Groovy is object oriented program language🡪 similar to java

**Why we need Scripting /Programming language?**

In project we are testing web services. You are sending xml document to one node to another node when you are validating inside response you need program language.

Request or response tag. How will you read it. You want to compare actual data to expected data.

**SoapUI Groovy Script Test step:**

[Groovy Script](http://www.groovy-lang.org/) test step is included for **custom automation test script creation in SoapUI / Pro.** It can be used for **functional/ load/regression.**

it helps **us to customize and add custom validations to SoapUI tests**

Groovy is a **scripting language** which internally includes **all the Java libraries,** therefore all java related keywords and functions can be used in the groovy script directly. The Java libraries come with SoapUI and are integrated during the SoapUI Pro installation itself.

**What is property Transfer?**

# We can transfer the properties from one test step to other test step, from properties step to request, one step response to other step request.

Groovy Script:1

-🡪 Log(object)

Default object, we can write something in logs. It is a standard log4j logger object.

Log.info “welcome”

* testRunner(object)

testRunner object is the entry point for soapui api for accessing project related items, result etc.

this object is actually executing test cases by going

//log.info "Hello"

//log.error "exception..."

def x=10

def y="hello"

def z=100.32

def a=true;

//log.info y

//log.info "x"

//log.info a

//log.info "value of x is " + x

//log.info x+z

//log.info "sum :" +(x+z) // sum....

//log.info "concate" +x+z

//log.info x+z+" is the sum"

//

//// CONDITION

//log.info x<z

//log.info x == z

//

//if statements --boolean

//if(true){

// log.info "Inside if statement"

//}

//if(x==z){

// log.info " x and z are equal"

// }

// else if(x>z){

log.info "x is greater than z"

// }

//else {

// log.info " less than"

//}

//for loop

for(i=0;i<10;i=i+1)

{ log.info i

}

Groovy:-2

**def arr = [23,53,63]**

**log.info arr[0]**

**log.info arr[1]**

**log.info arr[2]**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**def a=new int[4]**

**a[0]=10**

**a[1]=20**

**a[2]=30**

**a[3]=40**

**for(def b:a)**

**{**

**log.info b**

**}**

**ere is how Groovy script can be added to a test:**

**Step #1.** In SoapUI Pro create a [SOAP project with valid WSDL document](http://www.softwaretestinghelp.com/soapui-tutorial-4-working-with-soapui-projects/). Under the project, create a test suite with the desired name.

**//** – indicates single line comment and  
**/\* <some script> \*/** – denotes multi-line comment

**Arithmetic Operators:**

**+**   Addition operator / String concatenation  
**–**    Subtraction operator  
**\***   Multiplication operator  
**/**   Division operator  
**%**   Remainder operator

// Arithmetic Operators Samples

// Addition Operator  
int x1 = 100 + 200  
log.info (“Addition Result :” + x1);

// Concatenation of Two Strings using PLUS ( + ) operator  
String city =”Timothy E.” + ” Shepherd”;  
log.info(“String Concatenation:” + city);

// Subtraction Operator  
int x2 = 200 – 100  
log.info (“Subtraction :” + x2);

// Multiplication Operator  
int x3 = 10 \* 200  
log.info (“Multiplication :” + x3);

// Division Operator  
int x4 = 200 / 10  
log.info (“Division :” + x4);

// Modulus Operator  
int x5 = 10 % 3  
log.info (“Reminder or Modulus:” + x5);

**Assignment Operators:**

The basic assignment operator is an equal sign (=). Likewise, there are other useful assignment operators available. They are +=, -=, \*=, /=, %=.

Let us see the samples.

int A=100;  
A += 10;                // Similar to A = A + 10  
log.info(A);

The above script produces 110. If we use minus equal to operator in the below script, output will be 40.

int B=50;  
B -= 10;  
log.info(B);

Likewise we can use the remaining operators like this.

int C=10;  
C \*= 10;  
log.info(C);

And,

int D=50;  
D /= 10;  
log.info(D);

Here’s the reminder operator is used as

int E=10;  
E %= 3;  
log.info(E);

This will divide the value 10 by 3 and the remainder will be assigned to the variable “E”.

**Groovy-3**

Planet.log=log

Planet p1=new Planet()

Planet p2=new Planet()

Planet p3=new Planet()

p1.name="earth"

p2.name="jupiter"

p3.name="mars"

log.info p1.name

log.info p2.name

log.info p3.name

p1.printName()

class Planet{

def static log

def name

def shape

public void printName(){

log.info("name $name")//

log.info (name)//

}

}

**Global Properties:**

Now let us discuss global properties. These properties are defined in one place and we can access them across the project components like test suite, test case, test steps etc.

Here are the scripts for writing data to the global properties.

|  |  |  |
| --- | --- | --- |
|  | com.eviware.soapui.SoapUI.globalProperties.setPropertyValue | |
|  | ( "prjFromCurrency", "USD" ) |

|  |  |  |
| --- | --- | --- |
| 3 | com.eviware.soapui.SoapUI.globalProperties.setPropertyValue | |
| 4 | ( "prjToCurrency", "INR" )  testRunner.testCase.setPropertyValue("CaseProperty","101")// testcase  testRunner.testCase.setPropertyValue("userid","101")  testRunner.testCase.testSuite.setPropertyValue("SuiteProperty","102")// test suite..  testRunner.testCase.testSuite.project.setPropertyValue("ProjectProperty","hello")// project property....  log.info testRunner.testCase.testSuite.project.getPropertyValue ("ProjectProperty")  log.info testRunner.testCase.getPropertyValue("CaseProperty")  log.info testRunner.testCase.testSuite.getPropertyValue("SuiteProperty") |

Once we execute the above test step script, the mentioned properties will be created and the respective values will be assigned to those properties. Let us see how we can verify it.

* Click on the **File** menu
* Then, choose **Preferences** option
* In the left side, click on the **Global Properties** tab.
* **#3. Removing Properties through Script:**
* This can be done by right-clicking on the property name from the property panel. Then click on the Remove option from the context menu.  
  To do this using script for removing the custom properties use the following statements for project, test suite or test case levels respectively:
* testRunner.testCase.testSuite.project.removeProperty( “Testcase\_Property” );  
  testRunner.testCase.testSuite.removeProperty( “Testcase\_Property” );  
  testRunner.testCase.removeProperty( “Testcase\_Property” );

def p=testRunner.testCase.testSuite.project

log.info p.name// my project name..//"name" is method...

def s=testRunner.testCase.testSuite

log.info s.getLabel()// getlabel...testsuite name

def t=testRunner.testCase

log.info t.getLabel()// getlabel... testcase name..

**Data driven testing:-**

import java.io.\*

import jxl.\*

def f = new File("C:\\Smoke\\Book.xls") /// saved file...

def wk = Workbook.getWorkbook(f) // creating workbook

def ws = wk.getSheet("Sheet1") // wroksheet...

r = ws.getRows() // no of rows...

for (def i=1;i<r;i++)

{

Cell c1 = ws.getCell(2,i) // (col,row)

log.info c1.getContents()// status.....

if(c1.getContents().equalsIgnoreCase("Y"))

{

Cell c2=ws.getCell(3,i)// data

testRunner.testCase.testSuite.setPropertyValue("testData",c2.getContents())

Cell c3=ws.getCell(1,i)//stepname

testRunner.runTestStepByName(c3.getContents())//call our test step..eg. step1-add-request..

}

}