUN"));

}

Workbook workbook = Workbook.getWorkbook(new

File("D:\\myfile.xls"))

for (count in 1..< 11) // This is from row1 to row 11 based on

the number of properties that you have in the excel sheet in

this case the values were 10

{

Sheet sheet = workbook.getSheet(1)

Cell a1 = sheet.getCell(0,count) // getCell(row,column) —

place some values in myfile.xls

Cell b2 = sheet.getCell(s3.toInteger(),count) // values will

be acessed using a1, b2 & c3 Cell.

String s1 = a1.getContents();

s2 = b2.getContents();

testRunner.testCase.setPropertyValue(s1,s2);

}

workbook.close()

You can also see in the following code how the value is reset to 1:

if (s3 != '1' && s3 != '2' && s3 != '3')

{ // S3 means the value of

//the excel sheet column from where the data values of request are takenfrom

testRunner.testCase.setPropertyValue("RUN", '1' );

s3=(testRunner.testCase.getPropertyValue("RUN"));

**So following is the script that keeps tracks of the run, and can be compared to the data loop functionality**

**data loop functionality in SoapUI Pro replicated in OS**

import com.eviware.soapui.support.XmlHolder

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

)

Run1 = (testRunner.testCase.getPropertyValue("RUN")).toLong()+1;

testRunner.testCase.setPropertyValue("RUN",Run1.toString ());

**Now let's have look at the following example to connect to an Oracle database:**

import groovy.sql.Sql;

def regLogger =

org.apache.log4j.Logger.getLogger("RegressionTestLoger");

def GUID;

def delayStep =

testRunner.testCase.testSuite.getPropertyValue("delayStep")

def tryCount =

testRunner.testCase.testSuite.getPropertyValue("delayRetries")

[ 11 ]

Test Automation in SOA World

int x = 1

int y = 0

while ( x <= Integer.parseInt(tryCount) & y != 1 & y != 2 )

{

println "Delaying " + Integer.parseInt(delayStep)\* 0.001 + "

seconds."

Thread.sleep(Integer.parseInt(delayStep))

def sql =

groovy.sql.Sql.newInstance

("jdbc:oracle:thin:@10.252.168.219:1521:PSYMQA4","CWDEV4SL",

"CWDEV4SL", "oracle.jdbc.driver.OracleDriver")

row = sql.firstRow ("select ordstatus, ordtype, GUID, errorcode,

errorsource, errortext from com\_header where accountnumber= " +

account + "order by cwordercreationdate desc")

[Snippet]

while ( x <= Integer.parseInt(tryCount) & y != 1 & y != 2 ){

println "Delaying " + Integer.parseInt(delayStep)\* 0.001 + "

seconds."

Thread.sleep(Integer.parseInt(delayStep))

}

For multiple selections from the database you can use the following snippet:

sql.eachRow('select \* from tablename where name='Pranai'

You may also try to insert and update the tables by using the following statements:

def params = [10, 'Name', 'Age']

sql.execute 'insert into PROJECT (id, name, Age) values (?, ?, ?)',

params

Updating the values of a table

def newname = 'Nandan'

def project = 'Testing'

sql.executeUpdate "update PROJECT set Name=$newname where

name=$project"

**Now lets consider the need to validate a value from the database to the expected**

**values; in this case we would need to extend the script.**

**Validation using values retrieved from the database:**

import groovy.sql.Sql;

def regLogger =

org.apache.log4j.Logger.getLogger("RegressionTestLoger");

def GUID;

def delayStep =

testRunner.testCase.testSuite.getPropertyValue("delayStep")

def tryCount =

testRunner.testCase.testSuite.getPropertyValue("delayRetries")

def account =

testRunner.testCase.getPropertyValue("imsComboAcct2")

int x = 1

int y = 0

while ( x <= Integer.parseInt(tryCount) & y != 1 & y != 2 )

{

println "Delaying " + Integer.parseInt(delayStep)\* 0.001 + "

seconds."

Thread.sleep(Integer.parseInt(delayStep))

def sql =

groovy.sql.Sql.newInstance("jdbc:oracle:thin:@10.252.168.219:1521:PSYM

QA4","CWDEV4SL", "CWDEV4SL", "oracle.jdbc.driver.OracleDriver")

row = sql.firstRow ("select ordstatus, ordtype, GUID, errorcode,

errorsource, errortext from com\_header where accountnumber= " +

account + "order by cwordercreationdate desc")

if (row.ordstatus == "COM" ) { y = 1 }

w=5

GUID=row.GUID

Test Automation in SOA World

testRunner.testCase.setPropertyValue("GUID1",GUID.toString ());

testRunner.testCase.setPropertyValue("pranai",w.toString ());

if (row.ordstatus == "ERR" ) { y = 2 }

x++

}

if (row.ordstatus == "COM" )

{

return (" Passed account# =" + account + " in pending pre-

provisioning completed in : " +

x \* Integer.parseInt(delayStep) \* 0.001 + " sec" )

testRunner.testCase.setPropertyValue("checkorder","Passed");

}

else

{

regLogger.info("\*FAILED Acct: " + account + " , Case: " +

testRunner.testCase.name);

assert false: "error" + "account#" + account + " status=" +

row.ordstatus + " source= " +

row.errorsource + " text=" + row.errortext + " delay=" + x \*

Integer.parseInt(delayStep) \* 0.001 + " sec"

testRunner.testCase.setPropertyValue("checkorder","Failed");

}

Description of the script:

About the scenario: We need to verify the order status in the database and then

mark the test case passed or failed based on it.

if (row.ordstatus == "ERR" which means the account is in error state and hence

the test case should fail.

Or:

if (row.ordstatus == "COM" which means that the account is provisioned and the

**test case is passed.**

**We have used assert for validations in the script.**

**We may also, at certain points, need to validate the number of rows for particular**

**search criteria, and for that the code below might be useful:**

def rows = sql.rows("select \* from PROJECT where name like 'Pranai%'")

assert rows.size() == 2

So, let's move to our third asset in test automation reporting.

Strategy 2:

Another way of doing it is through Groovy script. If you want to use Groovy script

to do this please find it following:

import com.eviware.soapui.model.\*

import com.eviware.soapui.model.testsuite.Assertable

import com.eviware.soapui.support.XmlHolder

import java.io.File;

import java.util.\*;

import jxl.write.\*

import jxl.\*

import com.eviware.soapui.support.XmlHolder

import com.eviware.soapui.model.\*

import com.eviware.soapui.model.testsuite.Assertable

import com.eviware.soapui.support.XmlHolder

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

)

import java.io.File;

def regLogger = org.apache.log4j.Logger.getLogger("RegressionTestLog

er");

def properties = new java.util.Properties();

[ 30 ]

def environment = context.expand( '${#TestSuite#Environment}' )

def myEndpoint = environment

//def myEndpoint = "http://10.10.101.1/myApplicationAPI/version2.1/

service.svc"

def project = context.testCase.testSuite.project

testSuiteList = project.getTestSuites()

text = "~"

testSuiteList.each

{

testSuite = project.getTestSuiteByName(it.key)

testCaseList = testSuite.getTestCases()

log.info " ${text\*5} TestSuite :: $testSuite.name"

testCaseList.each

{

testCase = testSuite.getTestCaseByName(it.key)

log.info " ${text\*5} Testcase :: $testCase.name"

wsdlTestSteps = testCase.getTestStepsOfType( com.eviware.soapui.

impl.wsdl.teststeps.WsdlTestRequestStep.class )

wsdlTestSteps.each

{

it.properties['Endpoint'].value = myEndpoint

}

}

}

log.info "All the endpoints are now : $Endpoint desired"

**Example 1: Consider an application where a request once triggered by SoapUI now**

**hits multiple subsystems. You want to validate whether the request you sent hits**

certain domains at the backend or not, which is stored in the database:-

Here is an example:

import groovy.sql.Sql;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import com.eviware.soapui.model.\*

import com.eviware.soapui.model.testsuite.Assertable

import org.junit.Assert.\*;

import com.eviware.soapui.support.XmlHolder

import java.io.File; import java.util.\*;

def regLogger = org.apache.log4j.Logger.getLogger("RegressionTestLog

er");

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

)

def properties = new java.util.Properties();

int x = 1

int y = 0

def sql=groovy.sql.Sql.newInstance("jdbc:oracle:thin:@17.239.192.134:2

296:DAEH","app\_eai", "123", "oracle.jdbc.driver.OracleDriver")

def data =sql.rows("SELECT Distinct DOMAIN FROM EH\_TRACE\_LOG WHERE

EVENT\_LOCAL\_ID=" + Prop + "") // Prop is unique Identifier

testRunner.testCase.setPropertyValue("ActualDomainHit",data.

toString());

def Prop =testRunner.testCase.getPropertyValue("MSSIDtobeusedfordb");

def Expected=testRunner.testCase.getPropertyValue("ExpectedDomainHit")

def Actual=testRunner.testCase.getPropertyValue("ActualDomainHit")

assert Actual =~ Expected

Example 2: Data validation.

We usually want to validate the data in real time with the expected values. To do

that, SoapUI provides us with an XPath assertion and an XQuery assertion, but if

you want to do it with help of Groovy script, that is possible as well. The following is

an example of how to achieve that in real time:

import com.eviware.soapui.support.XmlHolder

import com.eviware.soapui.model.\*

import com.eviware.soapui.model.testsuite.Assertable

import com.eviware.soapui.support.XmlHolder

import java.io.File;

import java.util.\*;

import jxl.write.\*

import jxl.\*

import com.eviware.soapui.support.XmlHolder

import com.eviware.soapui.model.testsuite.Assertable

import com.eviware.soapui.support.XmlHolder

import java.io.File;

import com.eviware.soapui.support.\*;

import java.util.\*;

import java.lang.\*;

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

)

def project = context.testCase.testSuite.project

def holder = groovyUtils.getXmlHolder( "getSROrderData 02#Response" )

holder.namespaces["ProvTypes"] = "http://xml.comcast.com/

provisioning\_/types"

holder.namespaces["CSCTypes"] = "http://xml.comcast.com/common/types"

holder.namespaces["ProvServices"] = "http://xml.comcast.com/

provisioning\_/services"

def L=testRunner.testCase.getPropertyValue("productIDHSD")

def ProvisionStatus = holder.getNodeValue("//ProvServices:SubmitPro

visioning[1]/ProvServices:ProvisioningReqType[1]/ProvTypes:serviceRe

questCustomerService[1]/ProvTypes:serviceRequestCustomerService[1]/

ProvisionStatus/text()")

def NewOrderStatus =holder.getNodeValue("//ProvServices:SubmitProvi

sioning[1]/ProvServices:ProvisioningReqType[1]/ProvTypes:serviceReq

uestCustomerService[1]/ProvTypes:serviceRequestCustomerService[1]/

InstallStatus[1]/text()")

if (ProvisionStatus==Pass && NewOrderStatus =="SUCCESS!")

{

testRunner.testCase.setPropertyValue("New Install Status

scenario","Passed")

testRunner.testCase.setPropertyValue("scenario",m)

}

Test Automation in SOA World

else

{

testRunner.testCase.setPropertyValue("New Install Status

scenario","Failed")

}

The previous script is used to validate the response data for two values:

• ProvisionStatus

• NewOrderStatus

If the status is success for both fields, it signifies that the request was successful

and hence we can mark the test case as passed; otherwise we can mark the test

case as failed.

Assertions are the most powerful weapon of software automation testers, and we

should list down the assertion types before designing any automation framework or

planning for automation in the project. This gives you a foresight of things and you

can plan as per you needs.

Well this brings us to the end of the test automation framework design utilities

section, but before we close it, would like to share some sample script which

can be handy while you test automate your applications:

Running a test case using a Groovy script:

def testStep = testRunner.testCase.testSteps['Test Step name1']

for (count in 0..< 2)

{

testStep.run( testRunner, context)

}

def testStep1 = testRunner.testCase.testSteps['Test Step name2']

for (count1 in 0..< 2)

{testStep1.run( testRunner, context)

}

def testStep2 = testRunner.testCase.testSteps['Test Step name3']

for (count2 in 0..< 2)

{testStep2.run( testRunner, context)

}

**Script for creating a test project using Groovy script:**

import com.eviware.soapui.impl.wsdl.teststeps.registry.

GroovyScriptStepFactory

suite = context.testCase.testSuite.project.addNewTestSuite("TestAutom

ationDemo")

tc = suite.addNewTestCase("Order Orchestration for a New order ")

gs = tc.addTestStep( GroovyScriptStepFactory.GROOVY\_TYPE, "ali's

GroovyScript" )

gs.properties["script"].value = 'log.info(\'hello world\')'

context.testCase.testSuite.project.save()

**Below an example for test initialization script:**

import com.eviware.soapui.model.\*

import com.eviware.soapui.model.testsuite.Assertable

import com.eviware.soapui.support.XmlHolder

def regLogger = org.apache.log4j.Logger.getLogger("RegressionTestLog

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

def properties = new java.util.Properties();

Workbook workbook = Workbook.getWorkbook(new File("D:\\

TestinitializationProperties.xls"))

Sheet sheet = workbook.getSheet(1)

Cell a1 = sheet.getCell(0,count)

String s1 = a1.getContents();

Cell b2 = sheet.getCell(1,count)

String s7 = b2.getContents();

testRunner.testCase.testSuite.setPropertyValue(s1,s7);

**Requirement 5: Unique data**

**In certain cases, a situation arises where each time the test request is run we need to**

**send unique data in one or other of the parameters. For this specific purpose you can**

**use the random number generation script and use it for your test data creation:**

def UniqueID;

def regLogger = org.apache.log4j.Logger.getLogger("RegressionTestLoger");