**SoapUI中Groovy的使用**

SoapUI中支持使用Groovy脚本，通过Groovy脚本可以操作数据库，并可以对数据库结果进行校验，从而完成用例的检查。

1：在SoapUI中放置第三方Jar包连接的数据库的jar包需要放置于bin/ext目录下。

原文如下：

If you need to add external libraries to the soapUI classpath for your Groovy scripts (for example jdbc drivers), put these in the bin/ext folder under the soapUI installation, these will be read upon started and added to the soapUI classloader.

2： 在SoapUI的Test Case中新增数据库连接的配置，此配置的Properties文件可以直接从本地文件中导入

3：在SoapUI的Test Case中新建Groovy Script连接数据库接口如下：

def sql = Sql.newInstance(

          地址,

          用户名,

          密码,

          驱动 ）

实现样例如下：

import groovy.sql.Sql  
//通过读取配置文件连接数据库  
 def DBProperties = testRunner.testCase.getTestStepByName( "DBProperties" );  
def sql = Sql.newInstance(DBProperties.getPropertyValue( "connection-url" ),DBProperties.getPropertyValue( "sysdb-user-name" ),  
           DBProperties.getPropertyValue( "sysdb-password" ),DBProperties.getPropertyValue( "driver-class" ))

4：在SoapUI中通过Groovy脚本操作数据库

1）删除和新建表

//删除表  
try {  
   sql.execute("drop table PERSON")  
} catch(Exception e){}

//新建表  
sql.execute('''create table PERSON (  
    id integer not null primary key,  
    firstname varchar(20),  
    lastname varchar(20),  
    location\_id integer,  
    location\_name varchar(30)  
)''')

2）插入记录

插入记录有两种方式

//向表中插入记录

sql.execute("insert into PERSON (id,firstname,lastname,location\_id,location\_name) values (1,'gao','shuaihong',1,'hshen') ")

//插入记录另外一种方式  
def people = sql.dataSet("PERSON")  
people.add( firstname:"James", lastname:"Strachan", id:4, location\_id:10, location\_name:'London' )

3）查询记录

//选择一行记录  
def gaoshuaihong = sql.firstRow("select \* from PERSON where id = 1")    
log.info(gaoshuaihong.firstname)

//选择多条记录  
def allPerson  = sql.rows(" select \* from  PERSON")  
log.info(allPerson)  
log.info(allPerson[0])  
sql.eachRow("select \* from PERSON"){ row ->     
    log.info(row.lastname)  
 }

4)校验结果  
 assert allPerson[0].lastname== "shuaihong"

   SoapUI在测试WebService消息时，在构造用例过程中部分请求参数可能需要通过查询数据库或者通过随机数获取，此时可以借助Groovy脚本完成此功能，具体步骤如下：

(1)从TestCase中获取请求消息

def request = testRunner.testCase.getTestStepByName( "TestStep名称" );

def property = request.getProperty( "request" );

log.info(property.value)

(2)将请求消息转换为String,并通过XmlParser对其进行修改

def balanceQueryParser = new groovy.util.XmlParser(false,false).parseText(property.value);

//获取需要修改的Node节点，每个节点都需要指定前缀限定，通过此种方式获取的节点为NodeList，所以需要加上[0]  
def transactionid = balanceQueryParser["soapenv:Body"]["uvs:balanceQuery"]["uvs:BalanceQueryRequest"]["uvs:RequestMessage"]["uvs:MessageHeader"]["uvs:TransactionId"][0];

//通过Node.setValue方法可以对请求参数进行修改  
transactionid.setValue(System.currentTimeMillis())

(3)将请求消息写回到TestStep中  
def writer = new java.io.StringWriter();  
def printer = new groovy.util.XmlNodePrinter( new PrintWriter( writer ));  
printer.print( balanceQueryParser );  
property.setValue( writer.toString() )  
log.info(property.value)

在构造用例过程中可能需要对返回结果进行校验，此时可以借助Groovy脚本完成此功能，具体步骤如下：

(1)通过SoapUI提供的GroovyUtils获取返回的xml消息的操作XmlHolder

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

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

(2)在XmlHolder中使用Xpath获取返回字段内容  
log.info(holder.getNodeValue("//uvs:balanceQuery/uvs:BalanceQueryRequest/uvs:RequestMessage/uvs:MessageBody/uvs:SubscriberNo"));

def messageBody = holder.getDomNode("//uvs:balanceQuery/uvs:BalanceQueryRequest/uvs:RequestMessage/uvs:MessageBody")  
log.info(messageBody.getNodeValue())  
def subscriberNo = messageBody.getElementsByTagName("SubscriberNo");  
log.info(subscriberNo)

//获得节点对象的xml  
//log.info(holder.xml)

(3)如需通过xmlHolder获取其余信息参考如下：

GroovyUtils 与 XmlHolder 参考：

GroovyUtils currently includes the following (few) methods:

* projectPath : a property holding the path to the containing project, useful for accessing data files in same folder
* setPropertyValue( String testStepName, String propertyName, String value ) : sets the specified property value
* expand( string ) - expands the specified Property Expansion string
* getXmlHolder( String xmlPropertyOrString ) : Creates an XmlHolder object (see below) for easily accessing/modifying contents of an XML document using XPath expressions. The argument must either be a TestStep property in the TestStepName#PropertyName format or a valid XML string

XmlHolder object has the following methods:

* getNodeValue( String xpath ) : returns the value of the first node pointed to by the specified XPath expression (can be replaced by holder[xpath] expression, see below )
* getNodeValues( String xpath ) : returns a String array containing the values of all nodes pointed to by the specified XPath expression.
* getDomNode( String xpath ) : returns the DOM Node of the first node pointed to by the specified XPath expression.
* getDomNodes( String xpath ) : returns a DOM Node array containing all nodes pointed to by the specified XPath expression.
* setNodeValue( String xpath, String value ) : sets the content of the first node pointed to by the specified XPath expression to the specified value (can be replaced by holder[xpath] = value expression, see below )
* declareNamespace( String prefix, String namespaceURI ) : declares a namespace that will be used in a following get/set operation, can also be set with holder.namespaces[prefix] = namespaceUri (see example below)
* getNamespaces() - returns a Map of prefixes to namespace URI:s that will be used in XPath expressions
* removeDomNodes( xpath ) - removes all DOM nodes matching the specified XPath expression
* xml : property containing the updated xml string
* xmlObject : property containing the parsed XMLBeans XmlObject for the xml string
* prettyXml : property containing the pretty-printed updated xml string
* updateProperty() : if the XmlHolder was created from a TestStep property, that property will be updated with the currently held xml (see example below)
* updateProperty( boolean prettyPrint ) : same as previous, with option to pretty print the updated xml. Defaults to false when not specified.

    上面介绍了使用writer,printer修改请求的方法，SoapUI Groovy提供了更方便的方法，GroovyUtils 与 XmlHolder ,具体如下：

//获取GroovyUtil

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

//获取XmlHolder

def holder = groovyUtils.getXmlHolder( "TestStep#Request" )

//def holder = groovyUtils.getXmlHolder( "TestStep#Response" )

holder.declareNamespace("com", "<http://gsh/common>")

//这里的命名空间需要事先指定，xmlPath中不需要带Soap包装

holder.setNodeValue( "//com:NewSub/RequestHeader/com:TransactionId", "1" );

//将请求写入TestStep

groovyUtils.setPropertyValue( "TestStep", "Request", holder.prettyXml );

在SoapUI中可以定义一个个的测试用例TestCase，但是有些用例是依赖于之前的用例的，如果纯拷贝的话可能会导致用例比较臃肿而且不好维护，比如说存在如下两个TestCase:

1)CreateUserTestCase:测试创建用户，通过发送Soap报文方式创建用户同时需要校验数据库中值是否正确；

2)ChangUserInfoTestCase:测试修改用户信息，通过发送Soap报文方式修改用户信息，需要校验修改前和修改后的用户信息ChangUserInfo之前必须得创建一个用户，纯拷贝肯定是不可取的，因为后续如果创建用户的接口稍有变动，则需要同时在ChangUserInfoTestCase和CreateUserTestCase修改请求报文。

SoapUI在TestCase中提供Run TestCase的Step，可以直接调用指定的TestCase，但是需要前一个TestCase中将属性传递出来，步骤如下：

1）在被调用TestCase中设置返回属性

    testRunner.testCase.setPropertyValue("属性名称",“属性值”)

2）在调用TestCase中增加Run TestCase指向被调用TestCase

3）在调用TestCase中的其它Test Step中获取属性

例如：在CreateUserTestCase中将创建好的用户ID传给ChangUserInfoTestCase，则步骤如下：

1）在CreateUserTestCase中通过Groovy Script 设置返回属性：

     testRunner.testCase.setPropertyValue("UserID",context.getProperty("UserID"))

2) 在ChangUserInfoTestCase中增加Run TestCase:RunNewUserTestCase指向CreateUserTestCase并指定UserID属性为输入值

3）在ChangUserInfoTestCase中获取执行CreateUserTestCase得到的用户ID

def NewUserProperties = testRunner.testCase.getTestStepByName( "RunNewUserTestCase" );  
log.info(NewUserProperties .getPropertyValue( "UserID" ))