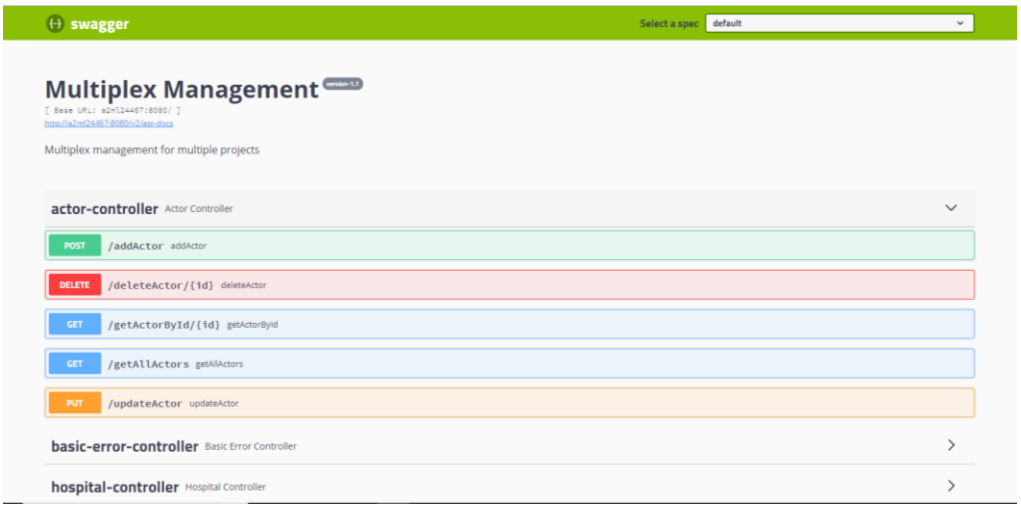
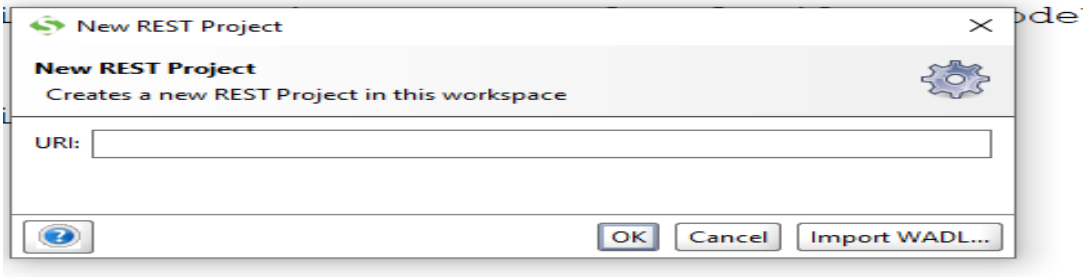
IMPLEMENTATION OF GET,PUT,POST AND DELETE OPERATIONS WITH VALIDATION IN A SINGLE GROOVY SCRIPT

Here I have used actor-controller as an example to display all the operations with validation in a single groovy script.

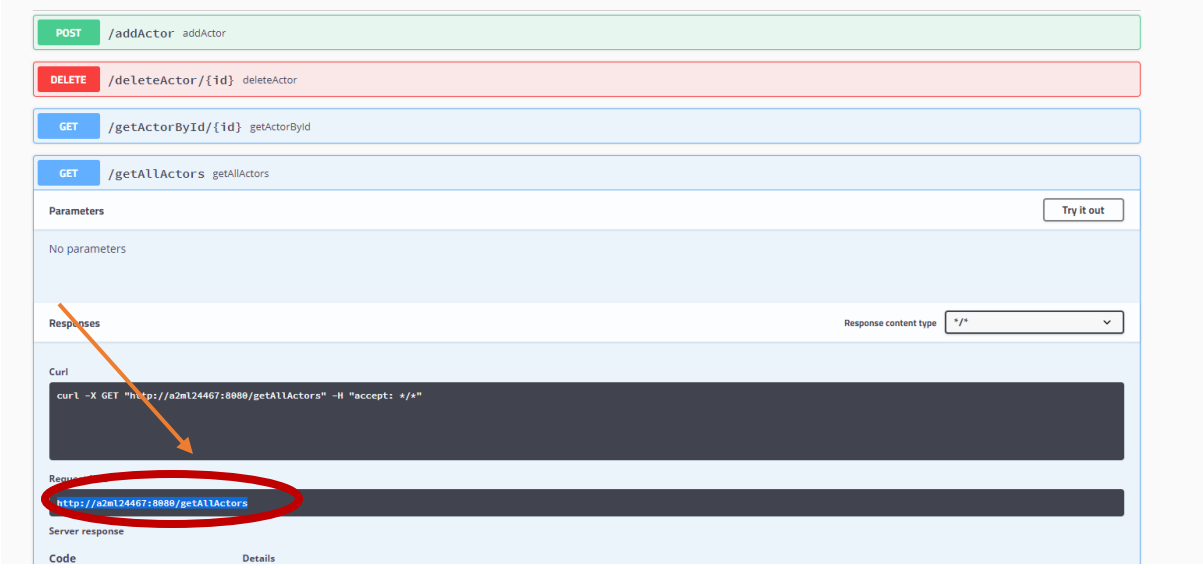
http://a2ml24467:8080/swagger-ui.html#/actor-controller



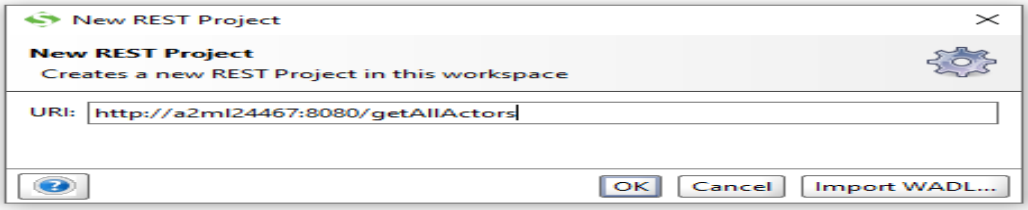
Step 1>Open soapUi and Click on the REST project present at the top of the application. You will get this.



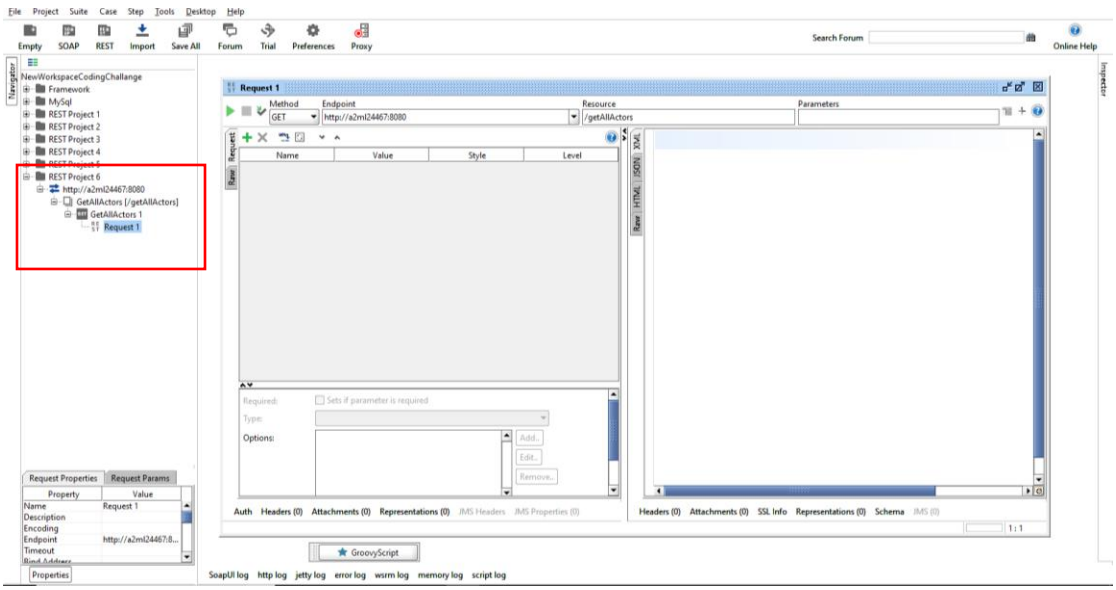
Step 2>Now go to the swagger link provided above->click GET/getAllActors->Click on try it out->Click Execute and then copy the link as shown in the figure.



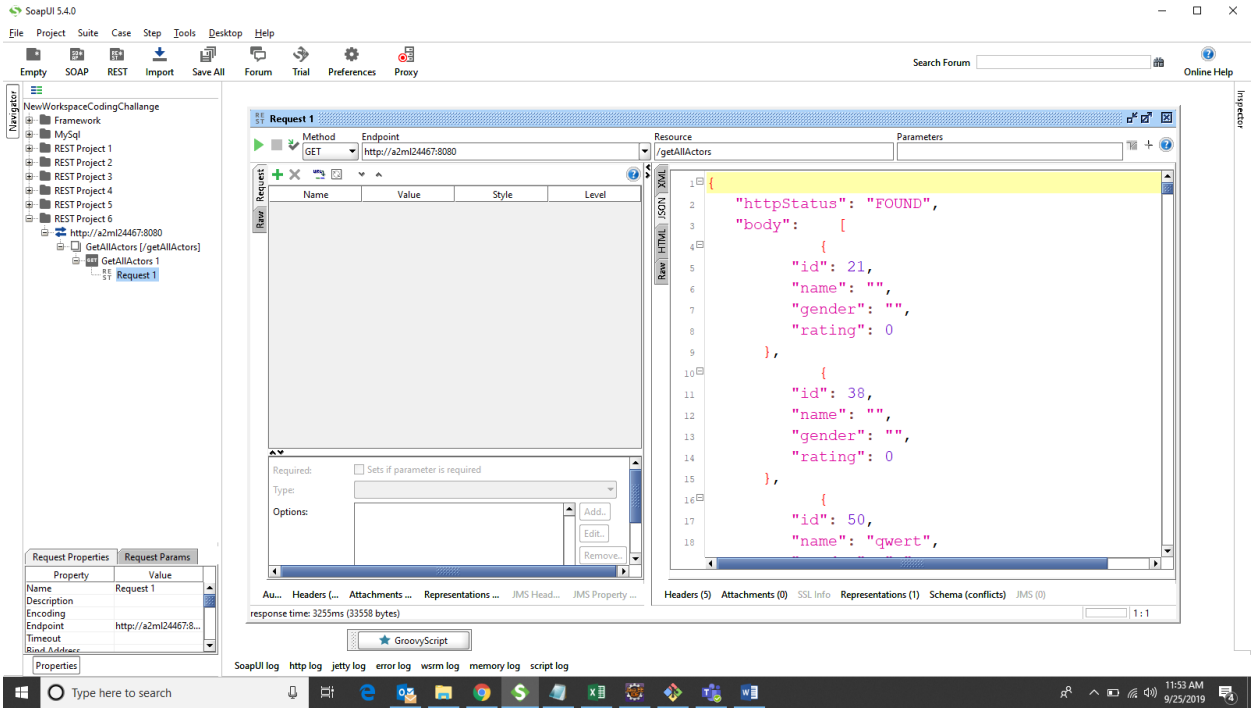
Step 3>Now open soapui and post it in the tsxt box of URI and then click ok.



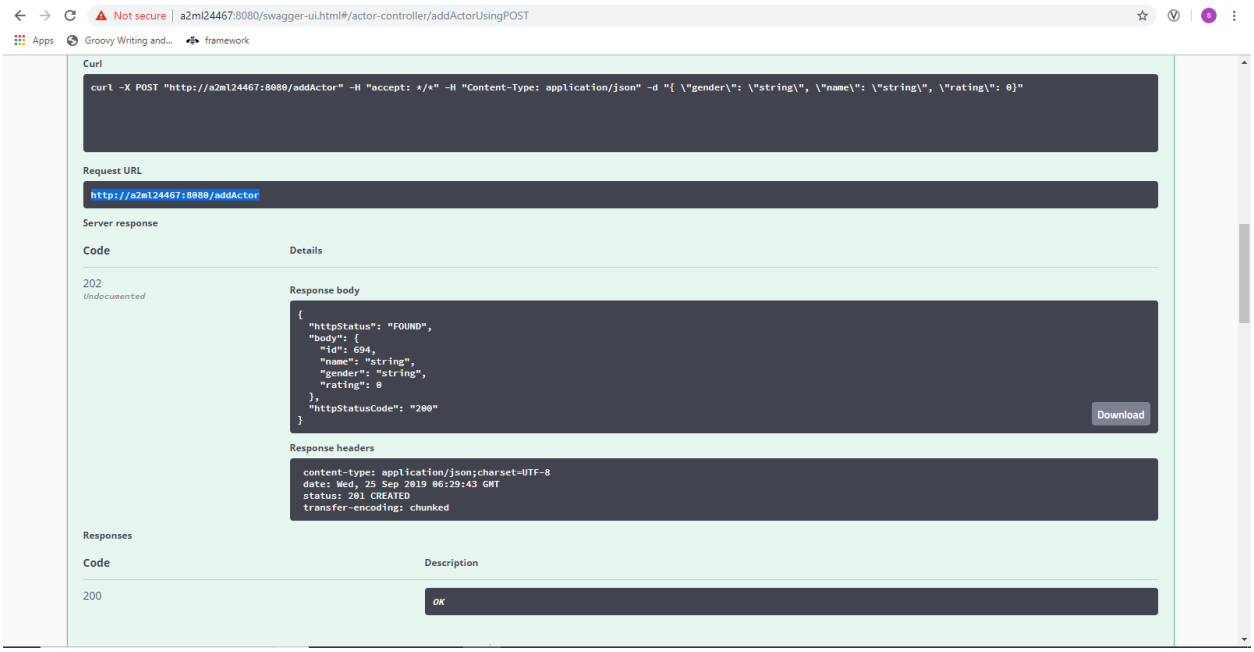
Step 4>You will get this as a folder structure as shown in the figure.



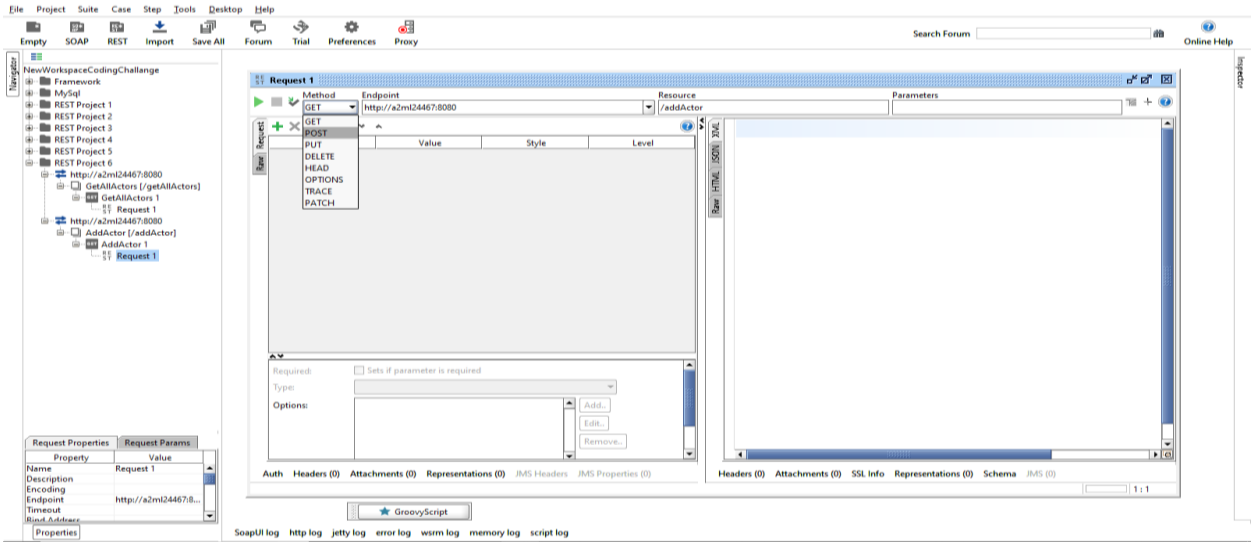
Step 5>Click on the green play button to see the output.



Step 6>Then right click on the project name and select “new Rest service from URI.Then go to the swagger link->click on POST-> Click on try it out->Execute and then copy the URL.

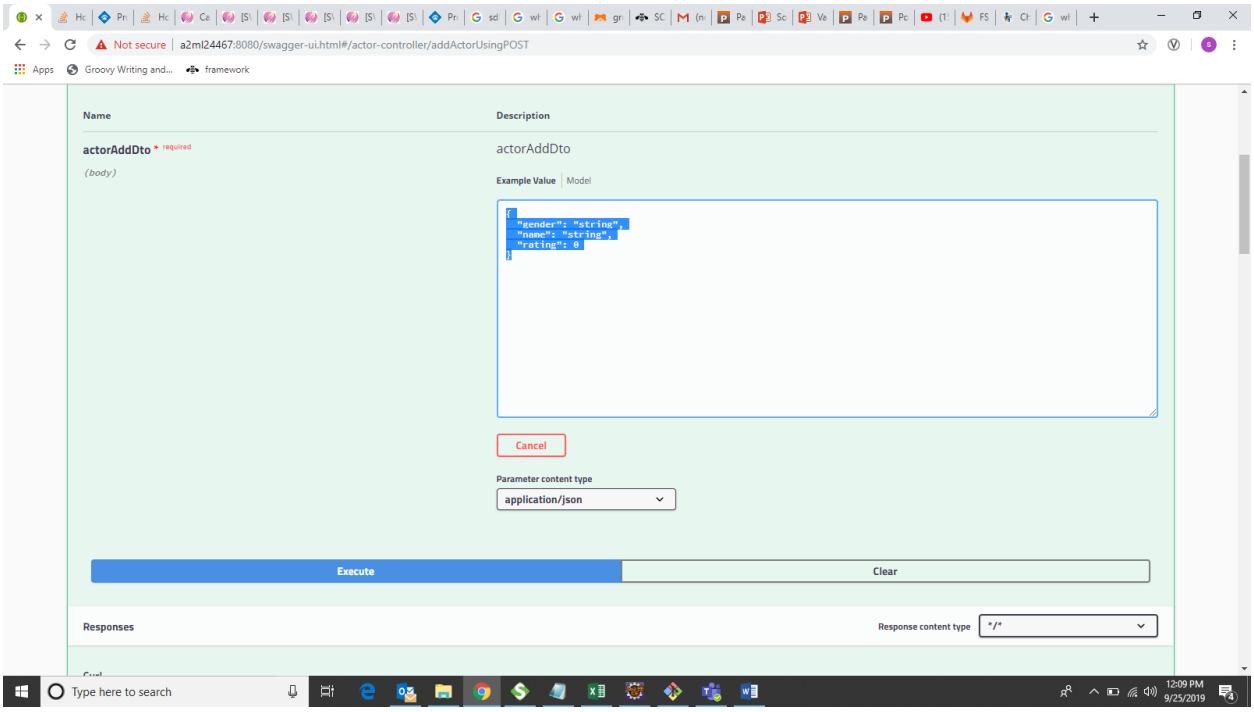


Step 7>Paste it in the textbox of URI and click OK. Then change the method from GET to POST as shown in the figure. And your folder structure will look like this.



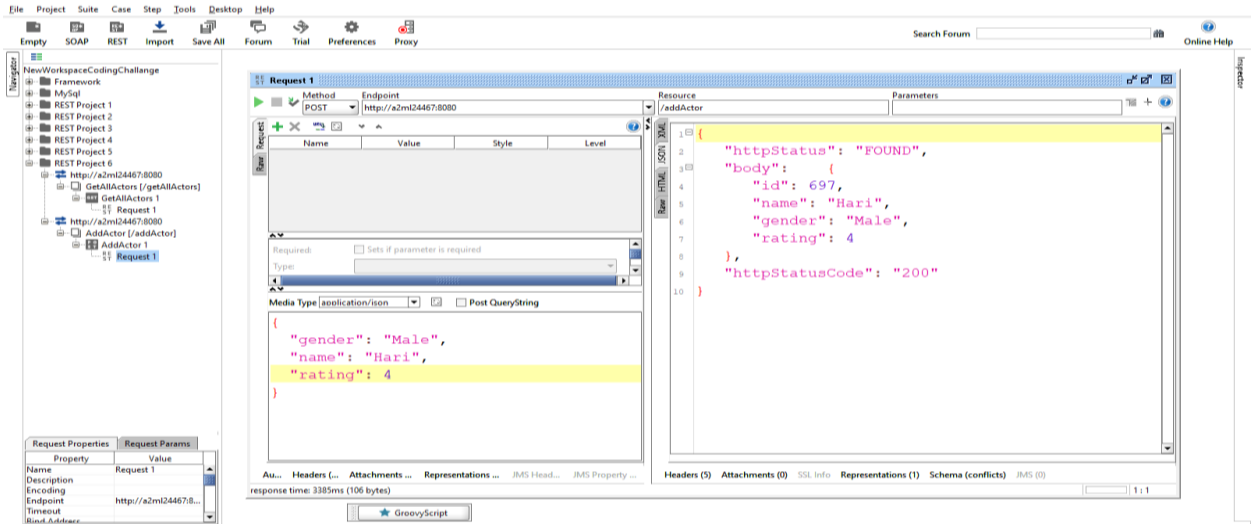
Step 8>Now go to the swagger link again and copy the body.

As shoen in the figure.



Step 9>In the post method there is one space below as Media-Type.Paste it there. And made the changes accordingly in the attributes values. Then click on the run button.

You will get the json response as shown below.



Step 10>Similarly in the same way create PUT and delete.

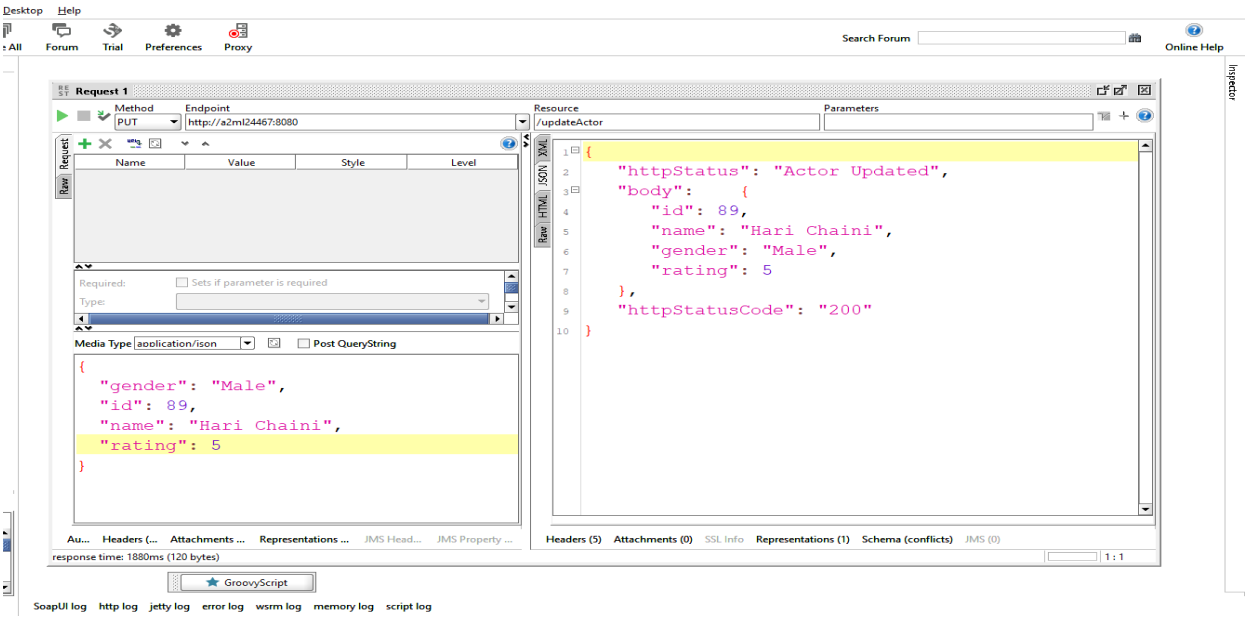
In PUT first copy the link paste it and then copy the body and make changes accordingly.

In DELETE copy the link and give the ID for which you want to perform the delete operation.

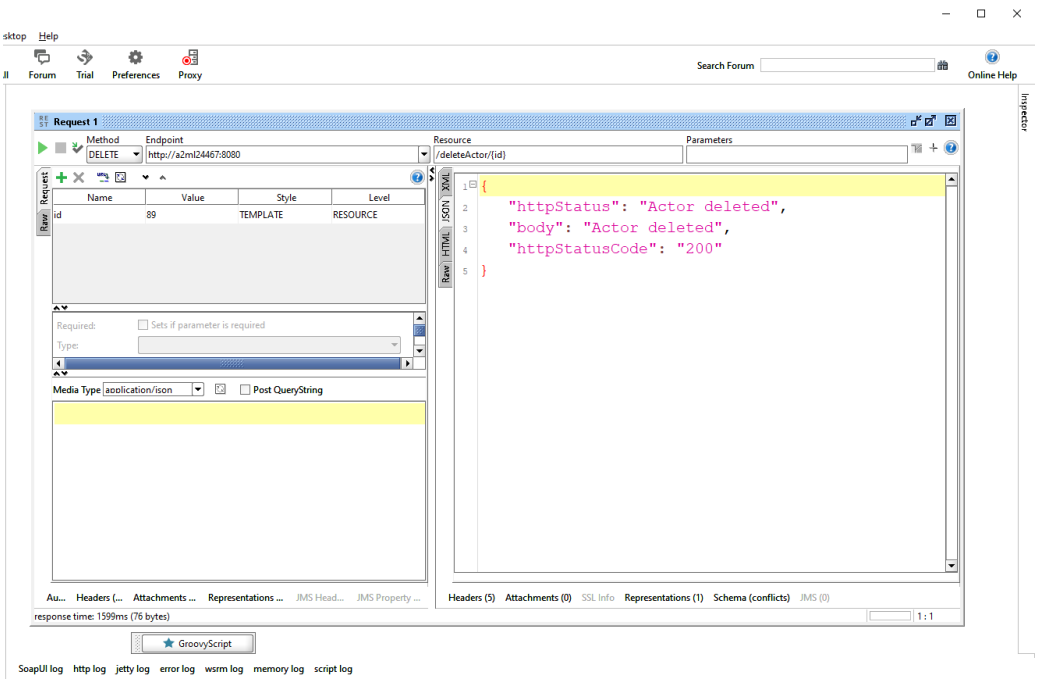
(In this case ID is auto generated. Whenever you are posting, one unique ID is generated for each and every case which is used to update and delete).

Please refer to the previous document where GET, PUT, POST and DELETE is explained in case of any confusion.

PUT method:-

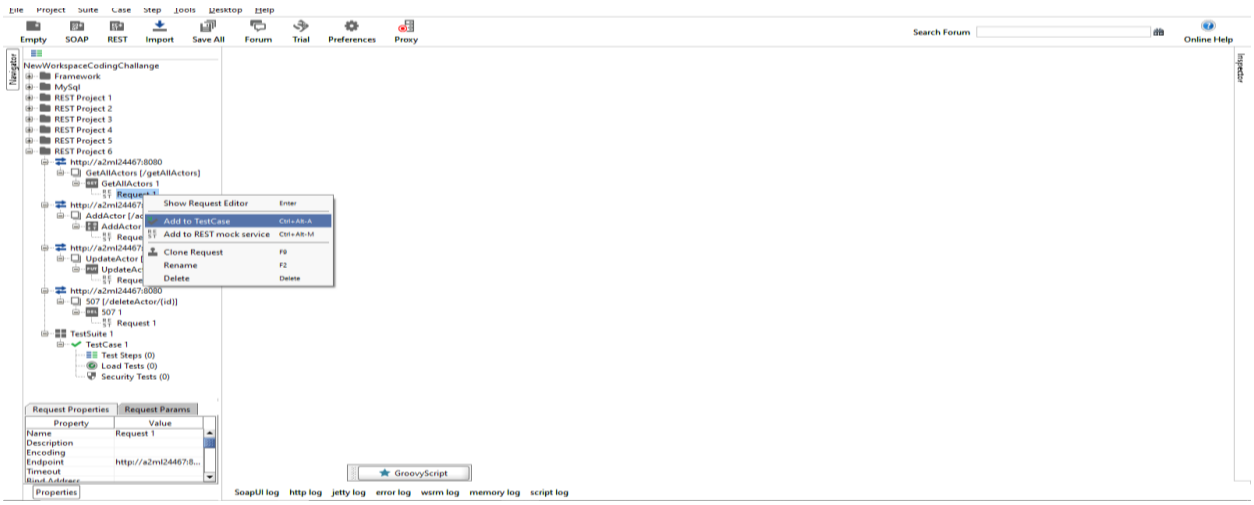


Delete Method:-

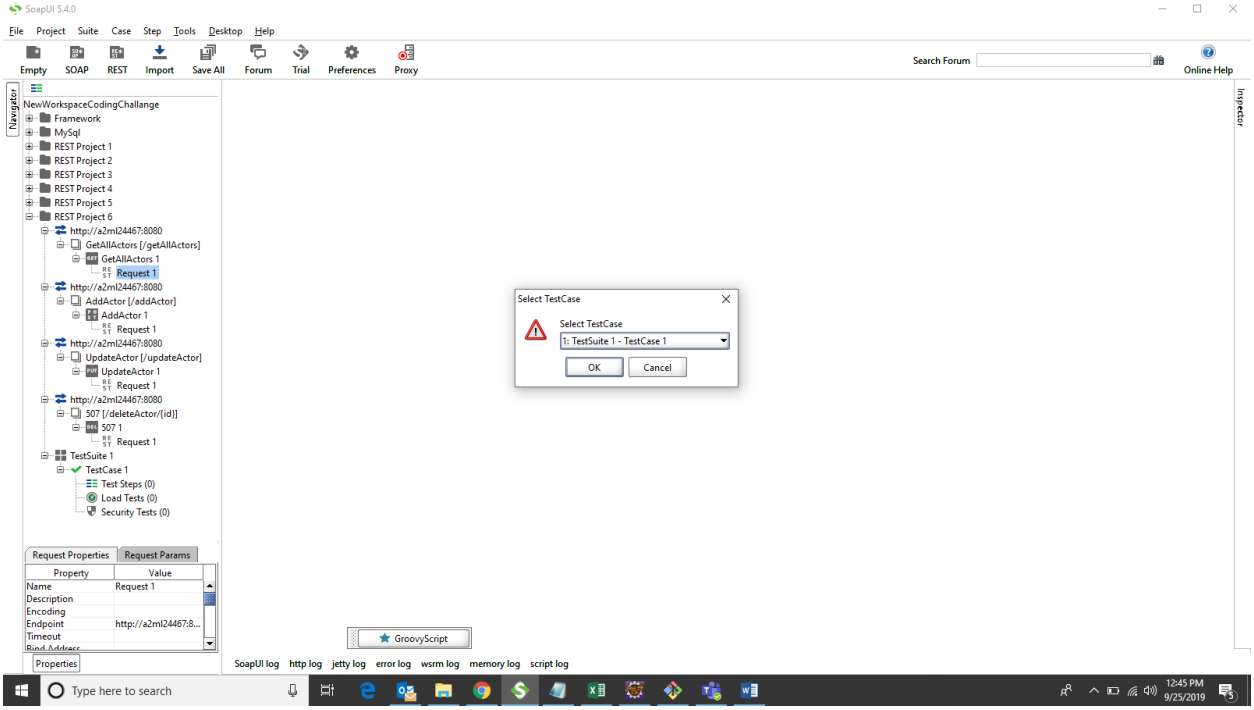


Step 11>Now right click on the project name->NEW TestSuite-> TestCase.(In the previous document it is mentioned how to create this).

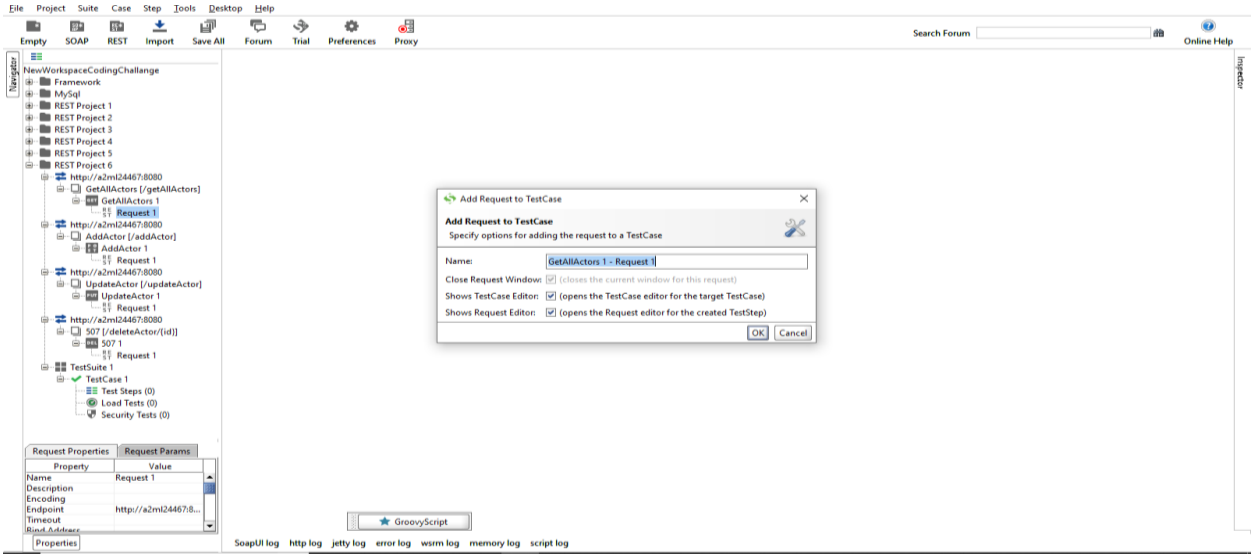
Step 12>Now right click on the Request1 of the GET method-> AddToTestCase.



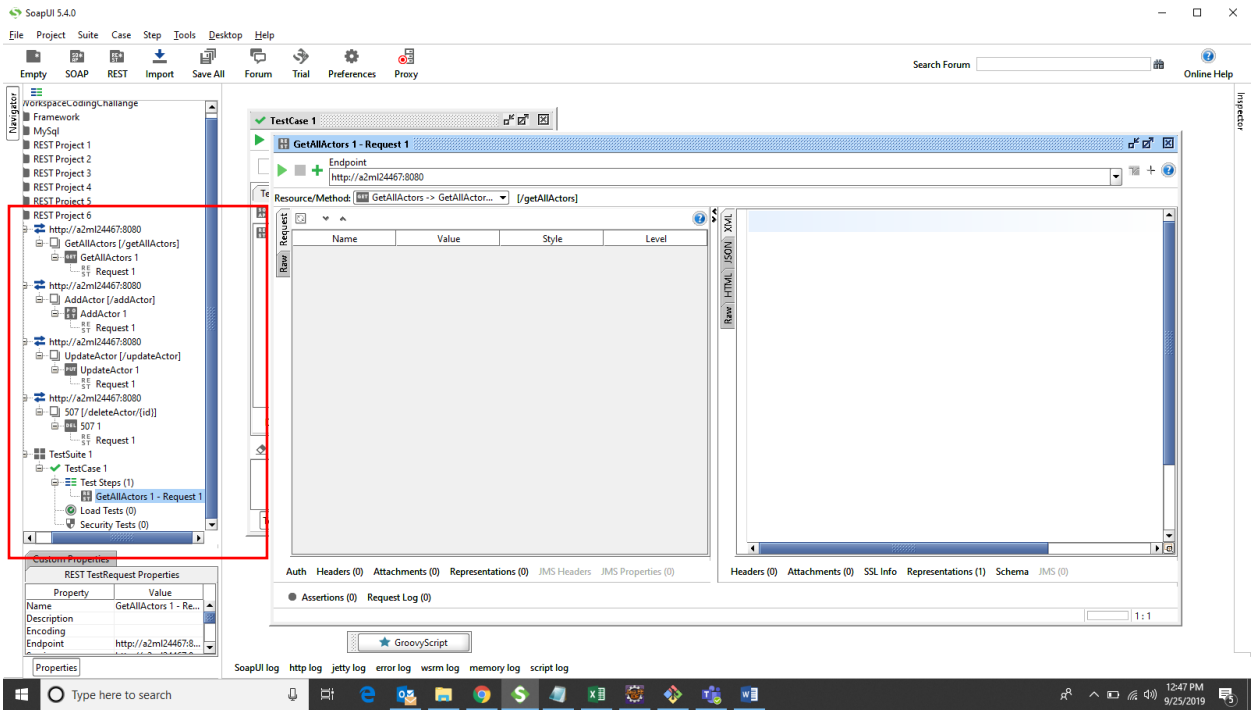
Step 13>Click OK.



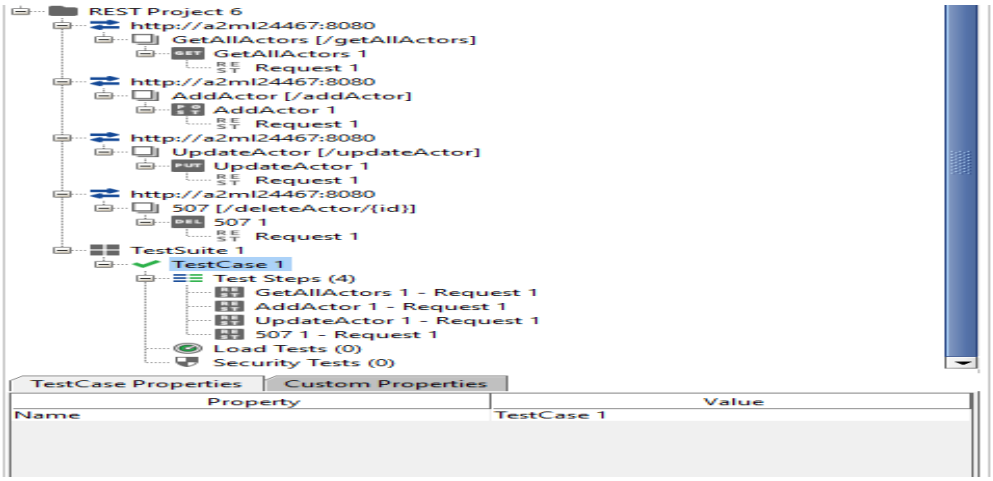
Step 14>You can change the name If you want and then click OK.



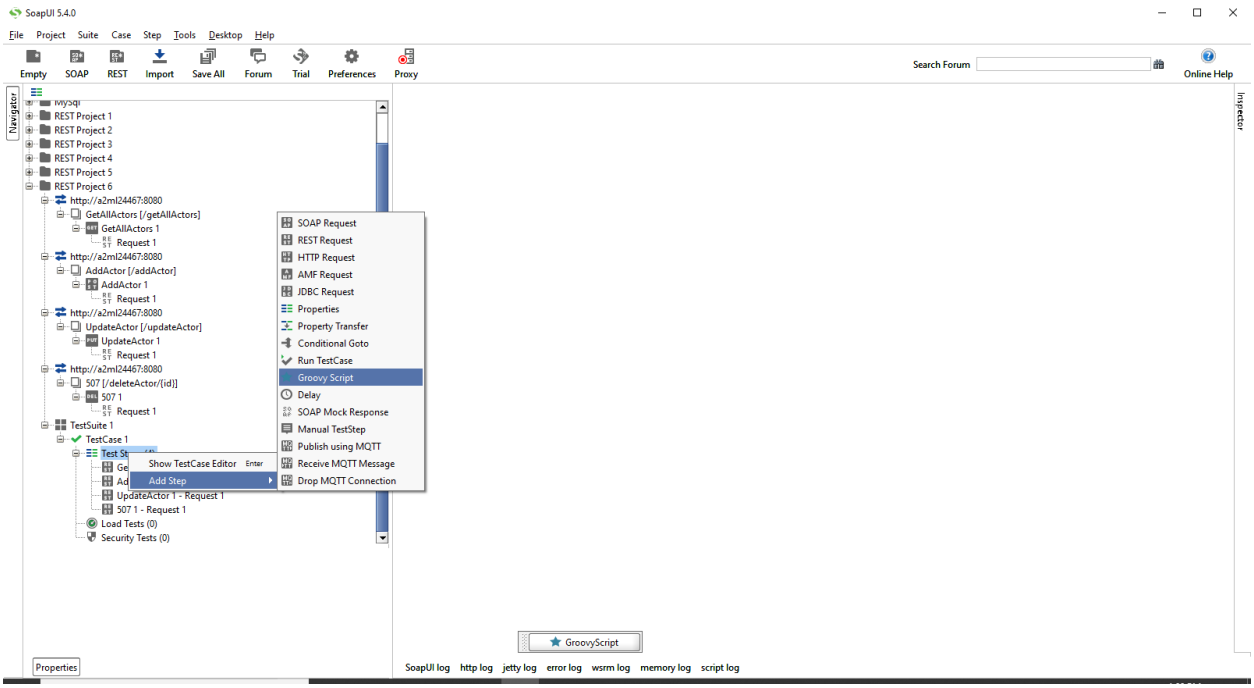
Step 15>Your folder structure will look something like that.



Step 16>Do the same thing for PUT, POST, DELETE. Click on the request of the methods ->AddToTestStep. Then your folder structure will look like that.

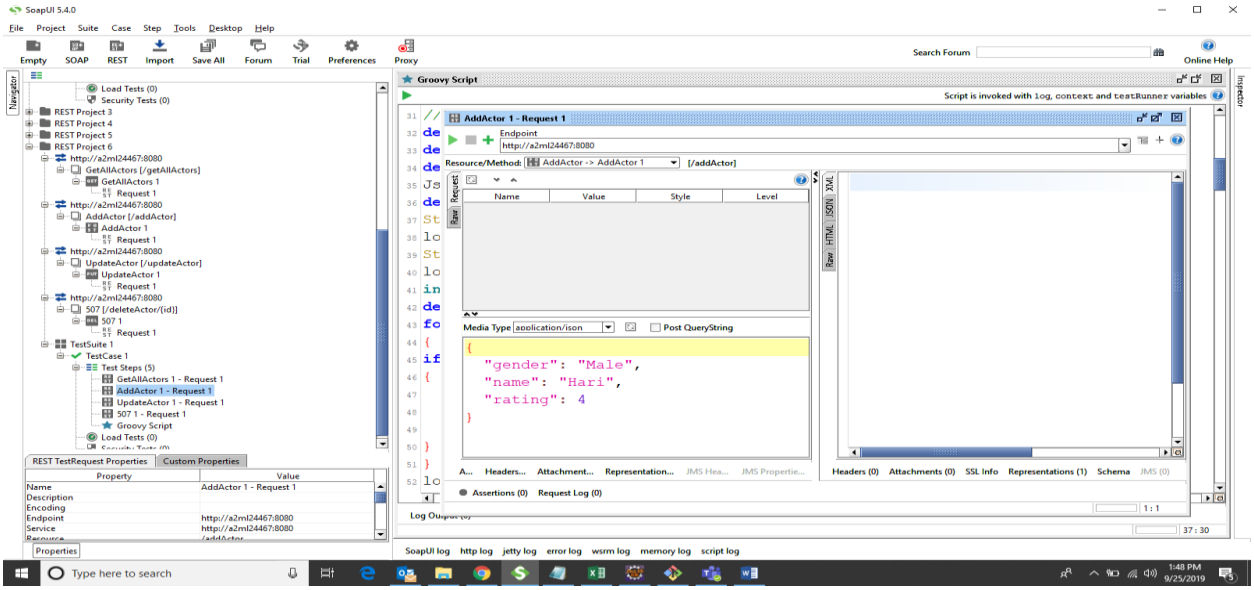


Step 17>Right click on testStep->AddStep->Groovy Script->ok.

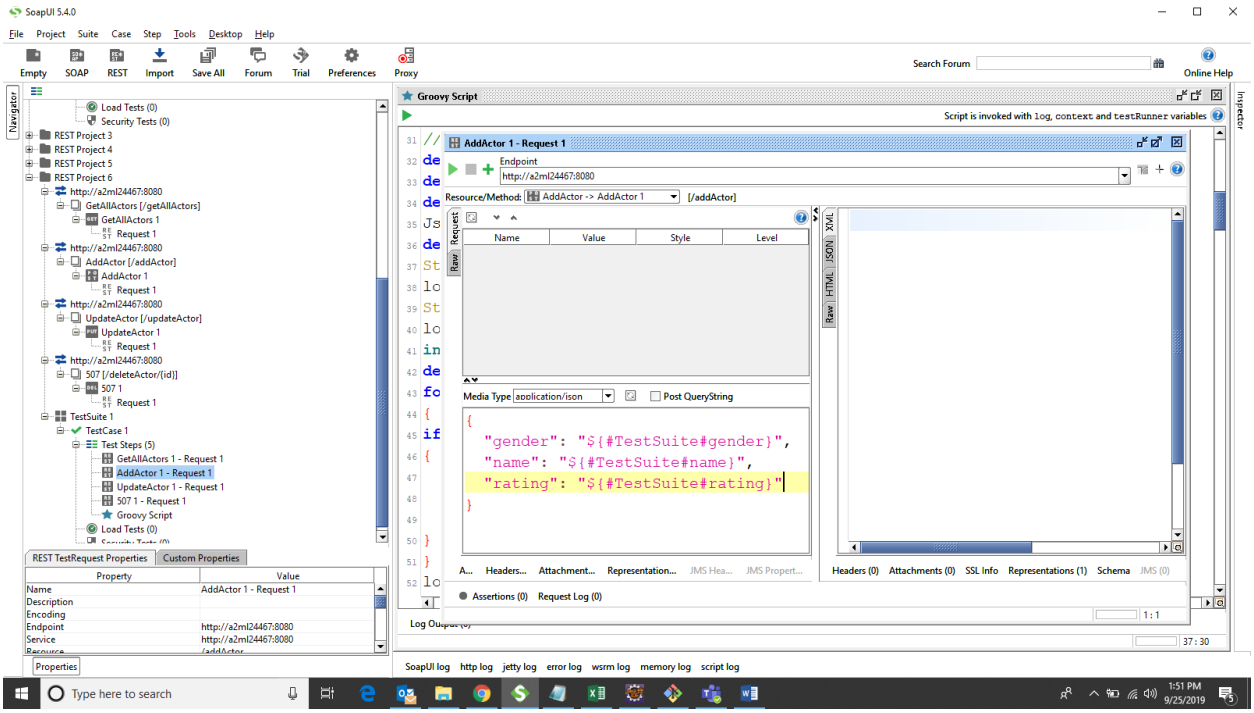


Step 18>Now We need to pass the element from the groovy script to the REST methods that are inside the TestSuite.

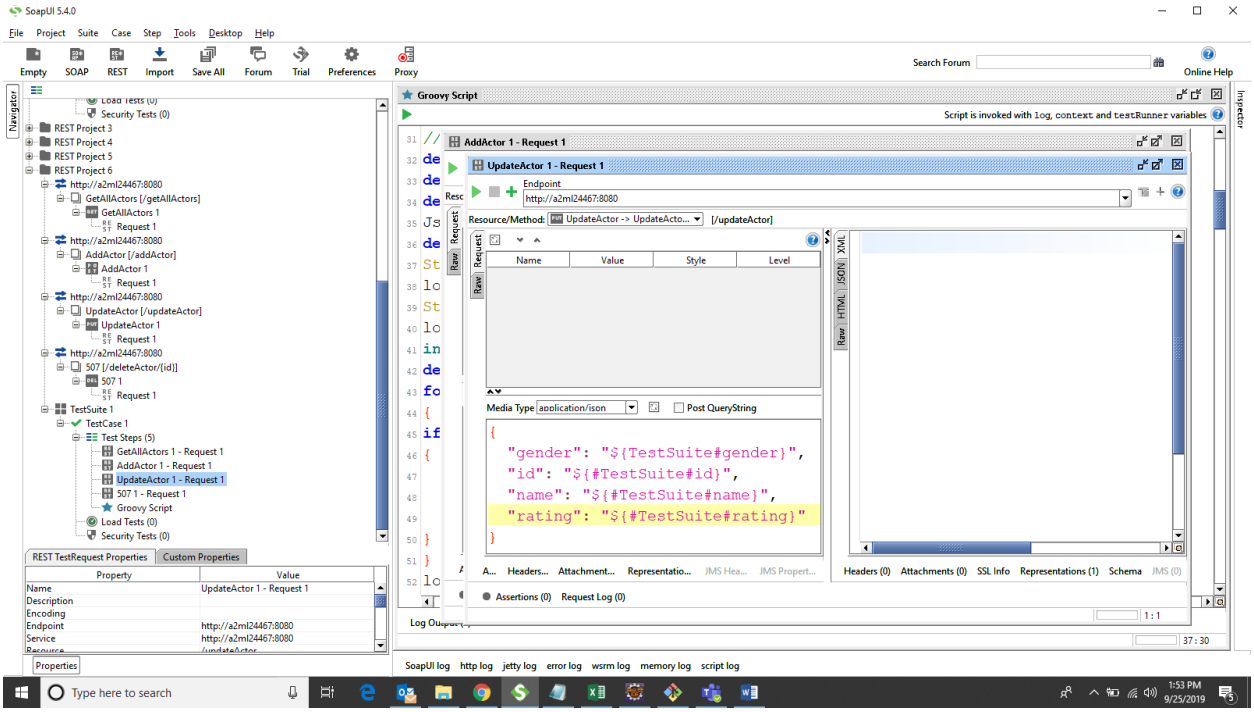
Click on the REST Request for the post methiod inside the TestSuite.



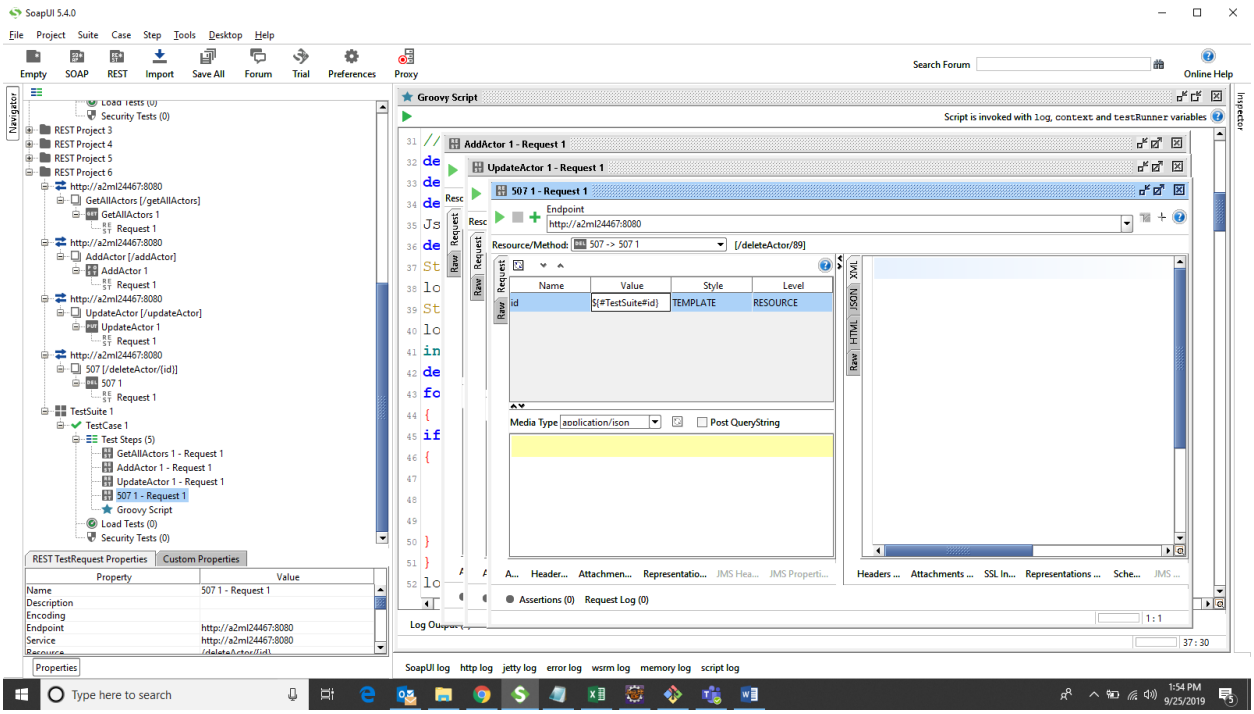
Step 19>Now you need to change it like shown in the below figure.



Step 20>Do the same for post.



Step 21>For Delete By id.



[Note:--

Since we are using here Get all therefore we don’t need to pass the parameter here. In case if we use Get By Id then we have to pass the parameter ‘id’ as we have done for the delete.]

Groovy script code

```
import groovy.json.JsonSlurper

//post


testRunner.testCase.testSuite.setPropertyValue("gender","male")

testRunner.testCase.testSuite.setPropertyValue("name","Hari")

testRunner.testCase.testSuite.setPropertyValue("rating","5")

def tCase = testRunner.testCase.testSuite.testCases["TestCase 1"]

def getIdTestStep = tCase.testSteps["AddActor 1 - Request 1"]

def as1=getIdTestStep.run(testRunner,context)

def responseJson =
getIdTestStep.testRequest.response.responseContent

log.info(responseJson)


//get

def getIdTestStep1 = tCase.testSteps["GetAllActors 1 - Request 1"]

def as6=getIdTestStep1.run(testRunner,context)
```

```
def responseJson1 =
getIdTestStep1.testRequest.response.responseContent

JsonSlurper slurper = new JsonSlurper()

def parsedJson = slurper.parseText(responseJson1)

String id1

def flag=0

for(def str:parsedJson.body)

{

if(str.gender.equals("male")  && str.name.equals("Hari"))

{

        id1=str.id

        flag=1

        //log.info("hi")

}

}

log.info(id1)


if(flag==1)

{

        log.info("posting")

}

else
```

```
{
	log.info("Not posted")
}

//put

testRunner.testCase.testSuite.setPropertyValue("id",id1)
testRunner.testCase.testSuite.setPropertyValue("gender","male")
testRunner.testCase.testSuite.setPropertyValue("name","Hari Chaini")
testRunner.testCase.testSuite.setPropertyValue("rating","5")


	def getIdTestStep2 = tCase.testSteps["UpdateActor 1 - Request 1"]

	def as2=getIdTestStep2.run(testRunner,context)


	def responseJson2 = getIdTestStep2.testRequest.response.responseContent

	log.info(responseJson2)
```

```
        def ass2=as2.getStatus().toString()

//get

def getIdTestStep3 = tCase.testSteps["GetAllActors 1 - Request 1"]

def as7=getIdTestStep3.run(testRunner,context)

def responseJson3 =
getIdTestStep3.testRequest.response.responseContent

JsonSlurper slurper1 = new JsonSlurper()

Map parsedJson1 = slurper.parseText(responseJson3)
log.info(parsedJson1)
String value10=parsedJson1.body.description
def flag1=0
for(def str1:parsedJson1.body)
{
if((str1.gender).equals("male")  && (str1.name).equals("Hari Chaini")
&& (str.id).equals(id1))
{
        flag1=1
```

```
}
else
{
        flag1=0
}
}
if(flag1==1)
{
        log.info("successfully updated")
}
else
{
log.info("Not Updated")
}

//delete

testRunner.testCase.testSuite.setPropertyValue("id",id1)

def getIdTestStep4 = tCase.testSteps["507 1 - Request 1"]

def as3=getIdTestStep4.run(testRunner,context)
```

```
def responseJson4 =
getIdTestStep4.testRequest.response.responseContent

log.info(responseJson4)

//get

def getIdTestStep5 = tCase.testSteps["GetAllActors 1 - Request 1"]

def as8=getIdTestStep5.run(testRunner,context)

def responseJson5 =
getIdTestStep5.testRequest.response.responseContent

JsonSlurper slurper3 = new JsonSlurper()

Map parsedJson4 = slurper.parseText(responseJson1)

log.info(parsedJson4)
def flag2=0
for(def str6:parsedJson4.body)
{
```

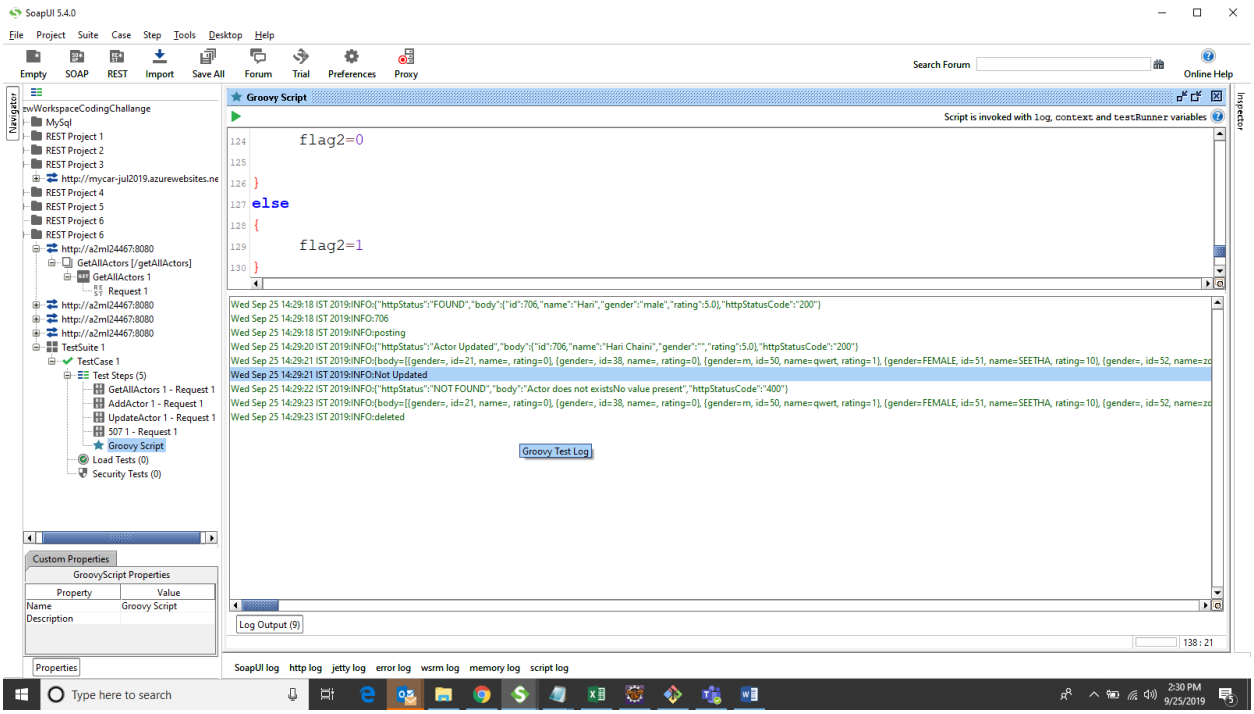
```
if((str6.name).equals("Hari Chaini"))
{
	flag2=0

}
else
{
	flag2=1
}
}
if(flag2==0)
{
	log.info("not deleted")
}
else
{
	log.info("deleted")
}
```

[Flow is->

Post -> get -> validate ->put -> get ->validate ->delete -> get -> validate]

Output

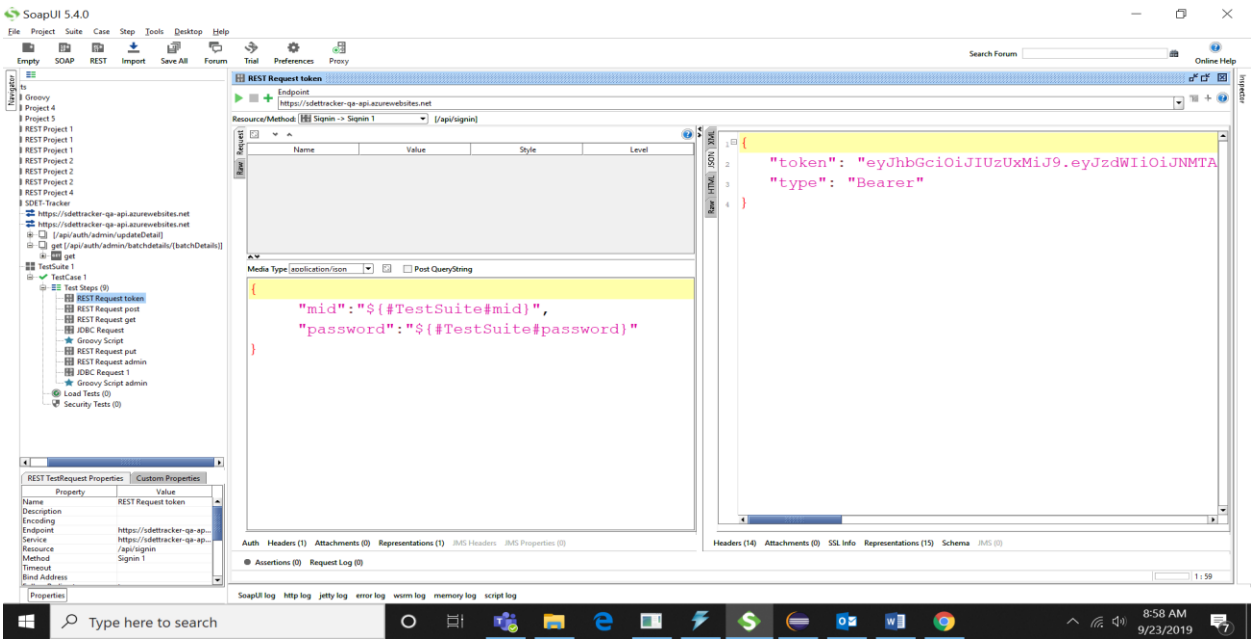


Passing JWT token as headers:

A very common use of a JWT token, and the one you should probably only use JWT for, is to use as an API authentication mechanism. Just to give you an idea, it's so popular and widely used that Google uses it to let you authenticate to their APIs.

So when authentication is required to access APIs we pass the authentication key as headers.

Using POST API given to generate token , pass authentication holder credentials as Json body and run the post request. Response body will contain the type and the token of authentication.



Groovy Script for POST operation

In the above attached picture, credentials are set as custom properties using groovy script.

Passing token to headers:

Performing any of the REST request which requires authentication key, will give Unauthorized error 401 if the token and type is not set in headers.

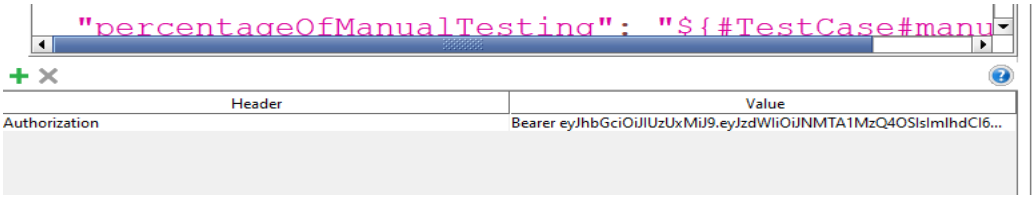
We can set headers in two ways

1. Manually.

2. Through groovy script.

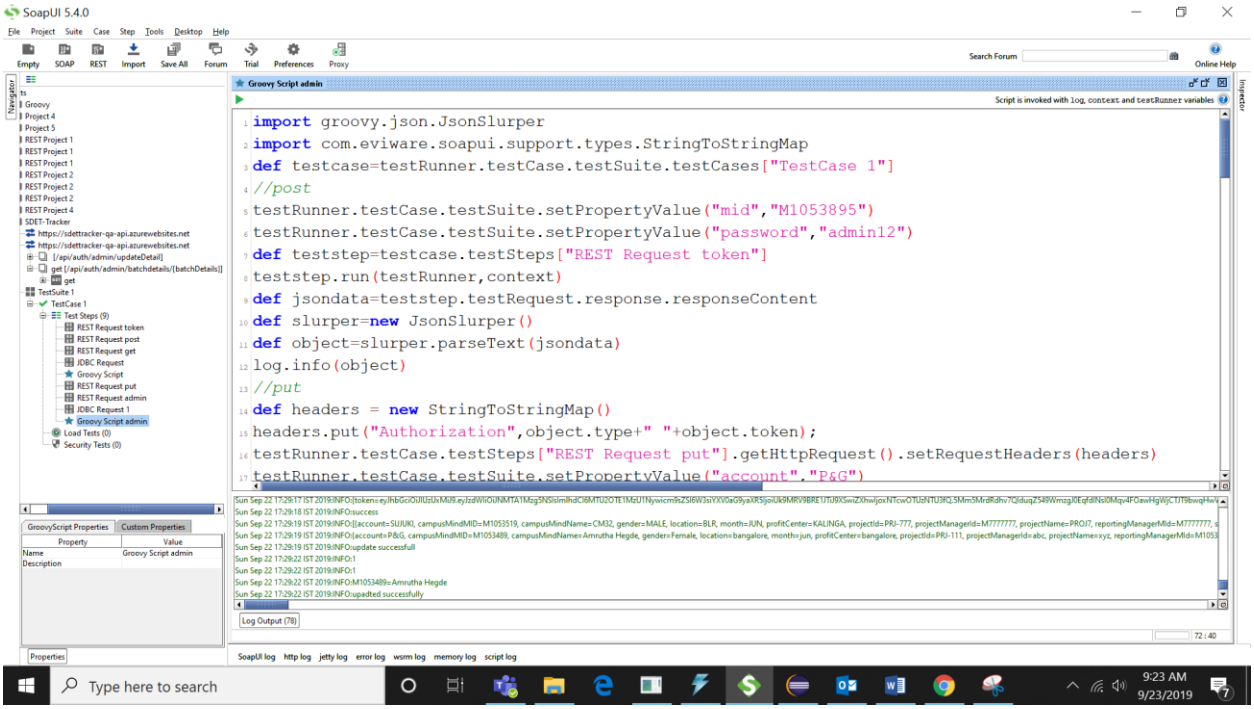
Manually:

1. Click on headers which is at the bottom of left hand side.
2. Click on + symbol and add header as authorization and value as "type of token+space+token"



Through groovy script:

We can also set headers from groovy script. Here the header and value acts as Map having key and value. i.e header acts as key and value as value.



Above attached snippet has the POST request to generate token , the response is stored in the variable(object).

```
def headers = new StringToStringMap()
headers.put("Authorization",object.type+" "+object.token);
testRunner.testCase.testSteps["REST Request put"].getHttpRequest().setRequestHeaders(headers)
```

Use the above code to set headers . Here object contains response of POST Request of token generation API.

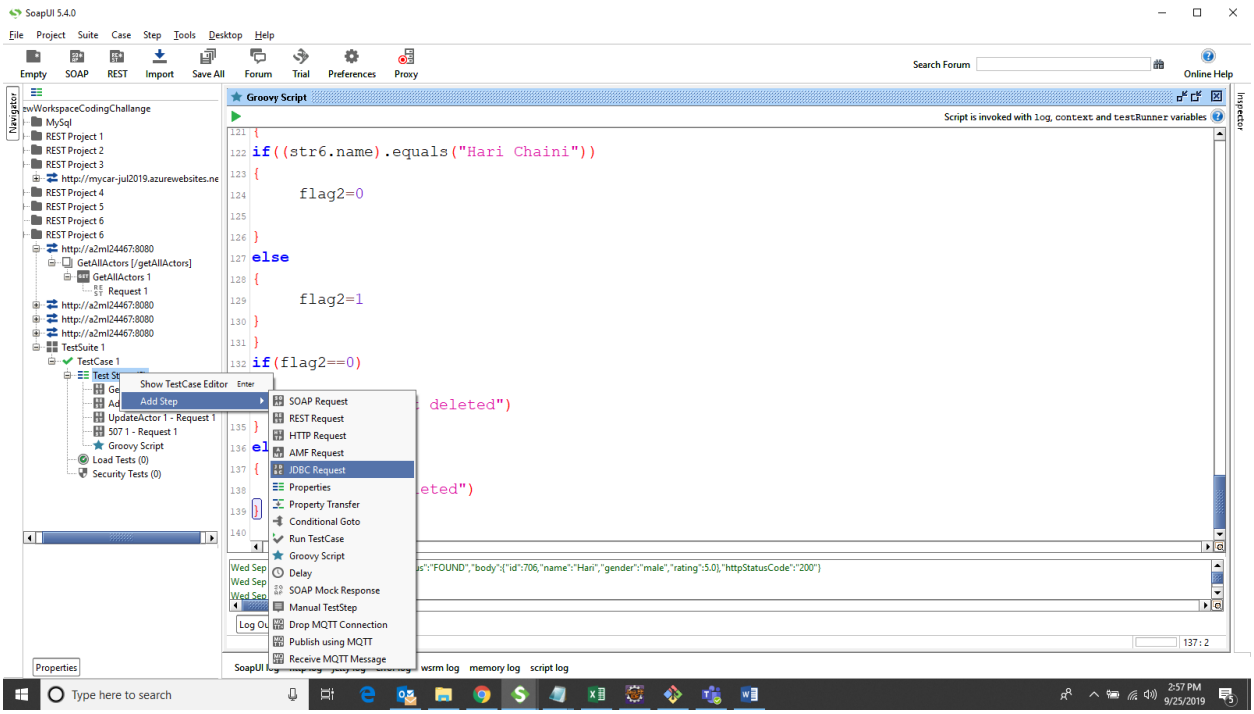
Reminder: JWT token remains valid only for pre-set time period. If JWT token expires, we have to re run the POST Request of token generation API and replace the old token with new token wherever used.

Setting headers through groovy is beneficials as it generates new token n sets it as headers everytime we run the groovy.

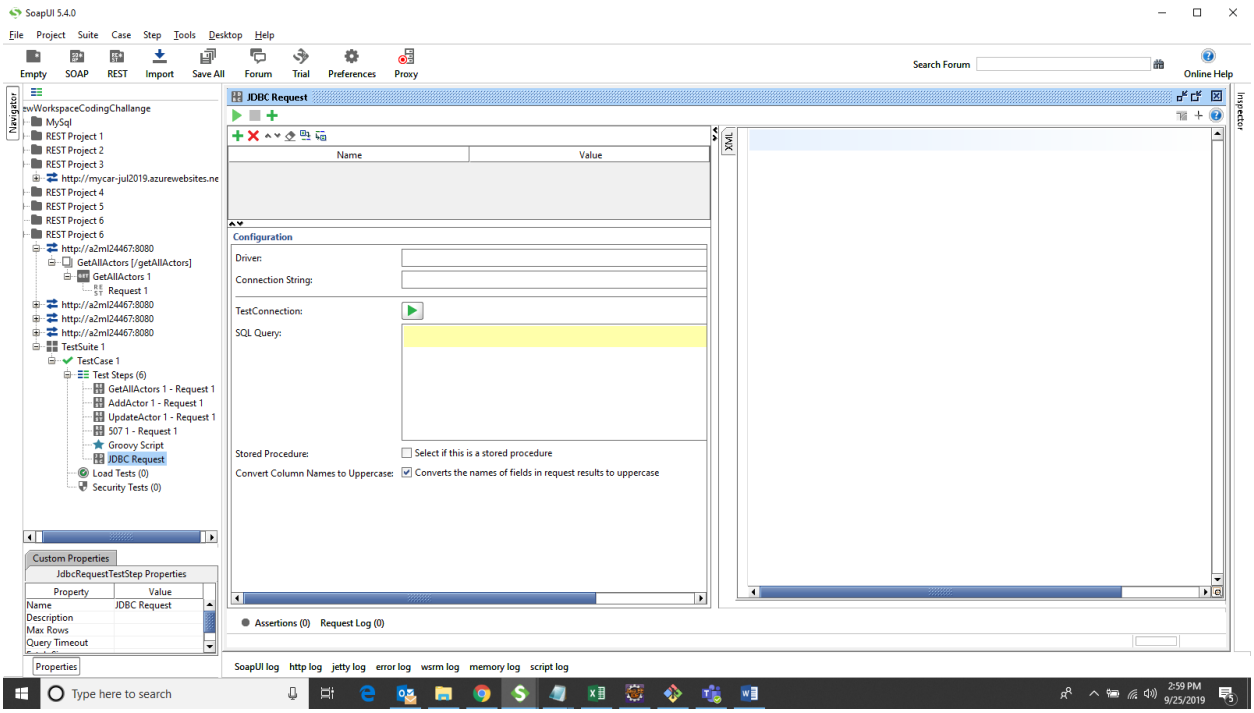
JDBC DataBase Validation

In the above problem only click on TestSteps->AddSteps-> JDBC Request.

Then click OK.



Step 2>You will get something like that.

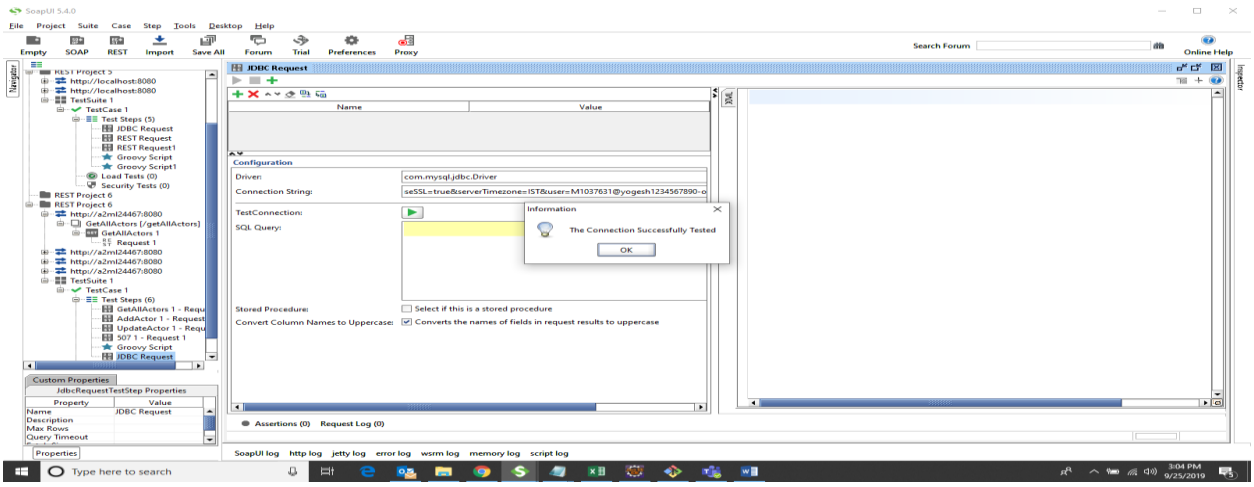


Step 3>In the Driver filed write- com.mysql.jdbc.Driver

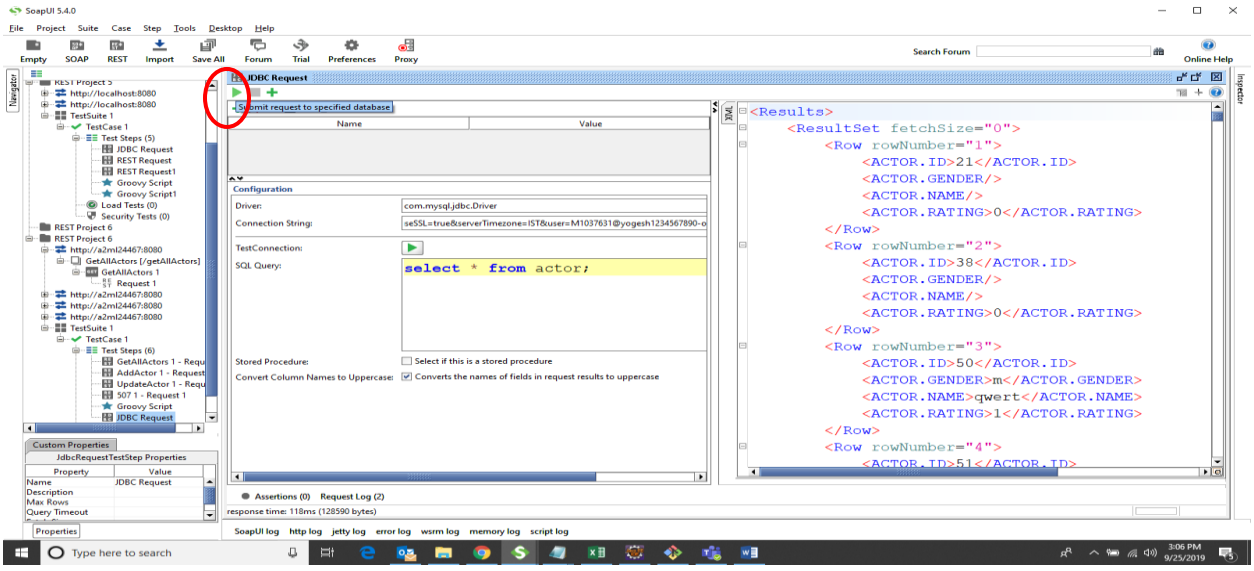
In the Connection String Write - jdbc:mysql://localhost:3306/databasename?user=root&password=Welcome123(This will only be valid for the spring boot problem in your local system. Otherwise, Leads will provide the credentials to fill in the Connection String).

Step 4>Click on the run button below ‘Connection String’.

You will get



Step 5>Now you can write any sql query for the tables present in the database. Click on run present at the top.



Step 6>You will get the XML response. You will get all the attributes as nodes and the values present in that table.

Now,Just Include some of the lines for JDBC validation in the existing groovy script.

Jdbc code

//jdbc

```
def flag=0

def teststep2=testcase.testSteps["JDBC Request"]

teststep2.run(testRunner,context)

def JDBC=teststep2.testRequest.response.responseContent

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

def xml=groovyUtils.getXmlHolder(JDBC)

for(obj in xml.getNodeValues("//ACTOR.ID"))
{
	if(obj.equals(""+40))
	{
		flag=1
		break;
	}
	else
	{
		flag=0
```

```
        }
}
if(flag==1)
{
        log.info("present")
}
else
{
        log.info("not present")
}
```

Step 7>After including this ,Run the groovy script.