**What is SoapUI:**

It is API testing tool for manual and automation testing of SOAP and REST API’s.

**Why to use SoapUI:**

1. To create quick and efficient API tests

2. To create API functional, performance and security tests

3. To create API Testing automation framework

**Download SoapUI from -** <https://www.soapui.org/downloads/soapui/>

**Install SoapUI:**

**Installation location: C:\Program Files\SmartBear\SoapUI-5.7.2**

**What Is WSDL:**

WSDL is xml document which contains all details about Web Service and all the API requests there in web service.

**Sample WSDL URL:** <http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso?WSDL>

**Hierarchy in SOAP UI:**

Project -> Test Suite -> Test Case -> Test Step

A screenshot of a computer

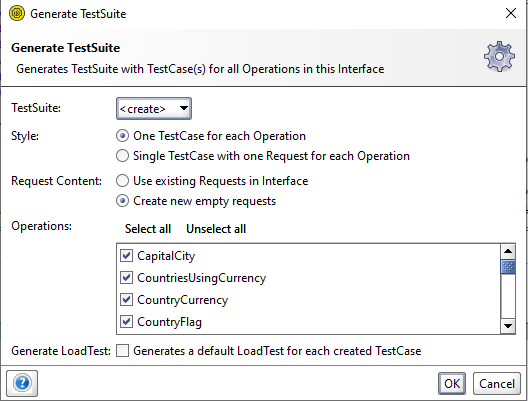
Description automatically generated

**How To Create New SOAPUI Project And Add WSDL URL:**

1. Launch SOAP UI Application
2. Click File -> New SOAP Project OR click on SOAP Icon 
3. Enter Project Name and enter WSDL URL
4. Click OK

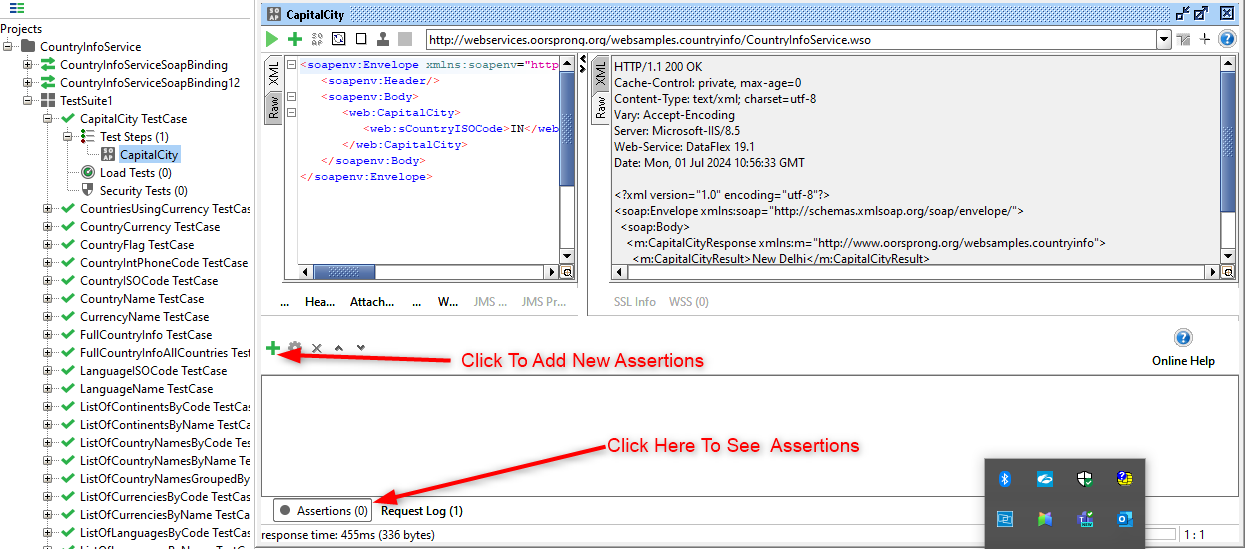
**How to create Test Suite - Test Cases:**

1. Right Click on WSDL and select “Generate TestSuite”
2. Keep Default configuration
3. Click OK
4. Enter Name of Test Suite
5. TestSuite will be created which will have separate test case for each request.
6. If we selected option **“Single TestCase with one Request for each Operation”** then single test case will be created with all requests inside that single test case.

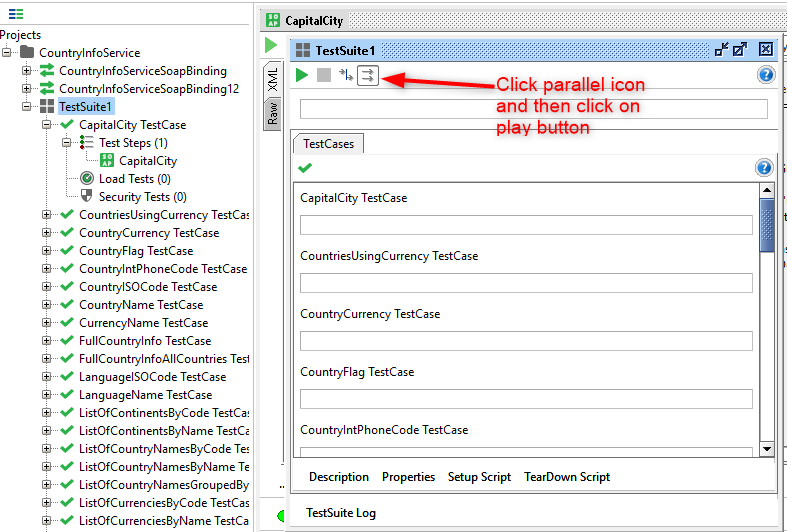


**How To Add Assertions:**

1. Select Test Step TO Which you want to add assertions.
2. Click On **Assertions**
3. Click On **“+” symbol**
4. Click on Add



**How to run Run Test Step ,Test Case , Test Suite and run Test Cases In Parallel:**

1. Select Test Suite or Test Case or Test Step whichever you want to run and click on play button.
2. To run test cases in Test Suite in parallel select parallel icon and click on play button  
     
   

**How To Create New REST API Project:**

1. Launch SOAP UI Application
2. Click File -> New REST Project OR click on REST Icon 
3. Enter URI
4. Click OK
5. You will see new REST API project created
6. You can get sample REST API’S from : <https://restcountries.com/>

**How to add new REST API Endpoints in existing project:**

1. Select REST API Project
2. Right click and select “New REST Service from URI”
3. Enter URI
4. Click OK

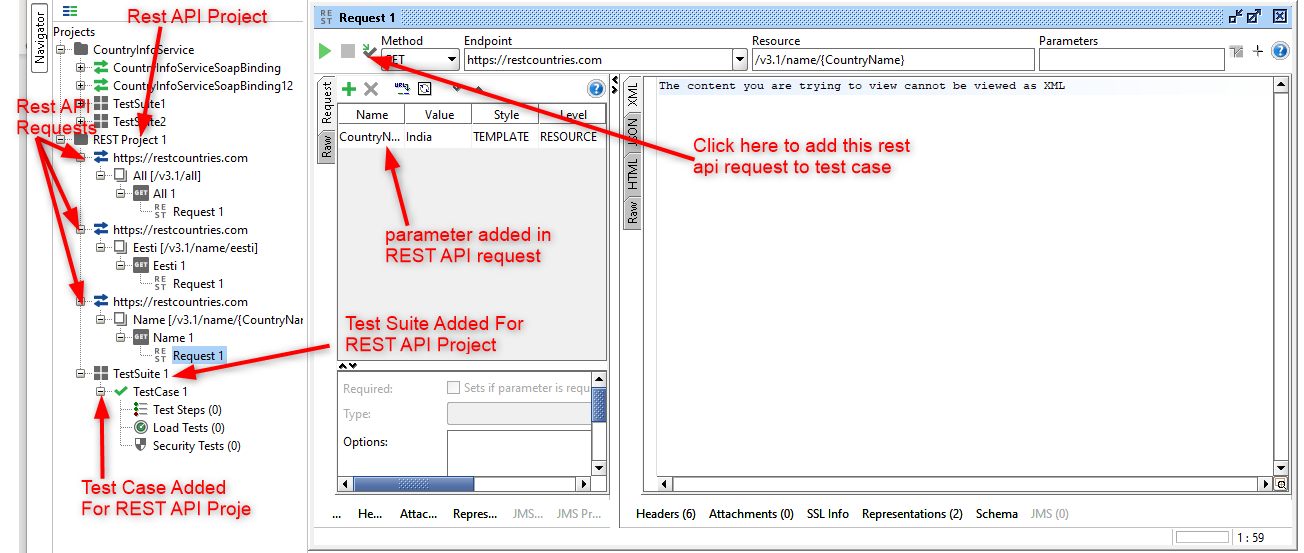
**How to Add Parameters in request:**

1. Double click on request
2. Click on “+” icon   
   A screenshot of a computer

   Description automatically generated
3. Add parameter name ,value and select type of parameter from **Style** dropdown

**How to create TestSuite and TestCase in REST API Project:**

There are two ways to create TestSuite and TestCases

1. By Creating TestSuite and TestCase by right clicking on project:
2. Right Click on Project
3. Select **New TestSuite**
4. Specify TestSuite name
5. Right Click Created TestSuite and select New TestCase
6. Specify TestCase name
7. Select any API Request -> click on icon to add request to created test case:  
     
   
8. Select TestSuite and TestCase name from dropdown and click Ok
9. By Clicking on icon in above screenshot in request:
10. Select and open any request
11. Click on Icon 
12. From dropdown select **“Create new TestSuite”**
13. Click “Ok”
14. Specify TestSuite name
15. Click “Ok”
16. Specify TestCase name
17. Click “Ok”
18. Specify request name
19. Click “Ok”

**What is Properties in SOAPUI:**

Properties can be used as variables to store values that can be referred in testing

Properties can be accessed at following levels:

Project - ${#Project#PropertyName}

TestSuite - ${#TestSuite#PropertyName}

TestCase - ${#TestCase#PropertyName}

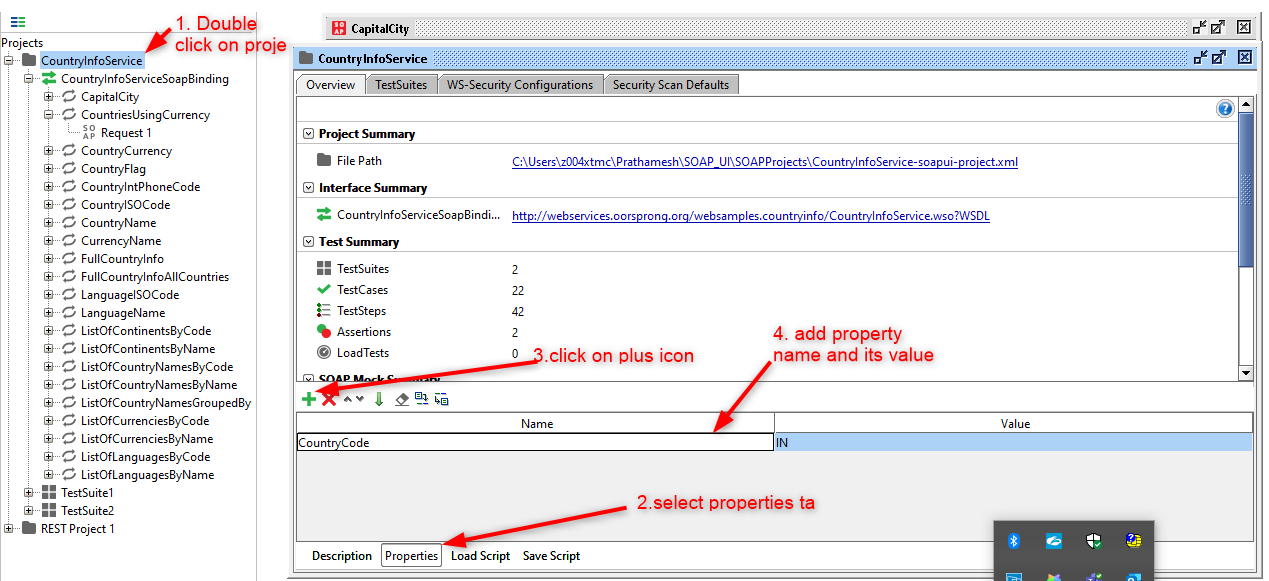
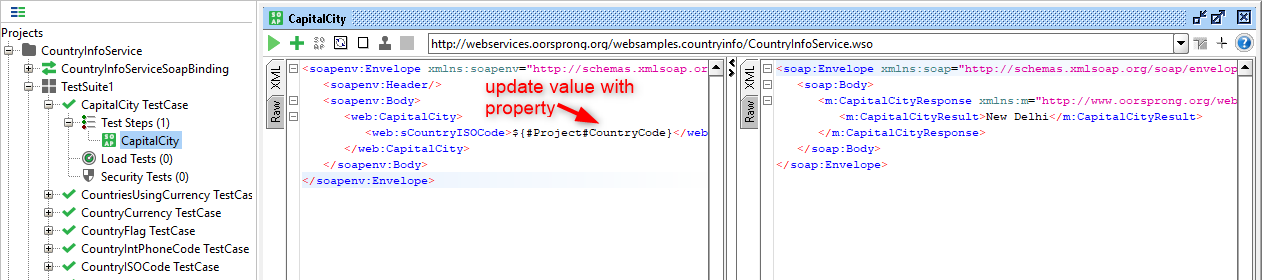
TestStep - ${TestStepName#PropertyName} -> here we will not use first #

System - ${#System#PropertyName}

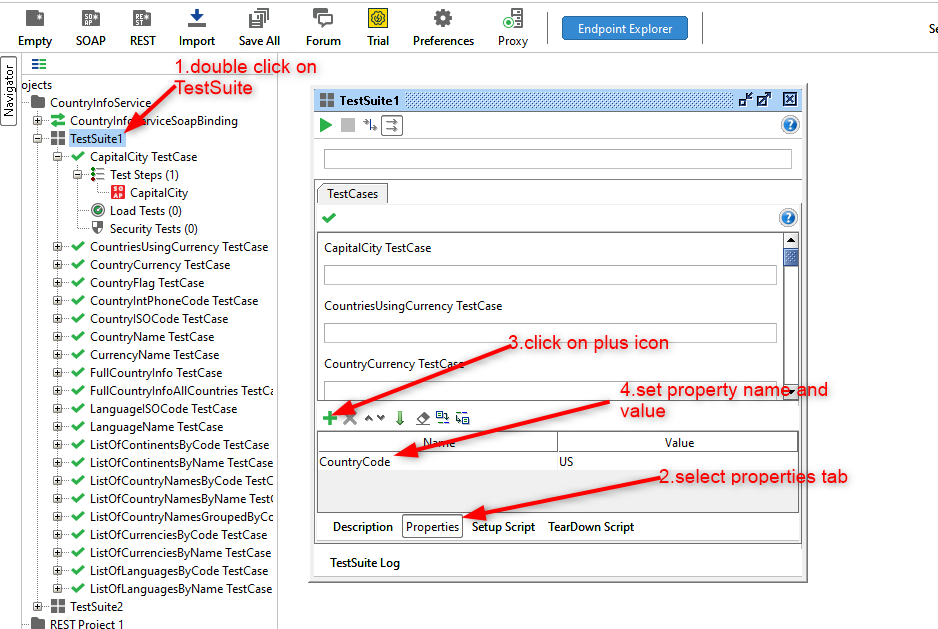
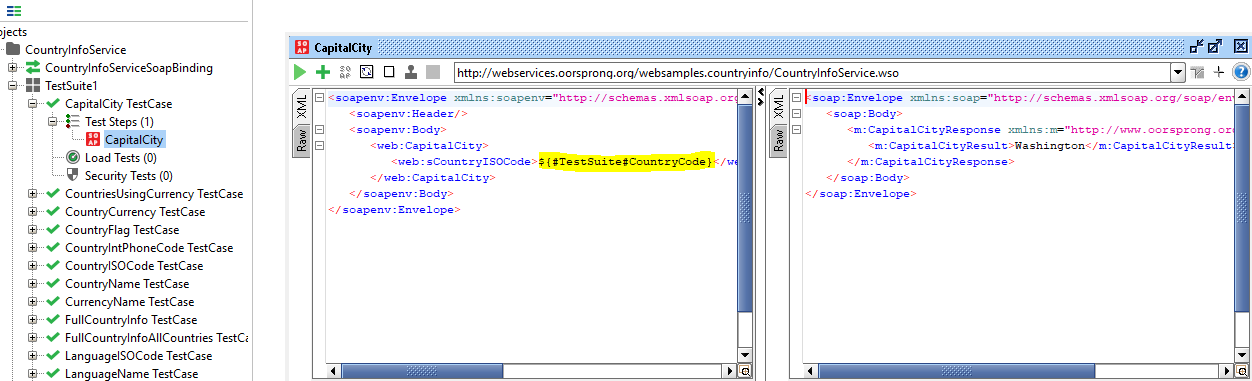
Env - ${#Env#PropertyName}

Global - ${#Global#PropertyName}

**How to add properties at project level and use it in request:**

1. Double Click on project
2. Click on properties tab
3. Click on “+” sign
4. Add property name and its value  
     
   
5. Open test step in TestSuite and set value as ${#Project#CountryCode}  
   

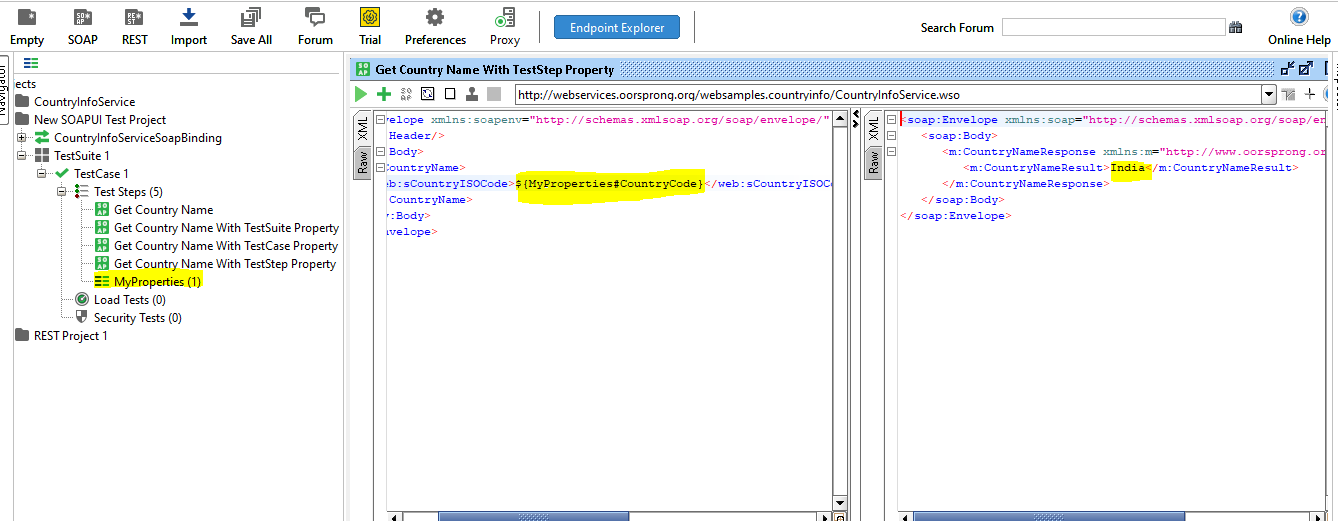
**How to add properties at TestSuite level and use it in request:**

1. Double Click on TestSuite
2. Click on properties tab
3. Click on “+” sign
4. Add property name and its value  
     
   
5. Open test step in TestSuite and set value as ${#TestSuite#CountryCode}  
     
   

Similarly we can set properties for TestCase

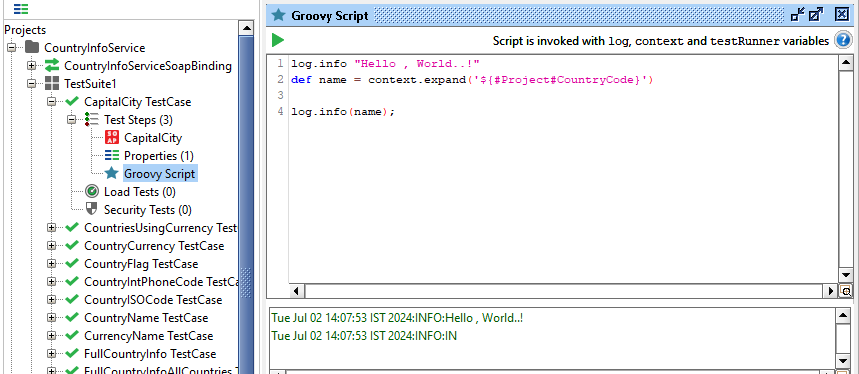
**How to add properties at TestStep level and use it in request:**

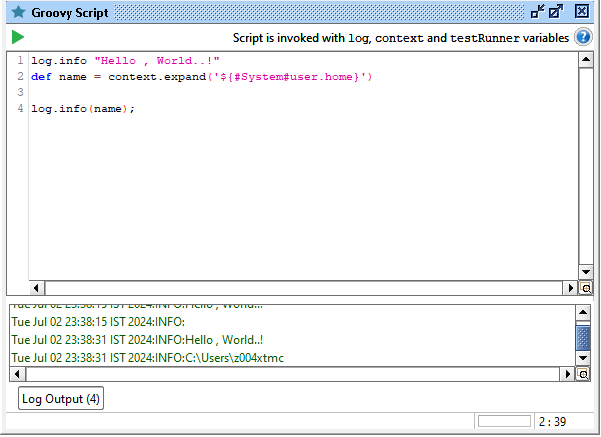
1. Select TestCase and right click
2. Select menu “Add Step” -> “Properties”
3. Give name to properties
4. Open added Property , click on “+” icon and add property name and value
5. Use this property in TestStep using string : ${Properties#CountryCode}  
     
   A screenshot of a computer

   Description automatically generated  
     
   

**How to Add Groovy script:**

Right click on TestCase or TestStep -> Select Menu **“Add Step”** -> **“Groovy Script”**



We can get system properties as well as per below screenshot in groovy script:  
  


**How to Get and Set Properties through groovy script:**

TestRunner is Interface which which provides entrypoint to SOAPUI API.

1. **getPropertyValue:**  
    to get value of property.
2. **addProperty:**

With addProperty we can add only key not value.

1. **setProperty:**

With setProperty we can add key as well as value OR we can set value to existing property.

1. **removeProperty:**

With remove we can remove property key and value.

1. **Loop through property:**

/\*\*\*\*\*\*\*\*\* GET Property \*\*\*\*\*\*\*\*\*\*\*\*/

//Project Level

def ProjectProperty = testRunner.testCase.testSuite.project.getPropertyValue("Name")

//TestSuite Level

def TestSuiteProperty = testRunner.testCase.testSuite.getPropertyValue("Name")

//TestCase Level

def TestCaseProperty = testRunner.testCase.getPropertyValue("Name")

//TestStep Level

def TestStepProperty = testRunner.testCase.getTestStepByName("CapitalCityProperties").getPropertyValue("Name")

/\*\*\*\*\*\*\*\*\* SET Property \*\*\*\*\*\*\*\*\*\*\*\*/

//Project Level

testRunner.testCase.testSuite.project.setPropertyValue("Name1","Test1")

//TestSuite Level

testRunner.testCase.testSuite.setPropertyValue("Name","Test2")

//TestCase Level

testRunner.testCase.setPropertyValue("Name","Test3")

//TestStep Level

testRunner.testCase.getTestStepByName("CapitalCityProperties").setPropertyValue("Name","Test4")

/\*\*\*\*\*\*\*\*\*\*\* add Property \*\*\*\*\*\*\*\*\*\*\*\*\*\*/

testRunner.testCase.testSuite.project.addProperty("Namee")

/\*\*\*\*\*\*\*\*\*\*\* remove Property \*\*\*\*\*\*\*\*\*\*\*\*\*\*/

testRunner.testCase.testSuite.project.removeProperty("Namee")

/\*\*\*\*\*\*\*\*\*\* loop through properties \*\*\*\*\*\*\*\*\*\*\*\*\*\*/

testRunner.testCase.testSuite.project.properties.each

{

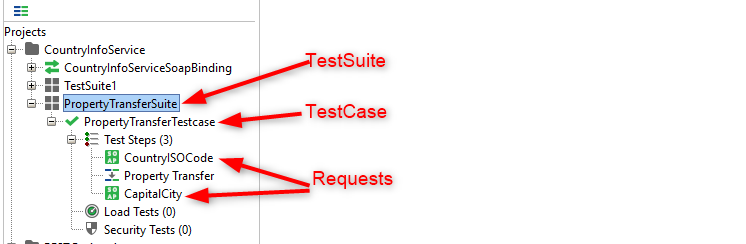
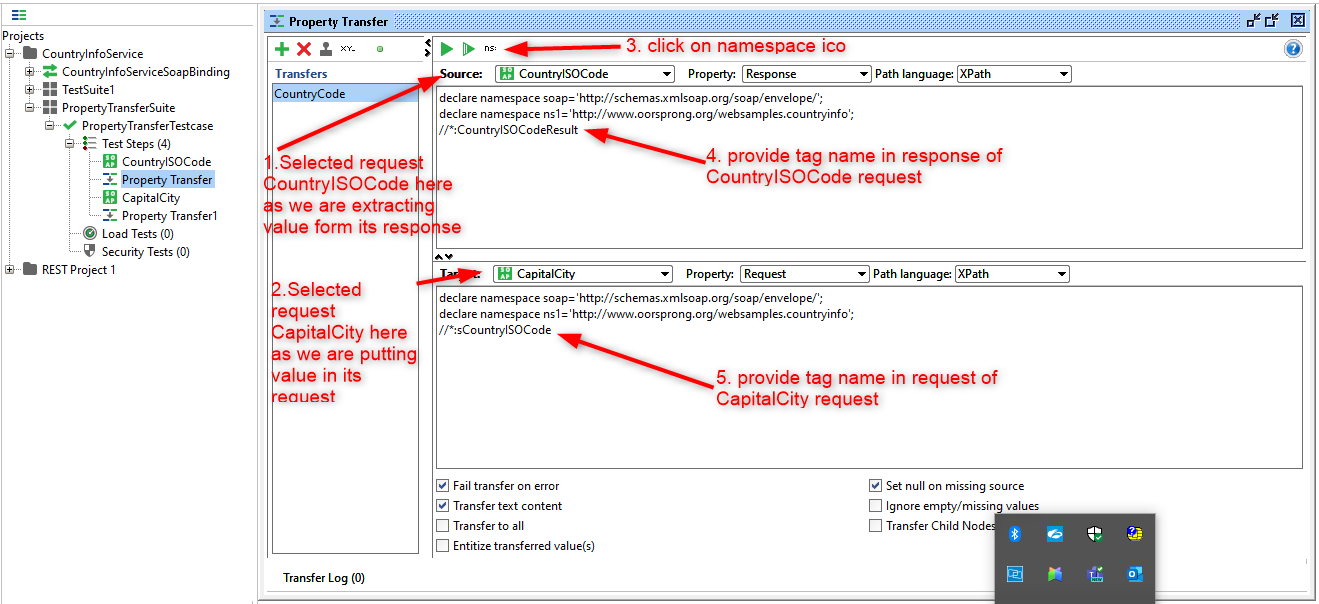
key,value ->

log.info(testRunner.testCase.testSuite.project.getPropertyValue(key))

}

**Property Transfer and Value Transfer:**

**How to send values from the response of one api to request of another api:**

1. Create new TestSuite
2. Create new TestCase inside TestSuite
3. Add requests inside TestCase which are dependent on each other  
     
   
4. Right click on TestCase select menu **“Add Step”** andSelect **“Property Transfer”**
5. Provide name to property transfer
6. Click on “+” icon on Property Transfer window and give name to Add transfer.
7. In source and Target select proper values from dropdown.  
     
     
     
   A screenshot of a computer

   Description automatically generated  
     
   A screenshot of a computer

   Description automatically generated
8. Add property at TestCase level and update request accordingly

**TestRunner Interface link:** [**https://www.soapui.org/apidocs/5.5.0/com/eviware/soapui/model/testsuite/TestRunner.html**](https://www.soapui.org/apidocs/5.5.0/com/eviware/soapui/model/testsuite/TestRunner.html)

**Context class:** [**https://www.soapui.org/apidocs/5.5.0/com/eviware/soapui/impl/wadl/inference/schema/Context.html**](https://www.soapui.org/apidocs/5.5.0/com/eviware/soapui/impl/wadl/inference/schema/Context.html)

**log.info testRunner.metaClass.methods\*.name.unique().sort() 🡪 this will give us all methods available in testrunner class**

**How to Run request from GUI, Groovy and Command Line:**

1. **How to Run Request from GUI:**Select Request from TestCase whichever you want to run and click on play button.
2. **How to Run Request from Groovy Script:**
3. **To run Request where groovy script and TestStep are in same test case:**def Status = testRunner.runTestStepByName("CountryISOCode")

def result = Status.getStatus().toString()

log.info(result)

A screenshot of a computer

Description automatically generated

1. **To run Request where groovy script and TestStep are in different test case:**def Project = testRunner.testCase.testSuite.project

def TestCase = Project.testSuites["PropertyTransferSuite"].testCases["TestCase 2"]

def TestStep = TestCase.getTestStepByName("CapitalCity")

def Status = TestStep.run(testRunner,context)

def result = Status.getStatus().toString()

log.info(result)  
  
A screenshot of a computer

Description automatically generated

1. **We cannot run Request from command line**

# **How to Run TestCase from GUI, Groovy and Command Line:**

1. **How to Run TestCase from GUI:**Select TestCase whichever you want to run and click on play button.  
   When you open test case there will be option to run, stop, loop TestCase.  
   There is also settings button from where we have different options.  
   In Settings if we check option **“Abort on Error”** in that case when any request fails then aborts execution.
2. **How to run TestCase using groovy:**A screenshot of a computer program

   Description automatically generated

//groovy code to run test case

def tCase = testRunner.testCase.testSuite.testCases["Assertions"]

runner = tCase.run(new com.eviware.soapui.support.types.StringToObjectMap(), false);

log.info(runner.getStatus().toString())

log.info(testRunner.testCase.name)

//groovy code to loop test cases in a test suite

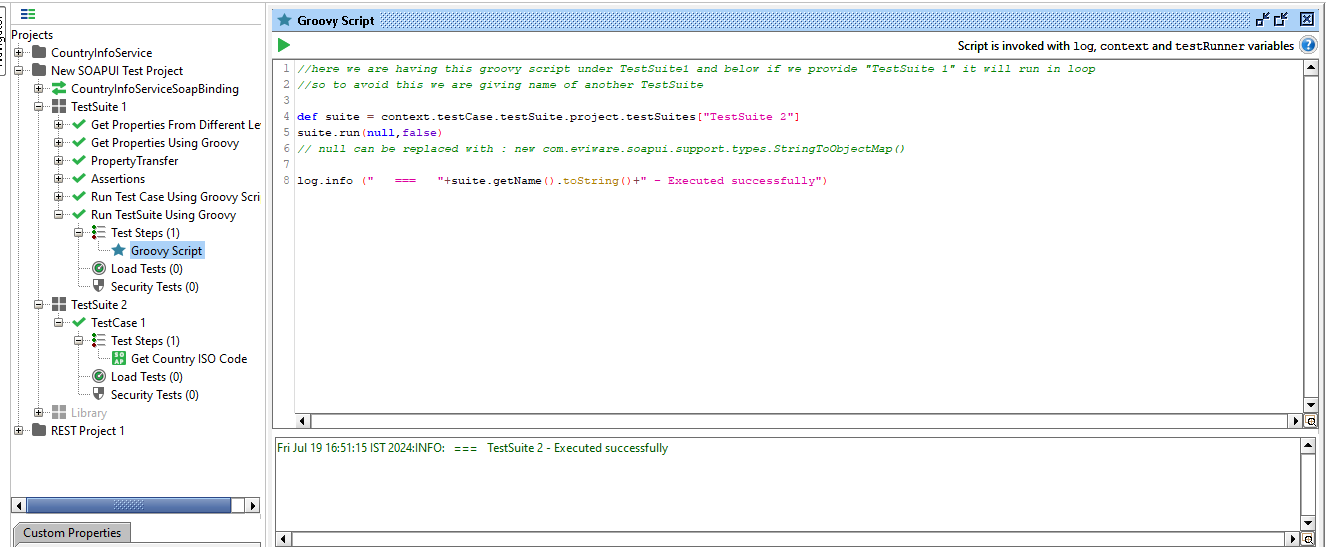
def testCases = context.testCase.testSuite.getTestCaseList()

testCases.each{

log.info(it.name)

}

# **How To Run TestSuite from UI,Groovy and Cmd line:**



//here we are having this groovy script under TestSuite1 and below if we provide "TestSuite 1" it will run in loop

//so to avoid this we are giving name of another TestSuite

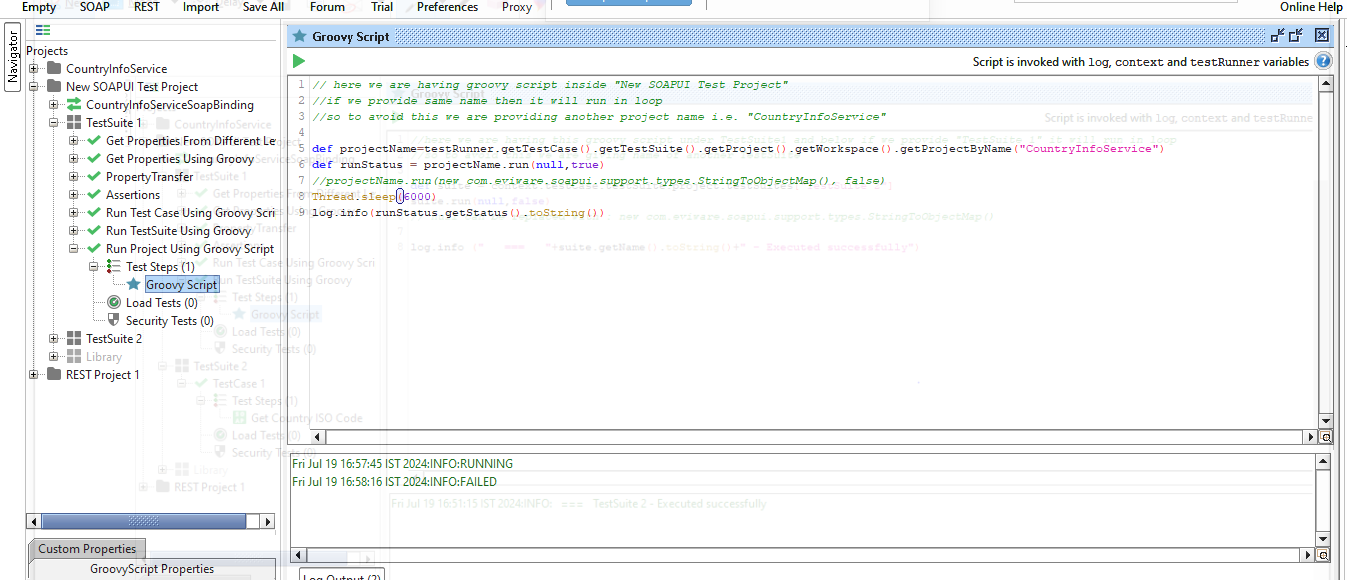
def suite = context.testCase.testSuite.project.testSuites["TestSuite 2"]

suite.run(null,false)

// null can be replaced with : new com.eviware.soapui.support.types.StringToObjectMap()

log.info (" === "+suite.getName().toString()+" - Executed successfully")

# **How To Run Project from UI,Groovy and Cmd line:**



// here we are having groovy script inside "New SOAPUI Test Project"

//if we provide same name then it will run in loop

//so to avoid this we are providing another project name i.e. "CountryInfoService"

def projectName=testRunner.getTestCase().getTestSuite().getProject().getWorkspace().getProjectByName("CountryInfoService")

def runStatus = projectName.run(null,true)

//projectName.run(new com.eviware.soapui.support.types.StringToObjectMap(), false)

Thread.sleep(6000)

log.info(runStatus.getStatus().toString())

# **Logs IN SOAPUI:**

# **SetUp and Teardown:**

**groovy code to set and get setup and teardown scripts :**

testRunner.testCase.testSuite.project.getTestSuiteByName('TestSuite1').getTestCaseByName('TestCase1').setSetupScript('log.info "setup"') testRunner.testCase.testSuite.project.getTestSuiteByName('TestSuite1').getTestCaseByName('TestCase1').setTearDownScript('log.info "teardown"')

log.info (" -- "+testRunner.testCase.getSetupScript());

log.info (" -- "+testRunner.testCase.getTearDownScript());

**groovy code can be given at project load script to load setup and teardown scritps from file:**

def su = new File("setup.txt").text

def td = new File("teardown.txt").text

project.getTestSuiteList().each { testSuite ->

testSuite.getTestCaseList().each { testCase ->

testCase.setSetupScript(su)

testCase.setTearDownScript(td)

} }

# **Assertions:**

Contains

XPath Match

XQuery Match

Compliance

JSON Path assertions

SLA (Service Level Agreement) 🡪 we can add maximum response time

# **How To Create Script Assertions:**

Script assertions worked on the last response received on SOAP UI.

It works with messageExchange object.

**Script Assertion samples For SOAP :**

//check response time

assert messageExchange.timeTaken < 4000

//check for Endpoint

log.info messageExchange.getEndpoint()

//check for TimeTaken

log.info messageExchange.getTimeTaken()

//check for header

log.info (messageExchange.responseHeaders["Content-Length"])

assert messageExchange.responseHeaders["Content-Length"] != null

//check attachments

assert messageExchange.responseAttachments.length == 0

log.info (messageExchange.responseAttachments.length)

//validate response nodes

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

def requsetHolder = groovyUtils.getXmlHolder( messageExchange.requestContent )

def responseHolder = groovyUtils.getXmlHolder( messageExchange.responseContent )

def refNum = responseHolder.getNodeValue("//m:CountryCurrencyResult/m:sName")

assert responseHolder.getNodeValue("//m:CountryCurrencyResult/m:sName") == "Rupees"

//to get response

def resp = messageExchange.responseContentAsXml.toString()

**For JSON response (For REST API):**

//get json response

import groovy.json.JsonSlurper

def responseMessage = messageExchange.response.responseContent

def json = new JsonSlurper().parseText(responseMessage)

//assert node values

log.info json.name

assert json.capital.toString() != null

assert json.name.toString() == "[Taiwan]"

testStepName = messageExchange.modelItem.testStep.name

//to get the Test Step Name

log.info testStepName

xmlHold = messageExchange.responseContentAsXml.toString() //to store the response as Xml string

# **How To Run SOAP UI Tests Using Jenkins:**

We can run SOAP UI test cases from cmd from location 🡪 **C:\Program Files\SmartBear\SoapUI-5.7.2\bin\**

**We can use below commands:**

**To run project:**

testrunner.bat C:\Users\z004xtmc\Prathamesh\SOAP\_UI\SOAPProjects\New-Test-Project-soapui-project.xml

**To run TestSuite:**

testrunner.bat -s"TestSuite 1" C:\Users\z004xtmc\Prathamesh\SOAP\_UI\SOAPProjects\New-Test-Project-soapui-project.xml

**To run TestCase:**

testrunner.bat -s"TestSuite 1" -cAssertions C:\Users\z004xtmc\Prathamesh\SOAP\_UI\SOAPProjects\New-Test-Project-soapui-project.xml

**As per our requirement we can configure Jenkins job and we can provide below code in windows batch command:**

cd “**C:\Program Files\SmartBear\SoapUI-5.7.2\bin\**”

testrunner.bat C:\Users\z004xtmc\Prathamesh\SOAP\_UI\SOAPProjects\New-Test-Project-soapui-project.xml

# **How to create API Monitor:**

# **How To Add SOAP UI Projects to Git:**

Step 1 : Download and Install Git

Step 2 : Create GitHub account

Step 3 : Add SoapUI Project to Git

Goto Command Prompt

Goto location of SoapUI Project

git init

git status

git add

git commit -m "commit message"

git push -u url master

# **Request Response Chaining with JSON:**

**Video 21**

# **How to extract XML response and use in next API request:**

Video 22

# **How to create CSV report in SOAP UI Open Source:**

**Follow steps in this article:**

[**https://dzone.com/articles/how-to-achieve-csv-reporting-in-soapui-open-source-1**](https://dzone.com/articles/how-to-achieve-csv-reporting-in-soapui-open-source-1)

**we can customize scripts on this page as per our requirements.**

**Imp Links:**

**Create class , method in groovy:** [**click here**](https://www.javatpoint.com/soapui-groovy-script#:~:text=To%20create%20a%20class%20in%20SoapUI%2C%20right%2Dclick%20on%20TestStep,following%20code%2C%20as%20shown%20below.&text=Click%20on%20the%20run%20button%20to%20execute%20the%20groovy%20script.)

**Generate HTML and CSV report:** [**https://www.coderscamp.tech/post/generate-reports-in-soap-ui-free-version**](https://www.coderscamp.tech/post/generate-reports-in-soap-ui-free-version) **https://softwaretestersforum.blogspot.com/2013/03/generate-junit-style-html-reports-in.html**

**DB Connection:** [**https://webautomationtestingblog.wordpress.com/2018/06/13/steps-to-connect-to-postgresql-database-from-soapui/**](https://webautomationtestingblog.wordpress.com/2018/06/13/steps-to-connect-to-postgresql-database-from-soapui/)

1. **Download postgres JDBC driver jar file 42.7.3 and put it under C:\Program Files\SmartBear\SoapUI-5.7.2\lib**[**https://jdbc.postgresql.org/download/**](https://jdbc.postgresql.org/download/)
2. **Download JDK from here and set env variables:**[**https://www.oracle.com/in/java/technologies/downloads/#jdk22-windows**](https://www.oracle.com/in/java/technologies/downloads/#jdk22-windows)
3. **Use below script to make connection:**

import groovy.sql.Sql

// Database connection parameters

def dbUrl = 'jdbc:postgresql://localhost:5432/postgres'

def dbUser = 'postgres'

def dbPassword = 'admin123'

def dbDriver = 'org.postgresql.Driver'

def sql = null

try {

// Load the PostgreSQL JDBC driver

Class.forName(dbDriver)

// Establish the database connection

sql = Sql.newInstance(dbUrl, dbUser, dbPassword, dbDriver)

// Execute a query and process the results

def query = 'SELECT \* FROM bt\_mytc\_executionreporttable'

sql.eachRow(query) { row ->

log.info "Row: ${row}"

// Access specific columns

// log.info "Column1: ${row.column1}, Column2: ${row.column2}"

}

} catch (Exception e) {

log.error "Error: ${e.message}"

} finally {

// Close the database connection

if (sql != null) {

sql.close()

}

}

**Our SOAP UI Project Overview:**

**TC\_Login**

**| -- TestSuite1**

**| -- Prerequisite**

**| -- TC\_Login**

**| -- TC\_Logout**

**| -- Libraries**

**| -- commonMethods**

**| -- Reporting\_Utility**

**| -- SoapRequests**

**| -- TC\_Login**

**| -- TC\_Logout**