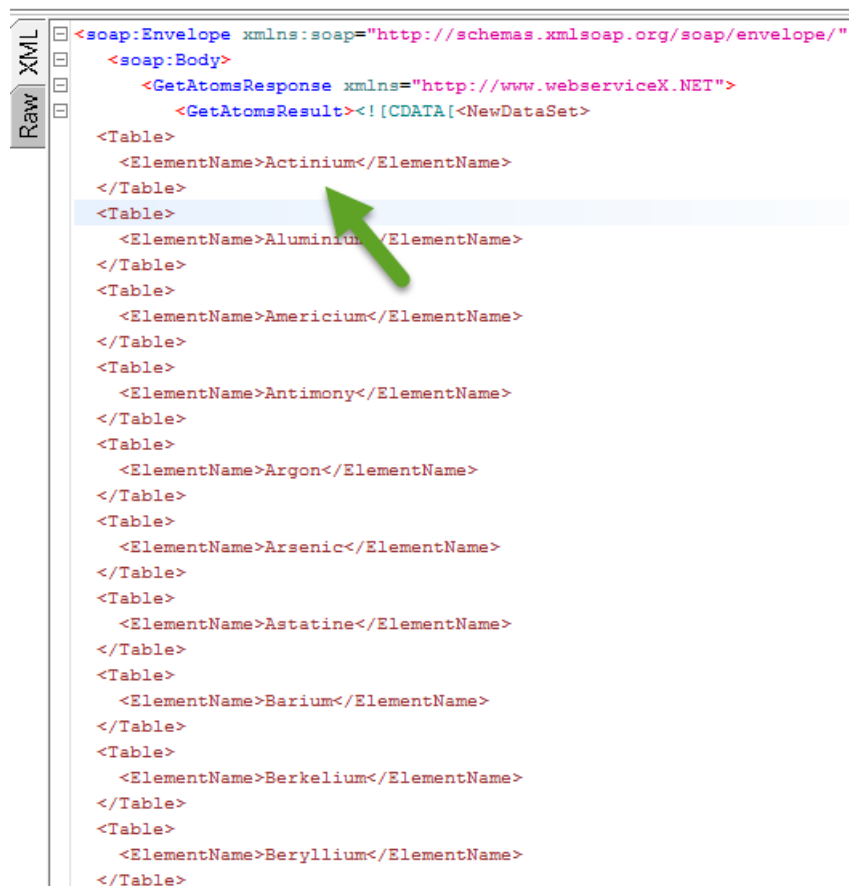


SOPAUI Assignments

1) SOAP

- 1) Open Soapui Tool
- 2) File -> Import Project (periodictable-soapui-project.xml)
- 3) Create "TestSuite" as "PeriodicTable_TestSuite"
- 4) Create "TestCase" as "PeriodicTable_TestCase"
- 5) Create Soap Request as "GetAtoms"
 - a. Select "periodictableSoap -> GetAtoms" method
 - b. click checkbox as "Create optional elements" then click "ok"
 - c. Run the "Test Step" and Verify the response



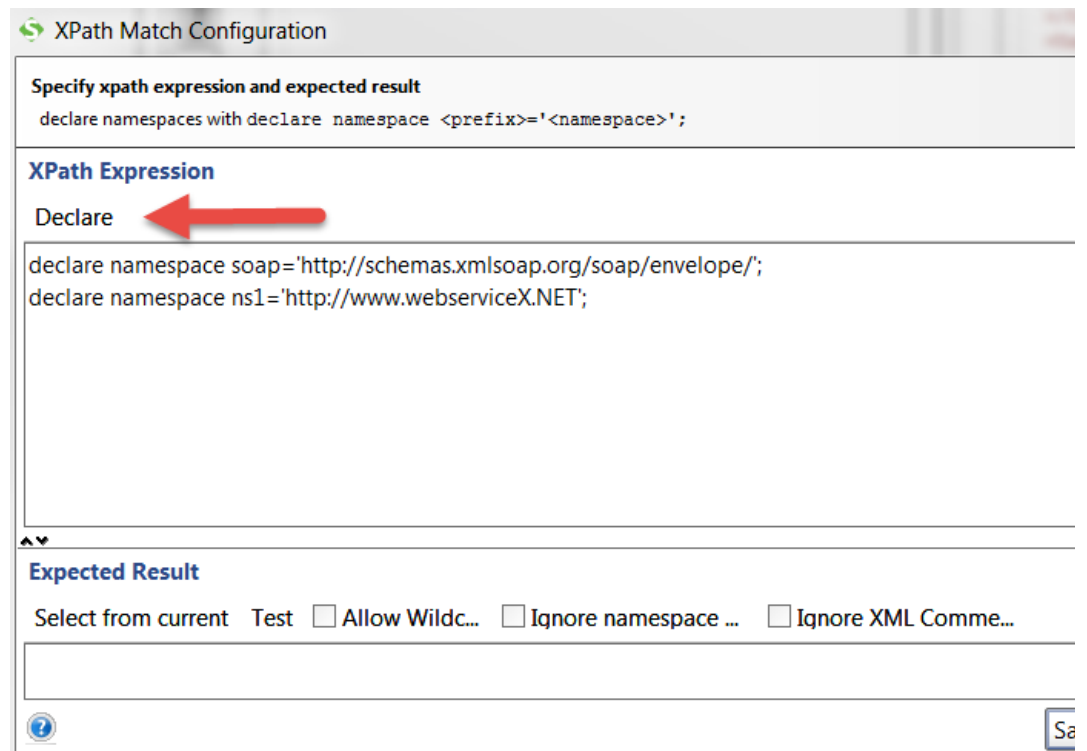
The screenshot displays the Raw XML view of a SOAP response in SOAPUI. The XML structure is as follows:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetAtomsResponse xmlns="http://www.webserviceX.NET">
      <GetAtomsResult><![CDATA[<NewDataSet>
        <Table>
          <ElementName>Actinium</ElementName>
        </Table>
        <Table>
          <ElementName>Aluminium</ElementName>
        </Table>
        <Table>
          <ElementName>Americium</ElementName>
        </Table>
        <Table>
          <ElementName>Antimony</ElementName>
        </Table>
        <Table>
          <ElementName>Argon</ElementName>
        </Table>
        <Table>
          <ElementName>Arsenic</ElementName>
        </Table>
        <Table>
          <ElementName>Astatine</ElementName>
        </Table>
        <Table>
          <ElementName>Barium</ElementName>
        </Table>
        <Table>
          <ElementName>Berkelium</ElementName>
        </Table>
        <Table>
          <ElementName>Beryllium</ElementName>
        </Table>
      </NewDataSet>]]>
    </GetAtomsResult>
  </GetAtomsResponse>
</soap:Body>
</soap:Envelope>
```

A green arrow points to the second `<Table>` element, which contains the `<ElementName>Aluminium</ElementName>` tag.

6) Set Assertion

- a. Valid http status code as “200”
- b. Contains “Actinium”
- c. Not contains “Sweden”
- d. XPath Match
 - a. Click Declare



The image shows a screenshot of the 'XPath Match Configuration' dialog box. The dialog has a title bar with a green icon and the text 'XPath Match Configuration'. Below the title bar, there is a section titled 'Specify xpath expression and expected result' with a text area containing the text 'declare namespaces with declare namespace <prefix>=<namespace>;'. Below this, there is a section titled 'XPath Expression' with a sub-section 'Declare' highlighted by a red arrow. The text area below 'Declare' contains the XPath expression: 'declare namespace soap='http://schemas.xmlsoap.org/soap/envelope/'; declare namespace ns1='http://www.webserviceX.NET;'. Below the 'XPath Expression' section, there is a section titled 'Expected Result' with a sub-section 'Select from current Test' and three checkboxes: 'Allow Wildc...', 'Ignore namespace ...', and 'Ignore XML Comme...'. The 'Allow Wildc...' checkbox is checked. At the bottom left, there is a question mark icon, and at the bottom right, there is a 'Sa' button.

XPath Match Configuration

Specify xpath expression and expected result
declare namespaces with declare namespace <prefix>=<namespace>;

XPath Expression

Declare

declare namespace soap='http://schemas.xmlsoap.org/soap/envelope/';
declare namespace ns1='http://www.webserviceX.NET';

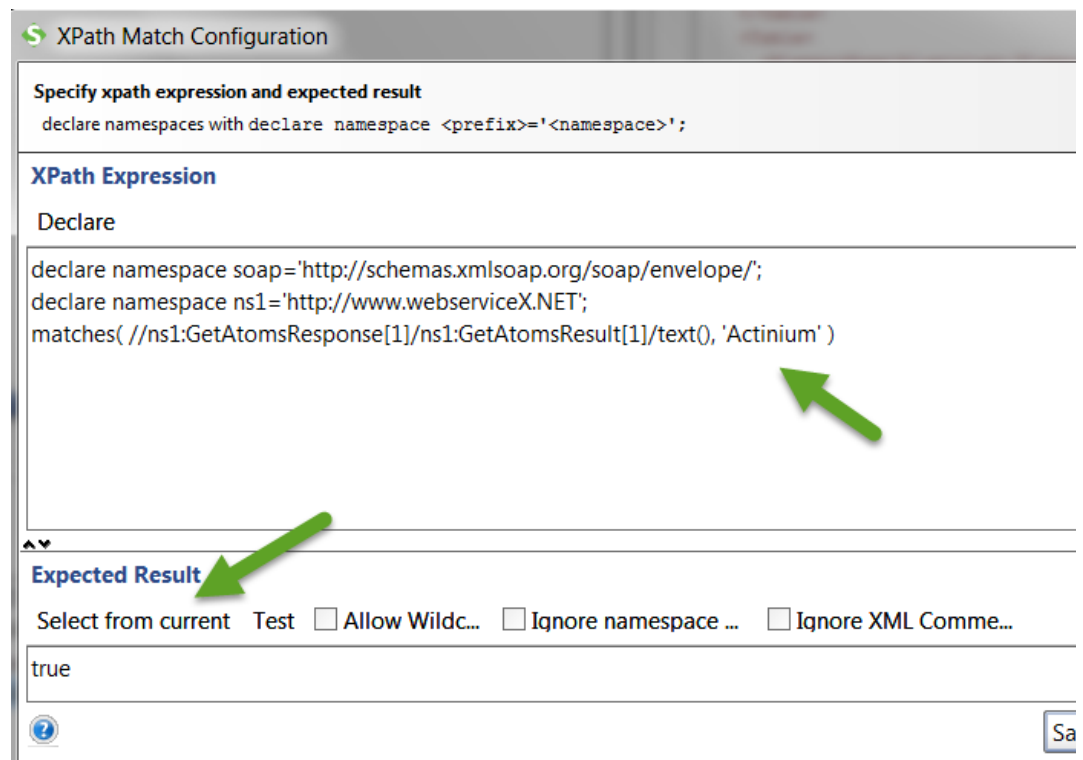
Expected Result

Select from current Test ☒ Allow Wildc... ☐ Ignore namespace ... ☐ Ignore XML Comme...

?

Sa

b. Write the Expression



The dialog box is titled "XPath Match Configuration". It has a section "Specify xpath expression and expected result" with a text area containing the XPath expression: `declare namespaces with declare namespace <prefix>='<namespace>';`. Below this is a section "XPath Expression" with a text area containing the following code: `declare namespace soap='http://schemas.xmlsoap.org/soap/envelope/';
declare namespace ns1='http://www.webserviceX.NET';
matches(//ns1:GetAtomsResponse[1]/ns1:GetAtomsResult[1]/text(), 'Actinium')`. A green arrow points to the closing parenthesis of the matches function. Below the XPath Expression section is a section "Expected Result" with a text area containing the value `true`. A green arrow points to the "Expected Result" section header. Above the text area in the "Expected Result" section are four checkboxes: "Select from current" (checked), "Test", "Allow Wildc...", "Ignore namespace ...", and "Ignore XML Comme...".

Specify xpath expression and expected result

declare namespaces with declare namespace <prefix>='<namespace>';

XPath Expression

Declare

```
declare namespace soap='http://schemas.xmlsoap.org/soap/envelope/';  
declare namespace ns1='http://www.webserviceX.NET';  
matches( //ns1:GetAtomsResponse[1]/ns1:GetAtomsResult[1]/text(), 'Actinium' )
```

Expected Result

Select from current ☒ Test ☐ Allow Wildc... ☐ Ignore namespace ... ☐ Ignore XML Comme...

true

c. Click on “Select from current”

7) Create Soap Request as “GetAtomicNumber”

- Select “periodictableSoap -> GetAtomicNumber” method
- Set “Actinium” in Request



The image shows a web browser window with the address bar displaying `http://www.webserviceX.net/periodictable.asmx`. Below the address bar is a "Raw XML" view of a SOAP request. The XML code is as follows: `<?xml version='1.0' encoding='utf-8'>
<soapenv:Envelope xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/'>
 <soapenv:Header/>
 <soapenv:Body>
 <web:GetAtomicNumber>
 <!--Optional:-->
 <web:ElementName>Actinium</web:ElementName>
 </web:GetAtomicNumber>
 </soapenv:Body>
</soapenv:Envelope>`. A green arrow points to the `<web:ElementName>Actinium</web:ElementName>` line.

c. Run the “Test Step” and Verify the response



```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetAtomicNumberResponse xmlns="http://www.webserviceX.NET">
      <GetAtomicNumberResult><![CDATA[<NewDataSet>
        <Table>
          <AtomicNumber>89</AtomicNumber>
          <ElementName>Actinium</ElementName>
          <Symbol>Ac</Symbol>
          <AtomicWeight>227</AtomicWeight>
          <BoilingPoint>3470</BoilingPoint>
          <IonisationPotential>6.94</IonisationPotential>
          <EletroNegativity>1</EletroNegativity>
          <AtomicRadius>1.8800000000000001</AtomicRadius>
          <MeltingPoint>1323</MeltingPoint>
          <Density>10070</Density>
        </Table>
      </NewDataSet>]]</GetAtomicNumberResult>
    </GetAtomicNumberResponse>
  </soap:Body>
</soap:Envelope>

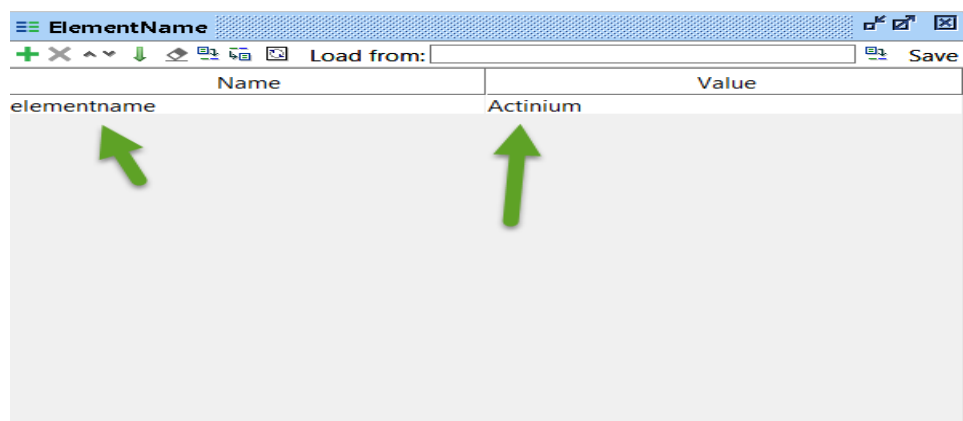
```

8) Set Assertion

- Valid http status code
- Contains "Ac"
- Not contains "AF"

9) Properties

- Right click Insert Step -> Properties
- Set Properties as "ElementName"
- Set name as "elementname" and value as "Actinium"



Name	Value
elementname	Actinium

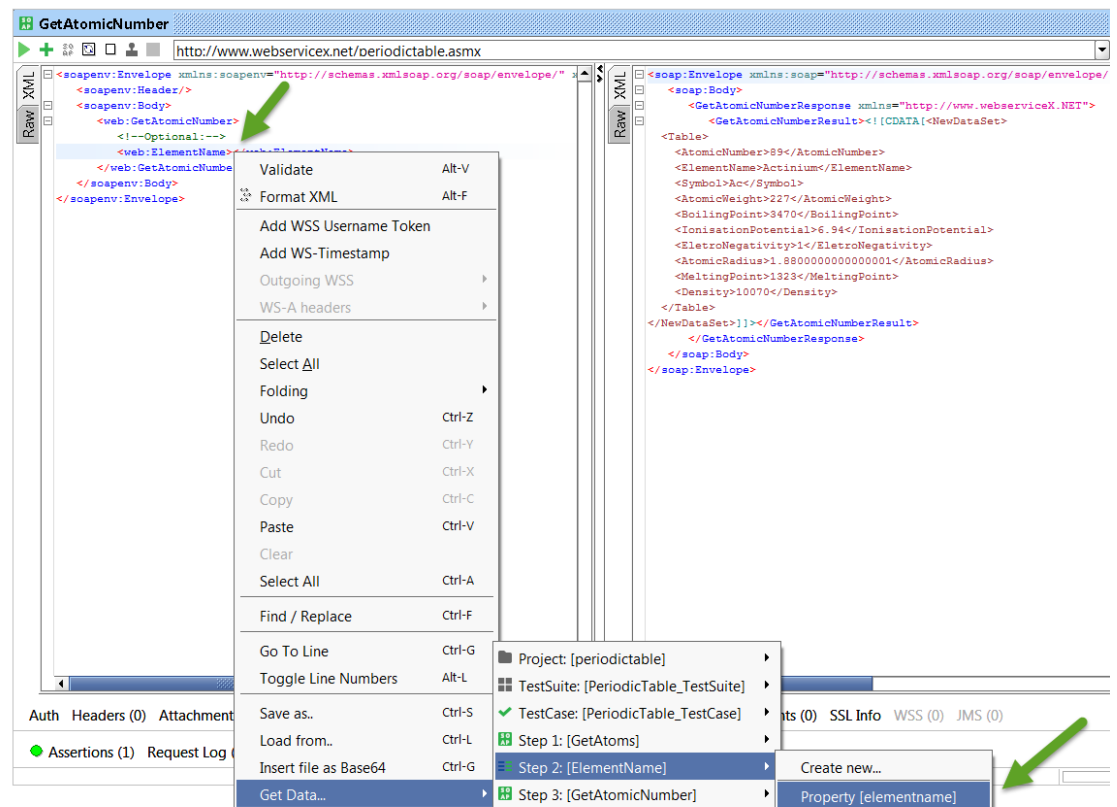
10) Set Properties in “GetAtomicNumber” Request

a. Remove “Actinium” from Request



b. Set Property value in Request

c. Right Click Get Data->ElementName-> Property[elementname]





d. Verify the response



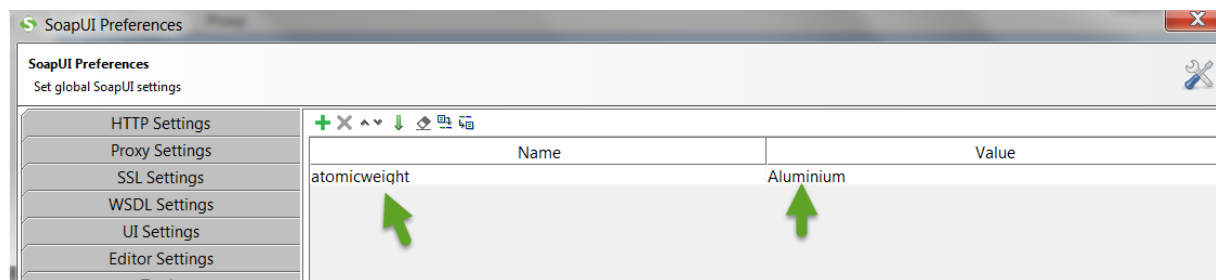
10) Set Global Properties

- Right click Insert Step -> Groovy Script
- Set name as "Weight"
- Write groovy script

★ **Weight**

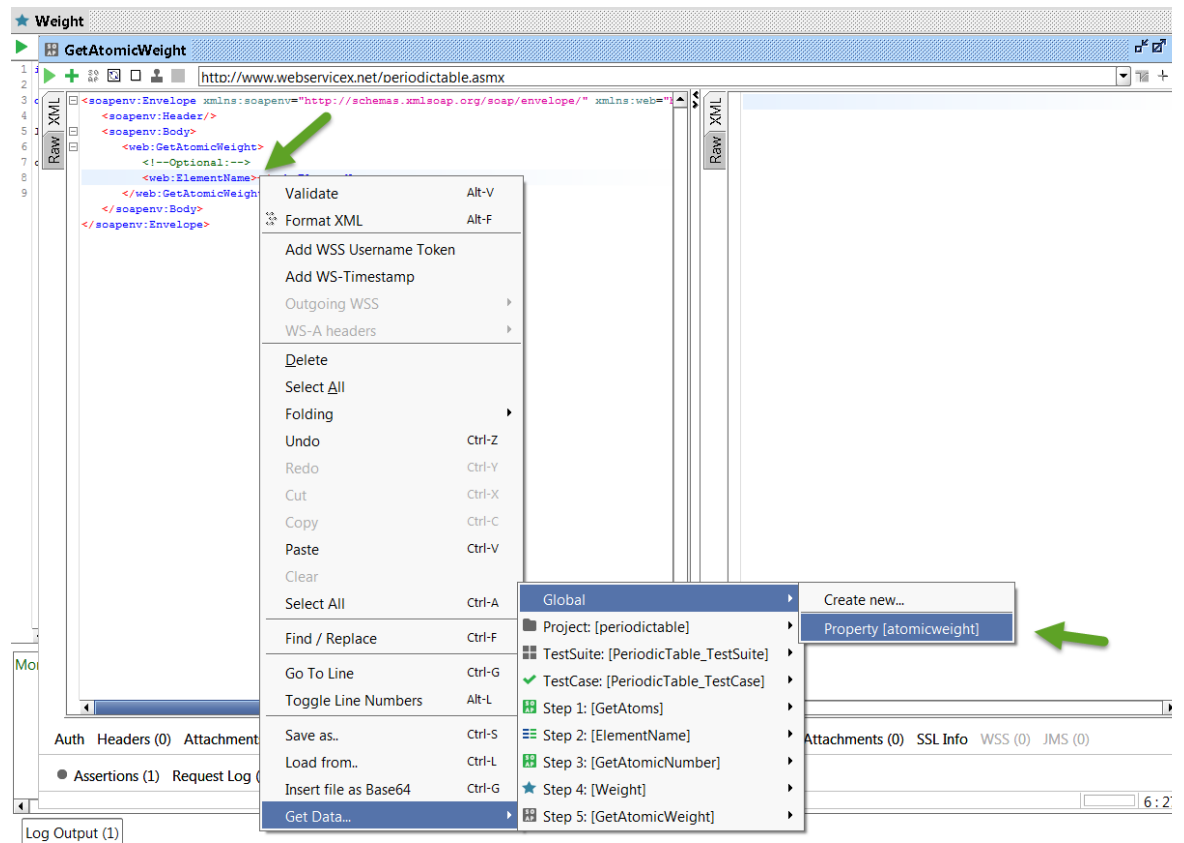
```
1 import groovy.xml.*
2
3 def atomicweight = "Aluminium"
4
5 log.info(atomicweight)
6
7 com.eviware.soapui.SoapUI.globalProperties.setPropertyValue( "atomicweight", atomicweight)
8
9
```

- d. Run the groovy script
- e. Verify in “Global Property” File-> Preferences -> Global Properties



11) Create Soap Request as “GetAtomicWeight”

- a. Select “periodictableSoap -> GetAtomicWeight” method
- b. Set Property value in Request
- c. Right Click Get Data->Global-> Property[atomicweight]



d. Run the “Test Step” and Verify the response




12) Set Assertion

- Valid http status code
- Contains “26.9815”
- Not contains “29.00”

13) Fetch value from “GetAtoms” Response

- Right click Insert Step -> Groovy Script
- Set name as “FetchResponse_Atoms”
- Fetch ElementName as “Antimony”

```
<?xml version='1.0' encoding='UTF-8'>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetAtomsResponse xmlns="http://www.webserviceX.NET">
      <GetAtomsResult><![CDATA[<NewDataSet>
        <Table>
          <ElementName>Actinium</ElementName>
        </Table>
        <Table>
          <ElementName>Aluminium</ElementName>
        </Table>
        <Table>
          <ElementName>Americium</ElementName>
        </Table>
        <Table>
          <ElementName>Antimony</ElementName>
        </Table>
      </NewDataSet>]]></GetAtomsResult>
    </GetAtomsResponse>
  </soap:Body>
</soap:Envelope>
```

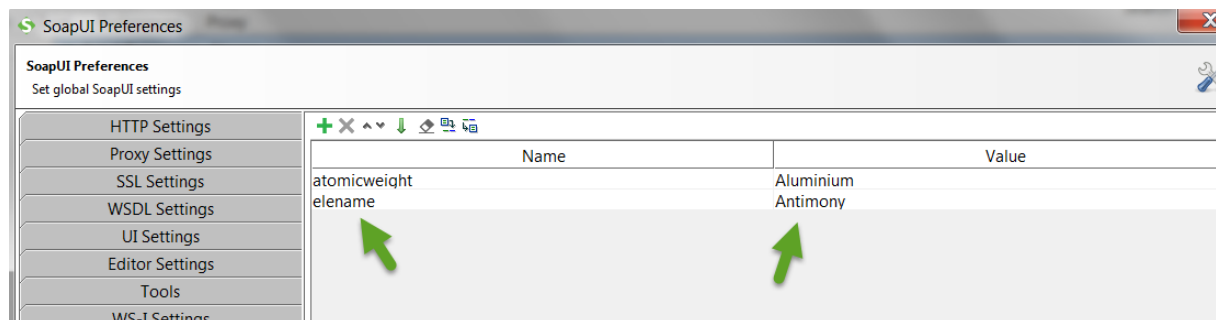


- Write groovy script

★ FetchResponse_Atoms

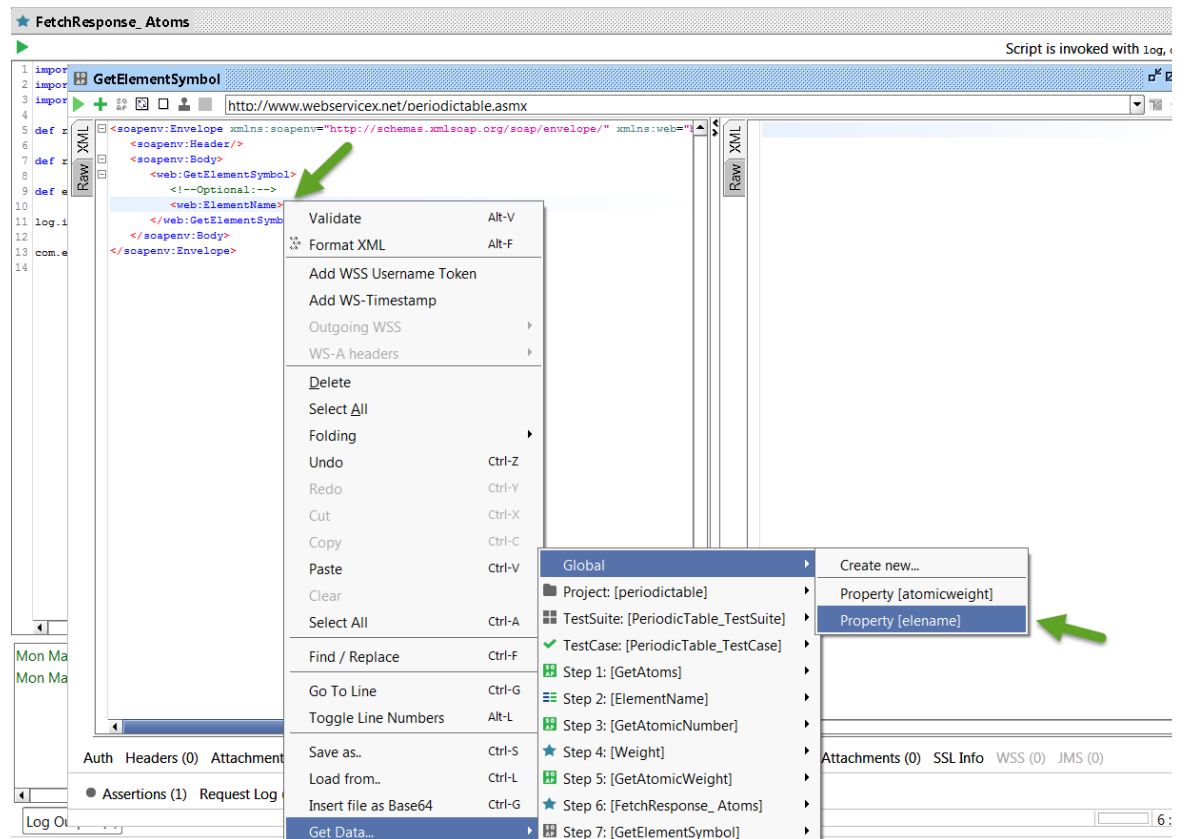
```
1 import groovy.xml.*
2 import groovy.util.XmlParser
3 import groovy.util.XmlSlurper
4
5 def responseContent = testRunner.testCase.getTestStepByName("GetAtoms").getPropertyValue("Response")
6
7 def response = new XmlSlurper().parseText(responseContent).Body.GetAtomsResponse.GetAtomsResult.text()
8
9 def elename = new XmlSlurper().parseText(response).Table.ElementName[3]
10
11 log.info(elename)
12
13 com.eviware.soapui.SoapUI.globalProperties.setPropertyValue("elename", elename.toString())
14
```

- Run the groovy script
- Verify in “Global Property” File-> Preferences -> Global Properties

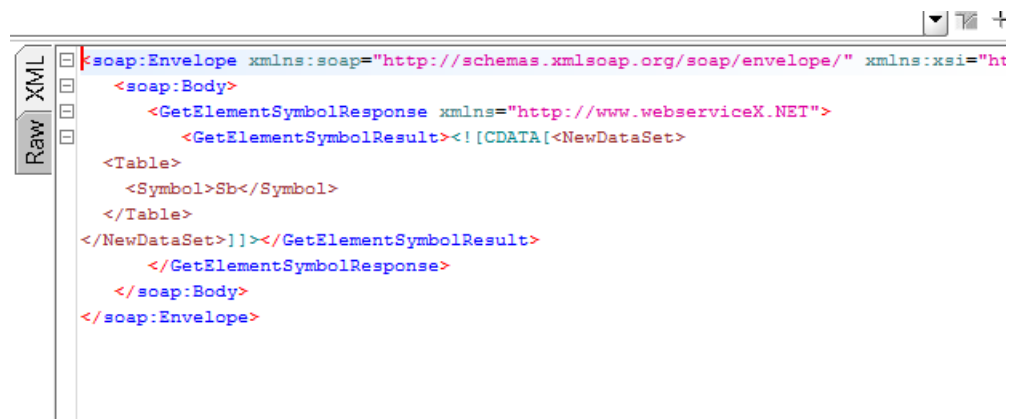


14) Create Soap Request as “GetElementSymbol”

- Select “periodictableSoap -> GetElementSymbol” method
- Set Property value in Request
- Right Click Get Data->Global-> Property[elename]



- Run the “Test Step” and Verify the response



15) Set Assertion


- a. Valid http status code
- b. Contains "Sb"
- c. Not contains "AF"

-----End of SOAP Project-----

2) REST

- 1) Open Soapui Tool
- 2) File -> Import Project (google-maps-soapui-project.xml)
- 3) Create "TestSuite" as "Geocoding API_TestSuite"
- 4) Create "TestCase" as "Geocoding API_TestCase"
- 5) Right click add "REST" Test Step as "GET - Sample Request"
 - a. Select "Geocoding API -> GET-> Sample Request" method
 - b. Run and Verify the response

```
{
  "results": [ {
    "address_components": [
      {
        "long_name": "1600",
        "short_name": "1600",
        "types": ["street_number"]
      },
      {
        "long_name": "Amphitheatre Parkway",
        "short_name": "Amphitheatre Pkwy",
        "types": ["route"]
      },
      {
        "long_name": "Mountain View",
        "short_name": "Mountain View",
        "types": [
          "locality",
          "political"
        ]
      },
      {
        "long_name": "Santa Clara County",
        "short_name": "Santa Clara County",
        "types": [
          "administrative_area_level_2",
          "political"
        ]
      },
      {
        "long_name": "California",
        "short_name": "CA",
        "types": [
          "administrative_area_level_1",
          "political"
        ]
      }
    ]
  }
],
```




6) Set Assertion

- a. Valid http status code as “200”
 - b. Contains “California”
 - c. Not contains “Sweden”
-

7) Right click add “REST” Test Step as “GET - Region Biasing Sample”

- a. Select “Geocoding API -> GET-> Region Biasing Sample” method
- b. Run and Verify the response

```
{
  "results": [ {
    "address_components": [
      {
        "long_name": "Toledo",
        "short_name": "Toledo",
        "types": [
          "locality",
          "political"
        ]
      },
      {
        "long_name": "Toledo",
        "short_name": "Toledo",
        "types": [
          "administrative_area_level_4",
          "political"
        ]
      },
      {
        "long_name": "Vega de Toledo",
        "short_name": "Vega de Toledo",
        "types": [
          "administrative_area_level_3",
          "political"
        ]
      },
      {
        "long_name": "Toledo",
        "short_name": "TO",
        "types": [
          "administrative_area_level_2",
          "political"
        ]
      }
    ]
  },
  {
    "address_components": [
      {
        "long_name": "Toledo",
        "short_name": "Toledo",
        "types": [
          "locality",
          "political"
        ]
      },
      {
        "long_name": "Toledo",
        "short_name": "Toledo",
        "types": [
          "administrative_area_level_4",
          "political"
        ]
      },
      {
        "long_name": "Vega de Toledo",
        "short_name": "Vega de Toledo",
        "types": [
          "administrative_area_level_3",
          "political"
        ]
      },
      {
        "long_name": "Toledo",
        "short_name": "TO",
        "types": [
          "administrative_area_level_2",
          "political"
        ]
      }
    ]
  }
],
  "status": "OK"
}
```



8. Set Assertion

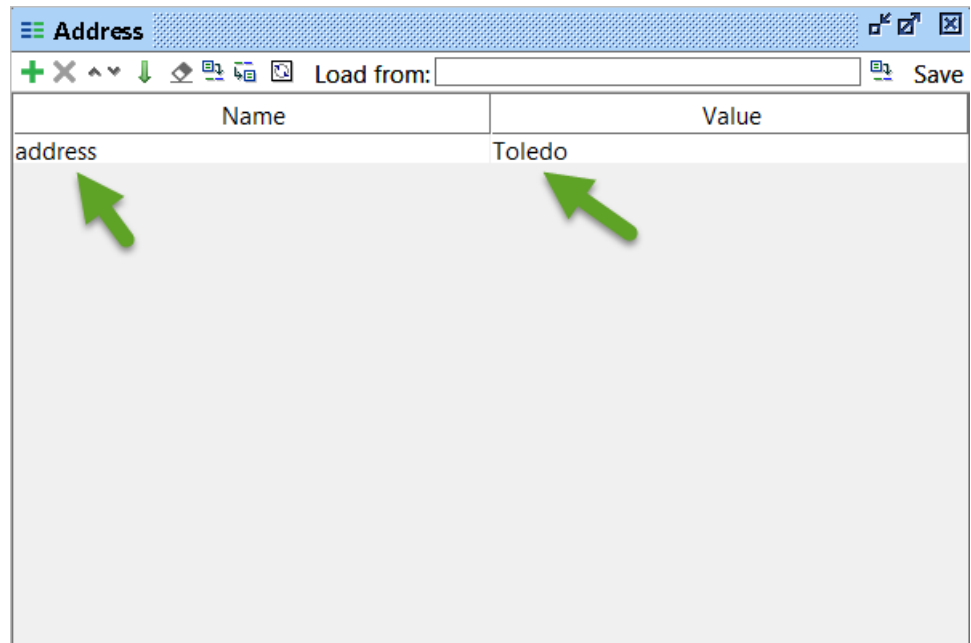
- a. Valid http status code as “200”
 - b. Contains “Spain”
 - c. Not contains “Sweden”
-

9. Properties

a. Right click Insert Step -> Properties

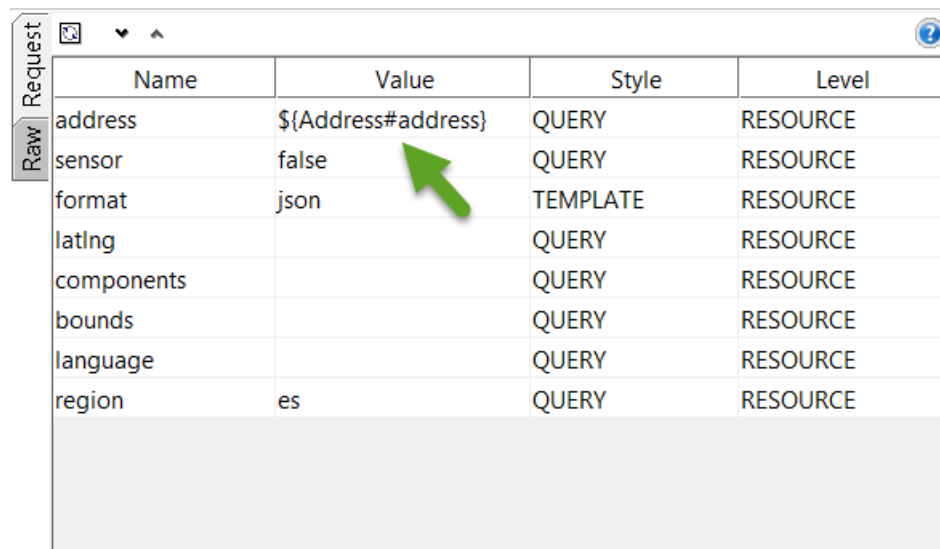
b. Set Properties name as “Address”

c. Inside the Properties Set name as “address” and value as “Toledo”



Name	Value
address	Toledo

11) Set Properties in “GET - Region Biasing Sample”



Name	Value	Style	Level
address	\$(Address#address)	QUERY	RESOURCE
sensor	false	QUERY	RESOURCE
format	json	TEMPLATE	RESOURCE
latlng		QUERY	RESOURCE
components		QUERY	RESOURCE
bounds		QUERY	RESOURCE
language		QUERY	RESOURCE
region	es	QUERY	RESOURCE

12) Right click add REST Test Step as “GET - Reverse Lookup Sample”

- a. Select “Direction API -> GET-> GET - Reverse Lookup Sample” method
- b. Run and Verify the response

```
{
  "results": [
    {
      "address_components": [
        {
          "long_name": "277",
          "short_name": "277",
          "types": ["street_number"]
        },
        {
          "long_name": "Bedford Avenue",
          "short_name": "Bedford Ave",
          "types": ["route"]
        },
        {
          "long_name": "Williamsburg",
          "short_name": "Williamsburg",
          "types": [
            "neighborhood",
            "political"
          ]
        },
        {
          "long_name": "Brooklyn",
          "short_name": "Brooklyn",
          "types": [
            "sublocality_level_1",
            "sublocality",
            "political"
          ]
        }
      ]
    }
  ]
}
```



13) Set Assertion

- a. Valid http status code as “200”
- b. Contains “Brooklyn”
- c. Not contains “Stockholm”

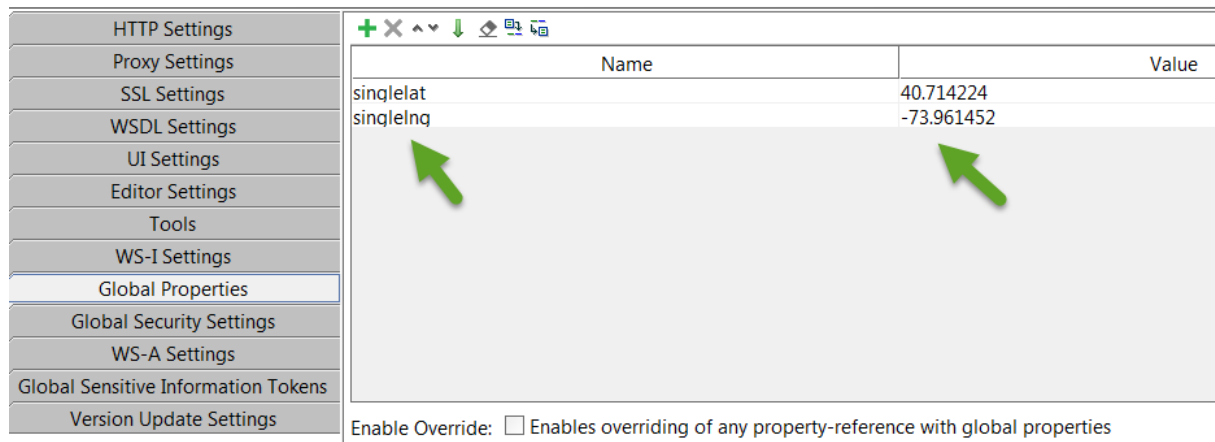
14) Set Global Properties

- a. Right click Insert Step -> Groovy Script
- b. Set name as “Geocoding API”
- c. Write groovy script



```
1 import groovy.xml.*
2
3 def singlelat = "40.714224"
4
5 log.info(singlelat)
6
7 com.eviware.soapui.SoapUI.globalProperties.setPropertyValue( "singlelat", singlelat)
8
9 def singlelng = "-73.961452"
10
11 log.info(singlelat)
12
13 com.eviware.soapui.SoapUI.globalProperties.setPropertyValue( "singlelng", singlelng)
14
15
```

- d. Run the groovy script
- e. Verify in “Global Property” File-> Preferences -> Global Properties



Name	Value
singlelat	40.714224
singlelng	-73.961452

Enable Override: ☐ Enables overriding of any property-reference with global properties


15) Set Properties in “GET - Reverse Lookup Sample” Test Step

Endpoint
http://maps.googleapis.com

Resource/Method: **GET** **neocode -> GFT** [/maps/api/geocode/json?sensor=false&]

Raw Request

Name	Value	Style	Level
address		QUERY	RESOURCE
sensor	false	QUERY	RESOURCE
format	json	TEMPLATE	RESOURCE
latlng	\${singlelat},\${singlelng}	QUERY	RESOURCE
components		QUERY	RESOURCE
bounds		QUERY	RESOURCE
language		QUERY	RESOURCE
region		QUERY	RESOURCE



16) Right click add REST Test Step as “GET - Viewport Biasing Sample”

- Select “Direction API -> GET -> Viewport Biasing Sample” method
- Run and Verify the response

```

{
  "results": [ {
    "address_components": [
      {
        "long_name": "Winnetka",
        "short_name": "Winnetka",
        "types": [
          "neighborhood",
          "political"
        ]
      },
      {
        "long_name": "Los Angeles",
        "short_name": "Los Angeles",
        "types": [
          "locality",
          "political"
        ]
      },
      {
        "long_name": "Los Angeles County",
        "short_name": "Los Angeles County",
        "types": [
          "administrative_area_level_2",
          "political"
        ]
      },
      {
        "long_name": "California",
        "short_name": "CA",
        "types": [
          "administrative_area_level_1",
          "political"
        ]
      }
    ]
  }
],

```

17) Set Assertion

- Valid http status code as “200”
- Contains “Winnetka”
- Not contains “Stockholm”

18) Fetch value from “GET - Reverse Lookup Sample” Response

- Right click Insert Step -> Groovy Script
- Set name as “FetchResponse_Reverse”
- Fetch lat and lng value for both “northeast” and “southwest”

```

"geometry": {
  "location": {
    "lat": 40.714232,
    "lng": -73.9612889
  },

```

- Write groovy script

★ FetchResponse_Reverse

```
1 import groovy.json.JsonSlurper
2 import groovy.json.JsonBuilder
3 import groovy.json.JsonOutput
4
5 def responseContent = testRunner.testCase.getTestStepByName("GET - Reverse Lookup Sample").getPropertyValue("response")
6 def response = new JsonSlurper().parseText(responseContent)
7 log.info(response)
8
9 def norlat = new JsonOutput().toJson(response.results.geometry.location.lat.get(0));
10
11 log.info (norlat)
12
13 com.eviware.soapui.SoapUI.globalProperties.setPropertyValue('norlat', norlat);
14
15
16 def norlng = new JsonOutput().toJson(response.results.geometry.location.lng.get(0));
17
18 log.info (norlng)
19
20 com.eviware.soapui.SoapUI.globalProperties.setPropertyValue('norlng', norlng);
```

e. Run the groovy script

f. Verify in “Global Property” File-> Preferences -> Global Properties

SoapUI Preferences

Set global SoapUI settings

Name	Value
singlelat	40.714224
singlelng	-73.961452
norlat	40.714232
norlng	-73.9612889

Enable Override: ☐ Enables overriding of any property-reference with global properties

19) Set Properties in “GET - Viewport Biasing Sample” Test Step

GET - Viewport Biasing Sample


Endpoint
<http://maps.googleapis.com>

Resource/Method: **GET** neocode -> GET [\/maps/api/geocode/json?address=Winnetka&sensor=

Request

Name	Value	Style	Level
address	Winnetka	QUERY	RESOUR...
sensor	false	QUERY	RESOUR...
format	json	TEMPLATE	RESOUR...
latlng		QUERY	RESOUR...
components		QUERY	RESOUR...
bounds	\${norlat},\${norlng} 34.236144,-118.500938	QUERY	RESOUR...
language		QUERY	RESOUR...
region		QUERY	RESOUR...

Raw



-----End of REST Project-----

3) GitHub

- 1) Upload both SOAP and REST Project in existing repository from last class assignment. Check in the repository should contains POM file
 - a. periodictable-soap-project.xml
 - b. google-maps-rest-project.xml
 - c. git status
 - d. git add .
 - e. git commit -m "new file"
 - f. git push

4) Jenkins

2) Create a two project

a. Set Title “Soap – PeriodicTable” for “periodictable-soap-project.xml”

b. Set Title “Rest – GoogleMaps” for “google-maps-rest-project.xml”

S	W	Name ↓	Last Success	Last Failure	Last Duration	
		Soap – PeriodicTable	2 days 10 hr - #1	N/A	37 sec	
		Rest – GoogleMaps	4 days 13 hr - #12	4 days 13 hr - #11	18 sec	

c. Change the file name in config according to the project

Add pre-build step ▾

Build

Root POM

pom.xml

Goals and options

com.smartbear.soapui:soapui-maven-plugin:test -DprojectFile=periodictable-soap-project.xml

Add pre-build step ▾

Build

Root POM

pom.xml

Goals and options

com.smartbear.soapui:soapui-maven-plugin:test -DprojectFile=google-maps-rest-project.xml

d. Run the jobs

e. Check Build is Success

f. Go to Console page and click view text

5) Save your SOAP and REST Project also Jenkins logs in text format in to the folder and set the folder name as your “Full Name”. Zip the folder before upload in to student portal

- 6) Upload your Zip folder in to student portal before 16.pm on Monday (23-05-2016)

<http://studentportal.nackademin.se/mod/assign/view.php?id=6913>

Tema 10



SoapUi - Examination for pass with distinction (VG)