WEB SERVICE

SOAP&REST

**Contents:**

* **SOAP Service**
* **SOAP UI**
* **How to Test SOAP service using SOAP UI**
* **REST Service**
* **How to test Rest API using SOAP UI & Postman**
* **Assertions in SOAP UI**
* **Comparison between SOAP and REST**

**Soap and Rest Services:**

**SOAP:**

* SOAP is an XML-based protocol for accessing web services over HTTP.
* SOAP was developed as an intermediate language so that applications built on various programming languages could talk easily to each other and avoid the extreme development effort.

A SOAP message is an ordinary XML document containing the following elements −

* **Envelope** − The envelope element is the mandatory element in the SOAP message and is used to encapsulate all of the data in the SOAP message.
* **Header** −The header element can be used to contain information such as authentication information or the definition of complex data types.
* **Body** −The body element is the main element which contains the definition of the web methods along with any parameter information if required.
* **Fault** − An optional Fault element that provides information about errors that occur while processing the message.

WSDL is an XML-based file which basically tells the client application what the web service does. It is known as the Web Services Description Language(WSDL).

* The WSDL file makes it very easy for the web service to be implemented in one programming language and called from a different programming language.
* SOAP can only work with XML format. As seen from SOAP messages, all data passed is in XML format.

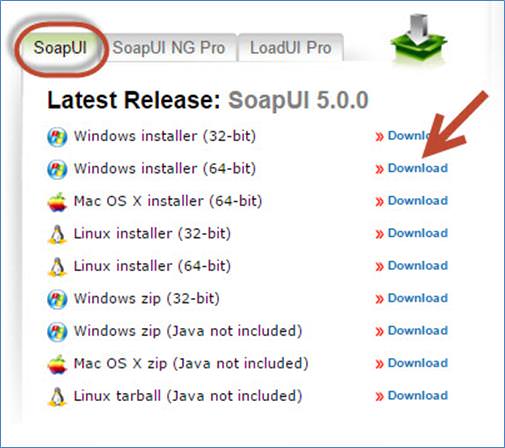
**SOAP Message Structure :**

A SOAP response could look something like this:

<?xml version="1.0"?>  
  
<soap:Envelope  
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"  
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">  
  
<soap:Body>  
  <m:GetPrice xmlns:m="https://www.w3schools.com/prices">  
    <m:Item>Apples</m:Item>  
  </m:GetPrice>  
</soap:Body>  
  
</soap:Envelope>

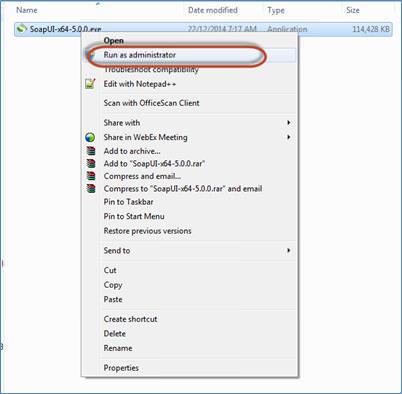
**SOAP UI:**

* SOAP UI is the leading open source cross-platform API[Testing](https://www.guru99.com/software-testing.html)tool
* SOAPUI allows testers to execute automated functional, regression, compliance, and load tests on different Web API.
* SOAPUI supports all the standard protocols and technologies to test all kinds of API's.
* **How to download SOAP UI:**
* Navigate to <http://www.soapui.org/>
* Scroll down and choose the downloader based on your operating system. In this tutorial, we will install SOAP UI on a 64 bit Microsoft Windows operating system.



* Upon clicking download, the user is automatically forwarded to [http://sourceforge.net/](http://sourceforge.net/projects/soapui/files/soapui/5.0.0/SoapUI-x64-5.0.0.exe/download?use_mirror=liquidtelecom&r=http%3A%2F%2Fwww.soapui.org%2F&use_mirror=liquidtelecom) and the installer download starts automatically.
* **How to install SOAP UI:**

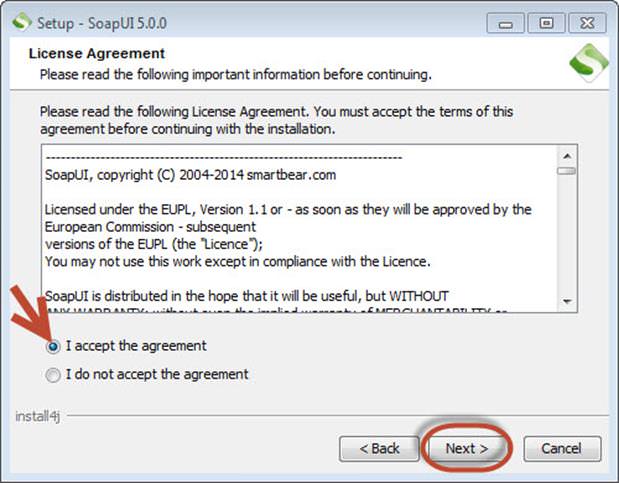
**Step 1:**After downloading, execute the file as 'Administrator' as shown below



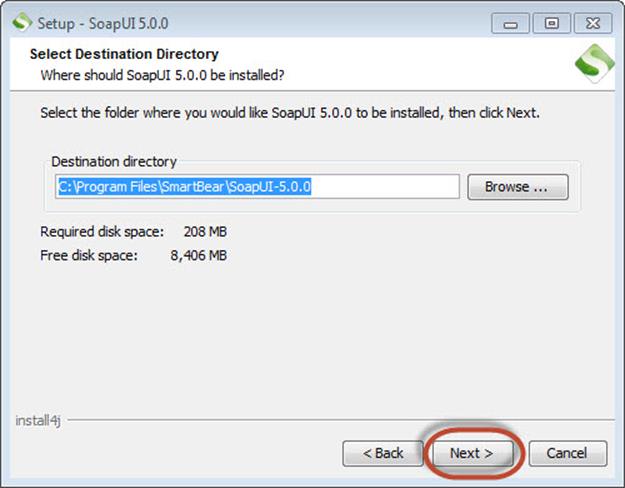
**Step 2:**In the setup wizard, click 'Next' to continue.

.

**Step 3:**Accept the license agreement and click 'Next' to continue.

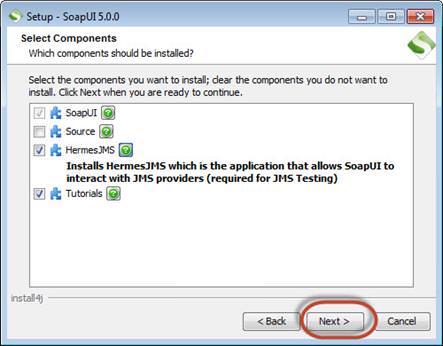


**Step 4:**Choose the installation directory or leave the default installation directory as is.



**Step 5:**Choose the components that you wish to install.

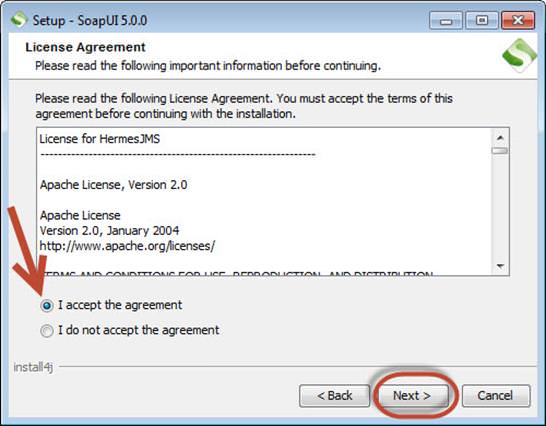
* SOAP UI is checked by default and NOT user configurable.
* Source – Enable, If you would like to get access to the source code of SOAP-UI. We have not selected it.
* Hermes JS – Enable, if the application requires JMS testing.
* Tutorial – Enable, if you want to access SOAP-UI tutorials Post installation.



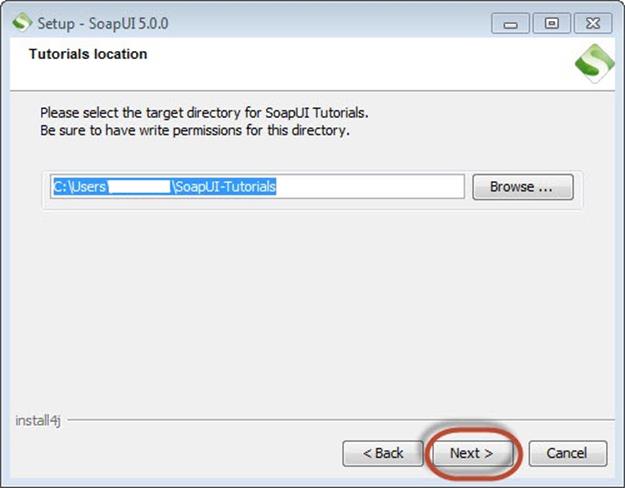
**Step 6:**The installation wizard asks the user to download and install 'Load UI'. Since Load[Testing](https://www.guru99.com/software-testing.html)is not the context of the discussion, we can proceed without selecting it.



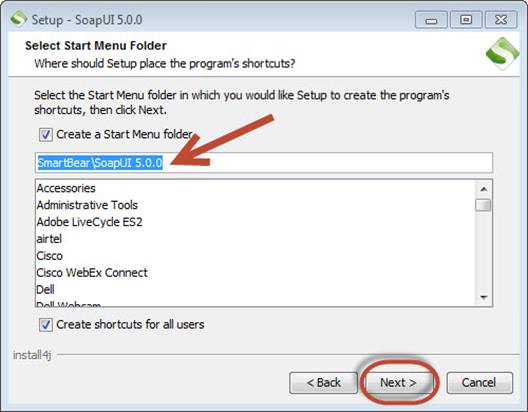
**Step 7:**If**'Hermes JMS'**is selectedin step#5, then the license agreement for 'Hermes JMS' pops up. Accept the license agreement and click 'Next'.



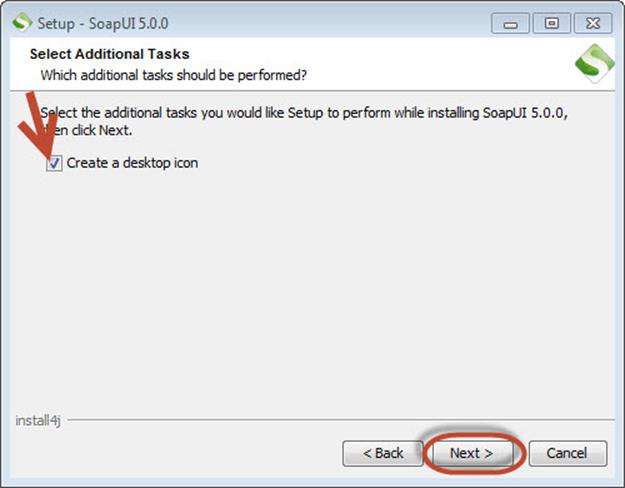
**Step 7:**Choose the folder location for tutorials or else leave the default location as is and click 'Next'.



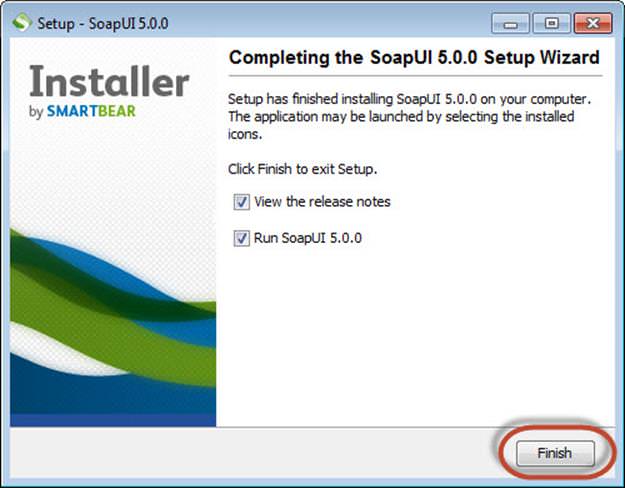
**Step 8:**Choose the start menu folder location or else leave the default location as is and click 'Next'.



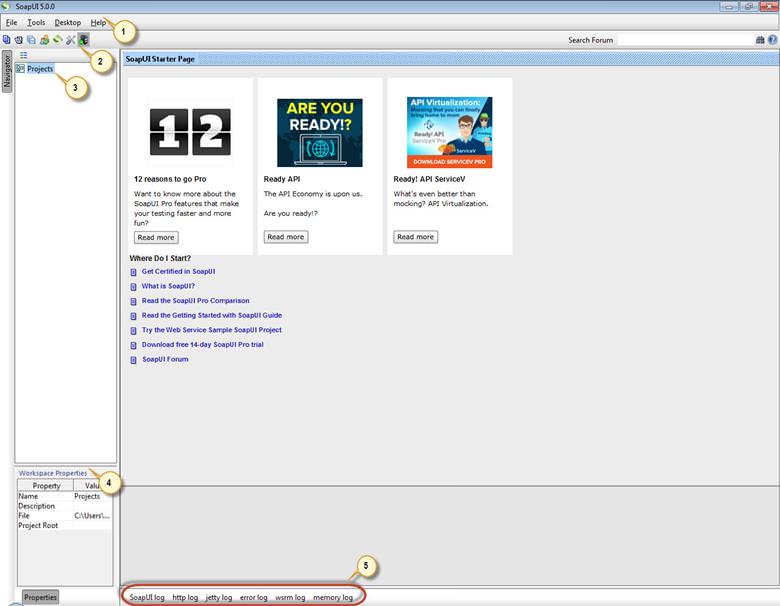
**Step 9:**Enable the checkbox 'create a desktop icon' and click 'Next'.



**Step 10:**The Installation starts and upon completing the same, the wizard shows the below status. Click 'Finish'.



**Step 11:**Upon clicking the 'Finish' button, SOAP UI is launched.

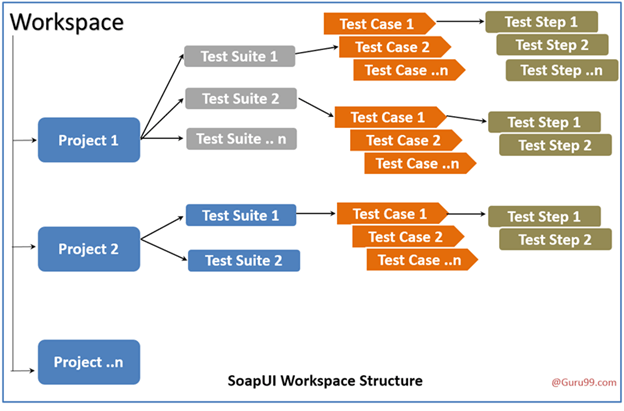


How to Configure SOAP UI:

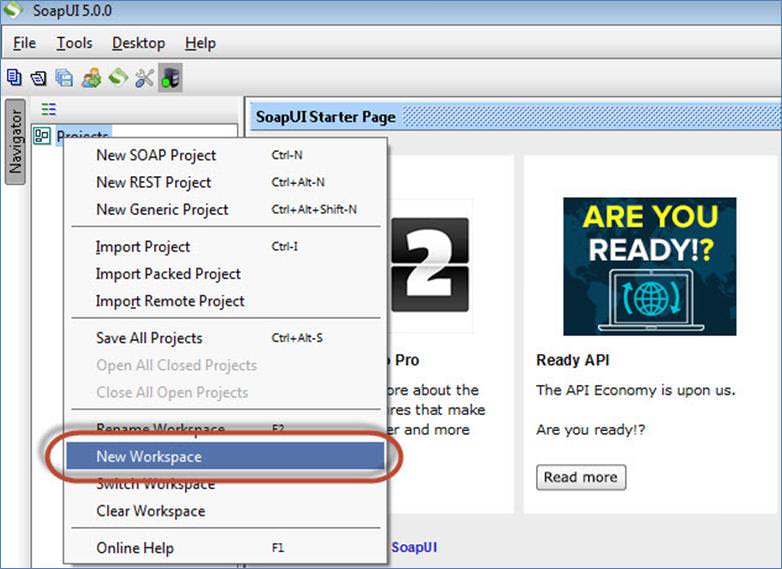
Let us first understand the project structure in SOAP UI.

* First step in SOAP UI is to create a workspace. There can be several projects associated with a workspace. Users can create more than one workspace. In SOAP UI Pro version, we can seamlessly switch environments to map it to different end points.
* For Each project, we can create several test suites.
* For Each Test Suite, there can be several test cases attached to it.
* For Each Test Case, there can be several test steps associated with it.

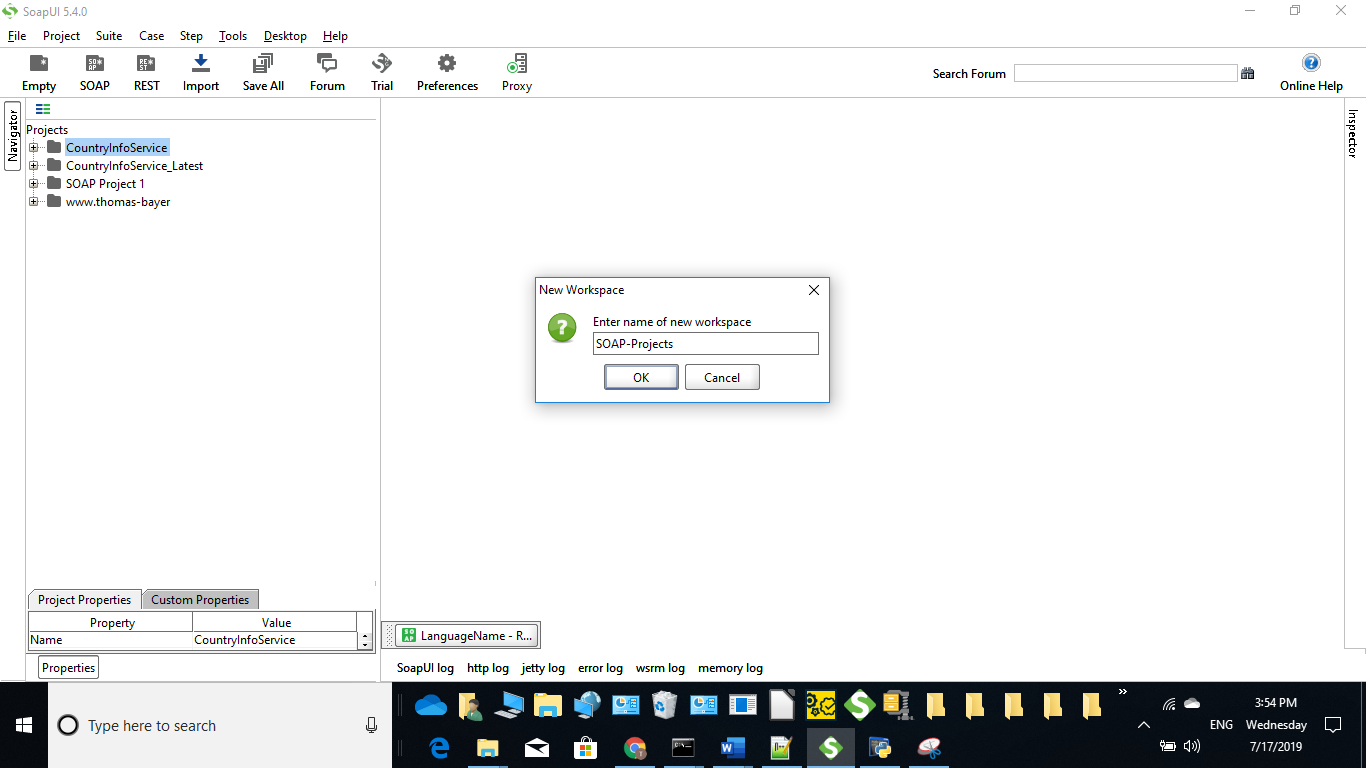
Below is the pictorial representation of a workspace structure in SOAP-UI.



**Step 1:**First step is to create a workspace. Create a workspace as shown below. All the artifacts that we are going to create from now on would be contained in this workspace.

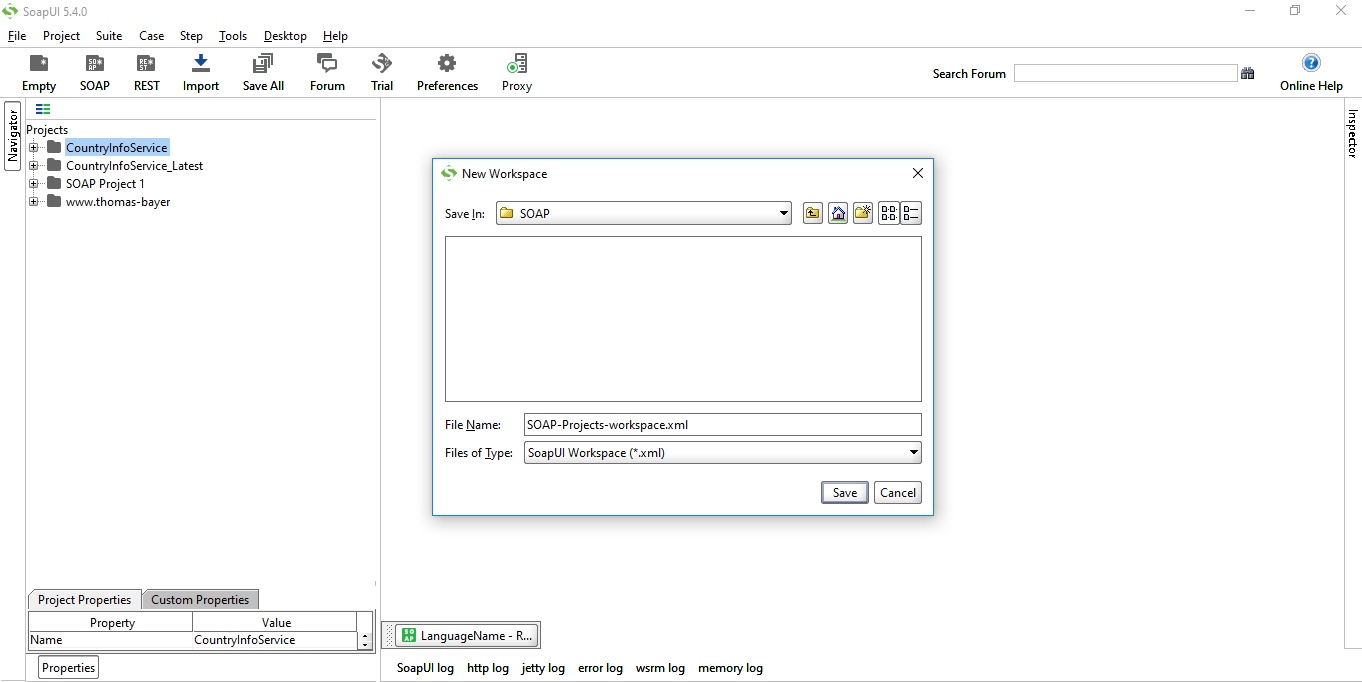


**Step 2:**Enter aname for the workspace and click 'OK.'

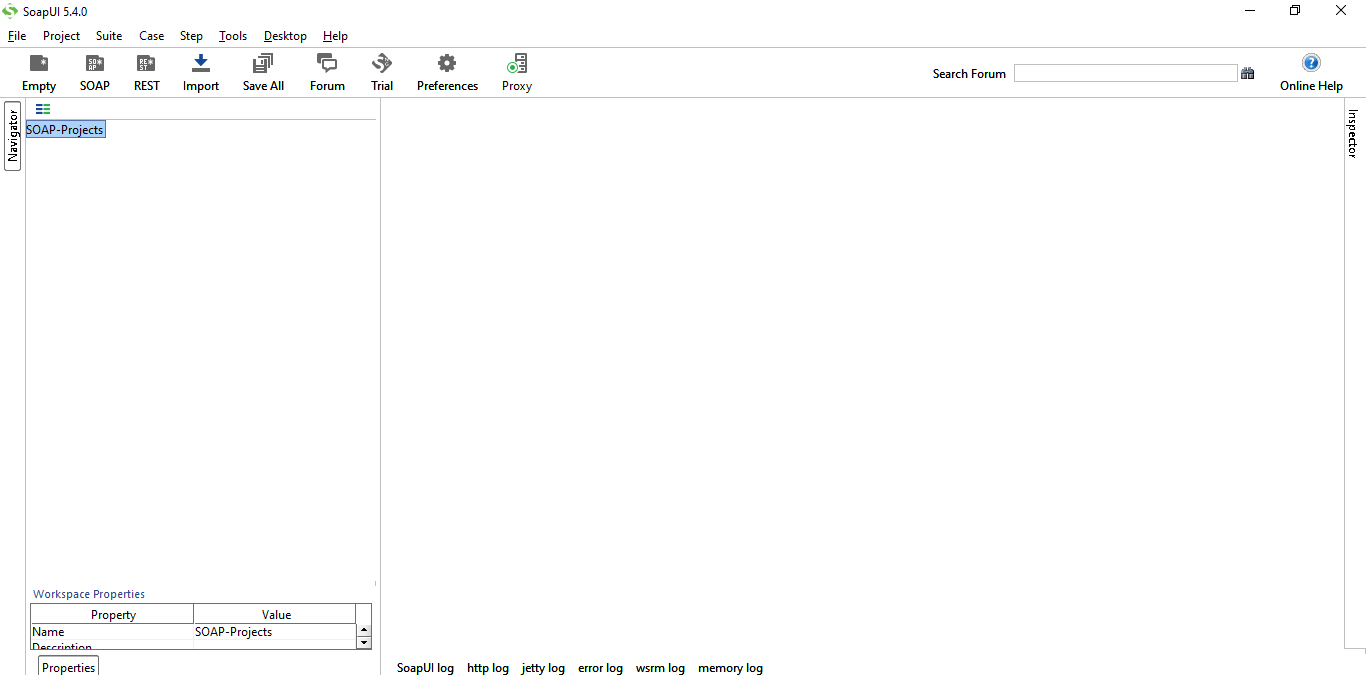


**Step 3:**Now the user has to select the path where this workspace has to be saved.

1. Select the path where the workspace has to be saved
2. The name of the workspace XML, which has to be located when user wants to open the workspace in the future.
3. Click 'Save'.



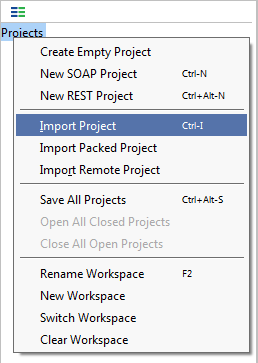
**Step 4:**The workspace is created as shown below. We can also access the workspace properties under 'Workspace Properties' Tab.



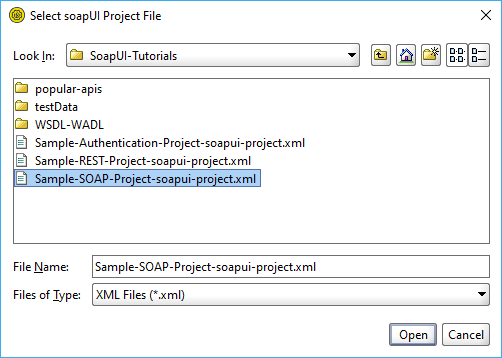
Now we have successfully configured SOAP-UI after downloading and installing such that we can continue to perform testing.

Importing the Project

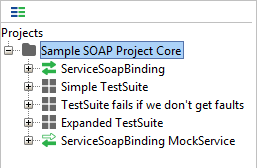
1.Click Import button  import on the main toolbar or right-click the root node in the Navigator panel and select Import Project:



1. In the **Select SoapUI Project File** dialog, select the *Sample-SOAP-Project-soapui-project.xml* file from the *<Home directory>/SoapUI-Tutorials* folder.

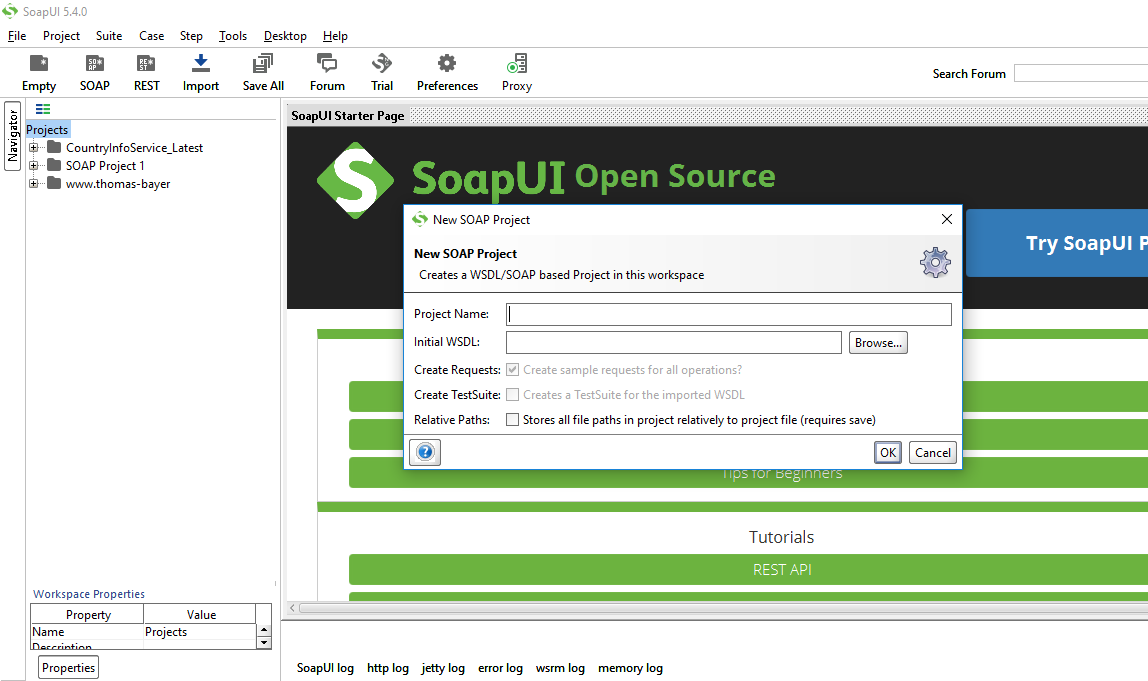


1. The sample project will be shown in the SoapUI Navigator.



**Another way to create SOAP project:**

Click on SOAP project icon



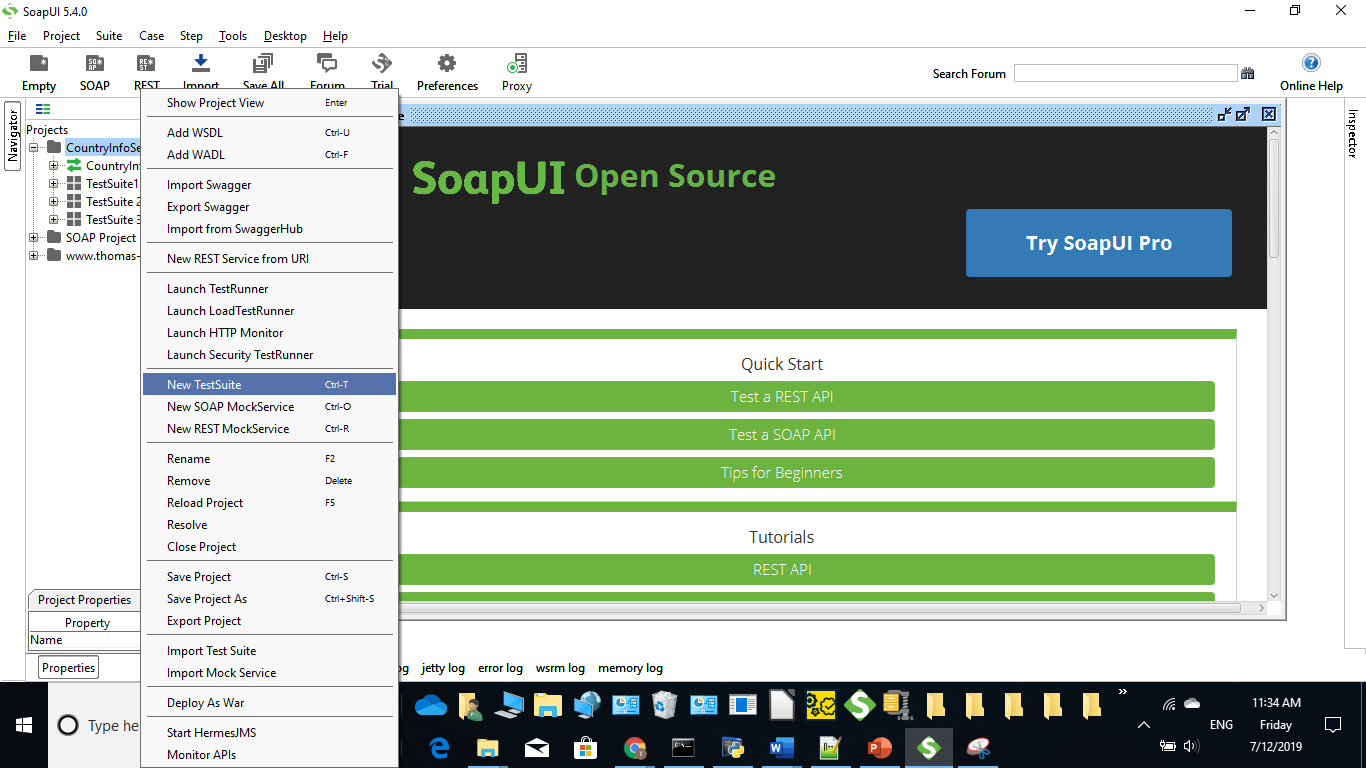
Give Project Name as CountryInfoService\_Latest

Enter wsdl Url as below:

[**http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso?WSDL**](http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso?WSDL)

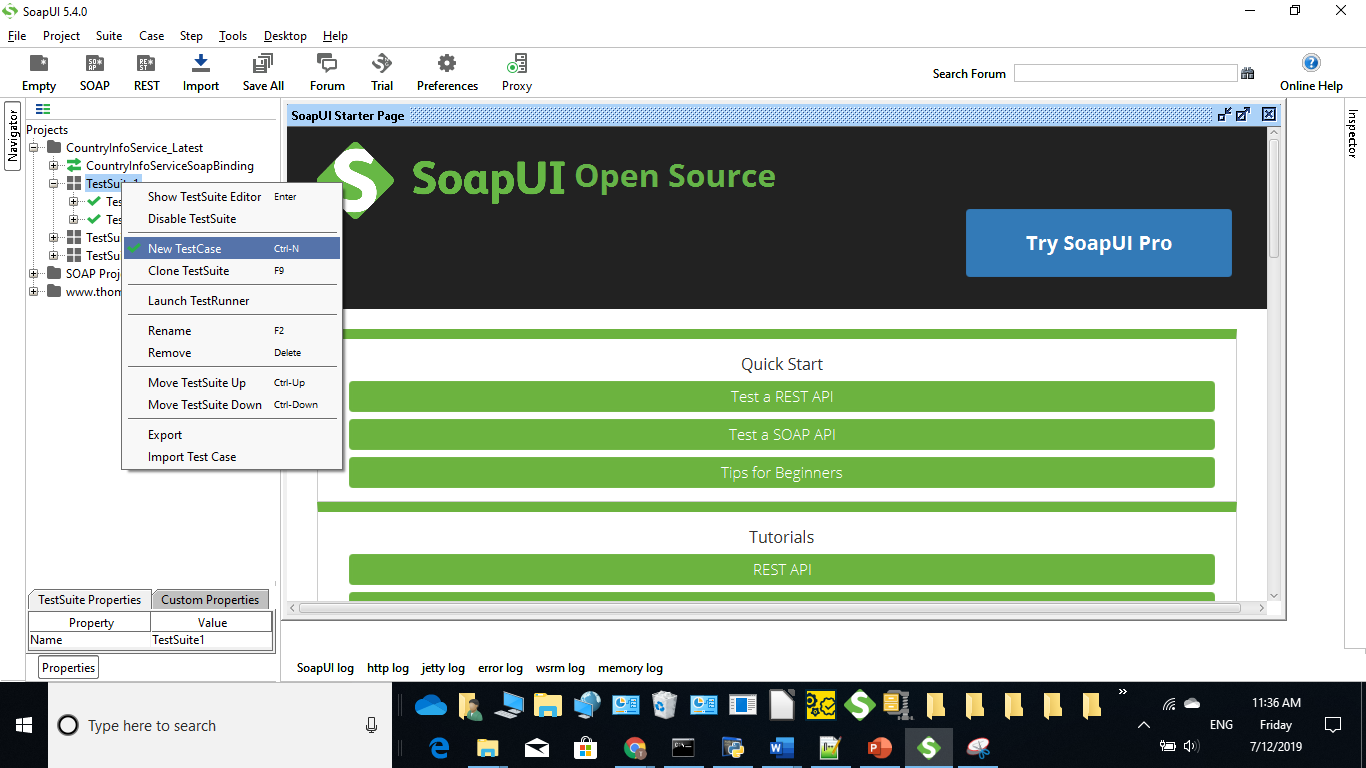
**🡪To Create new Test Suite:**

Right click on Project 🡪click on New Test Suite



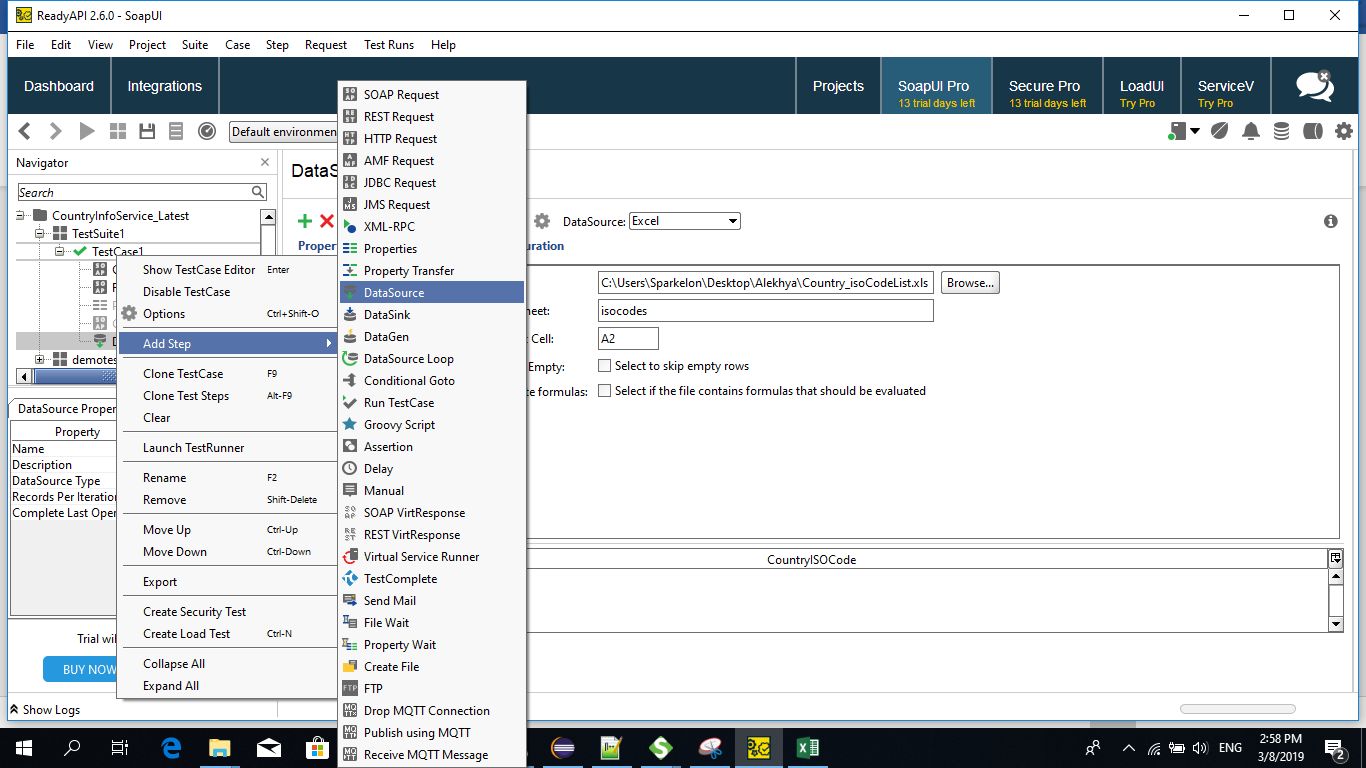
**🡪To create new testcase**

Right click on Test Suite 🡪click on New Test Case



**Adding Data Source:**

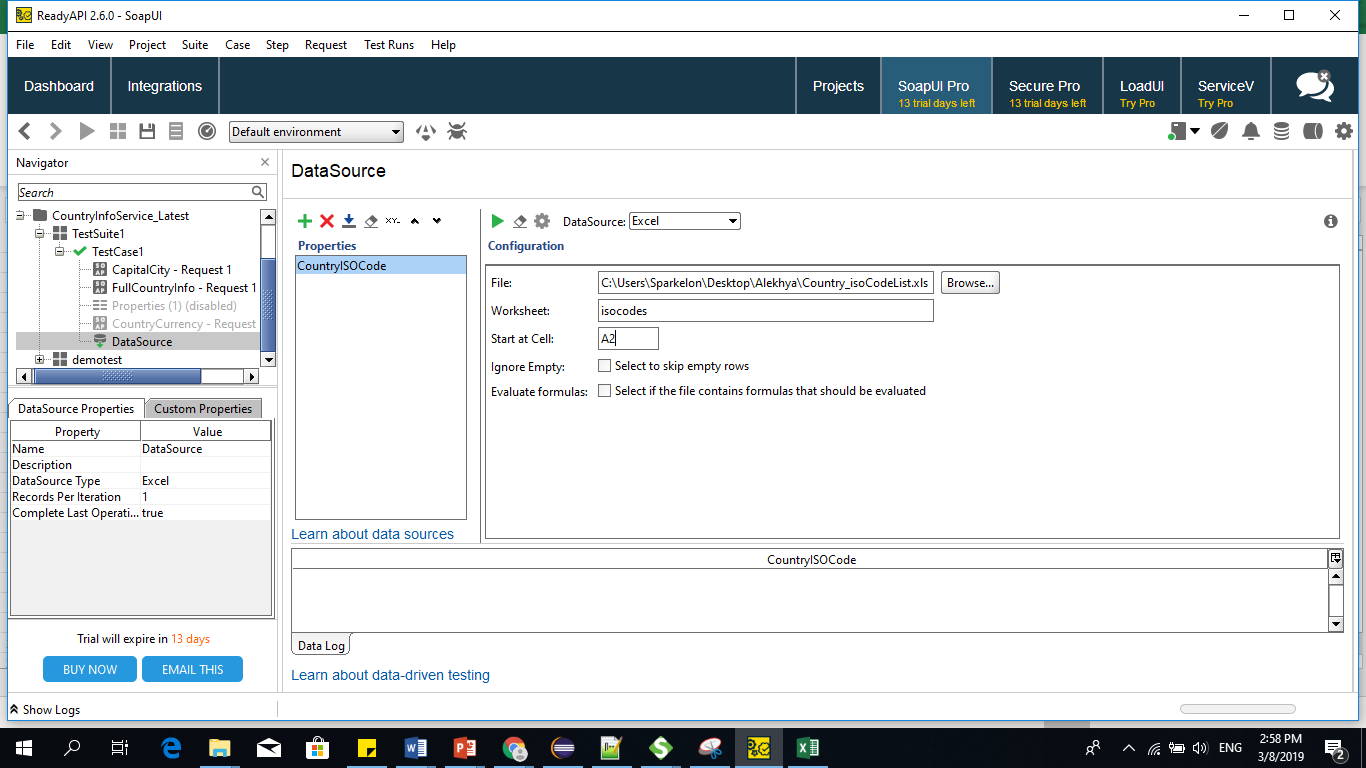
Right click on Test Step🡪Add Step🡪DataSource

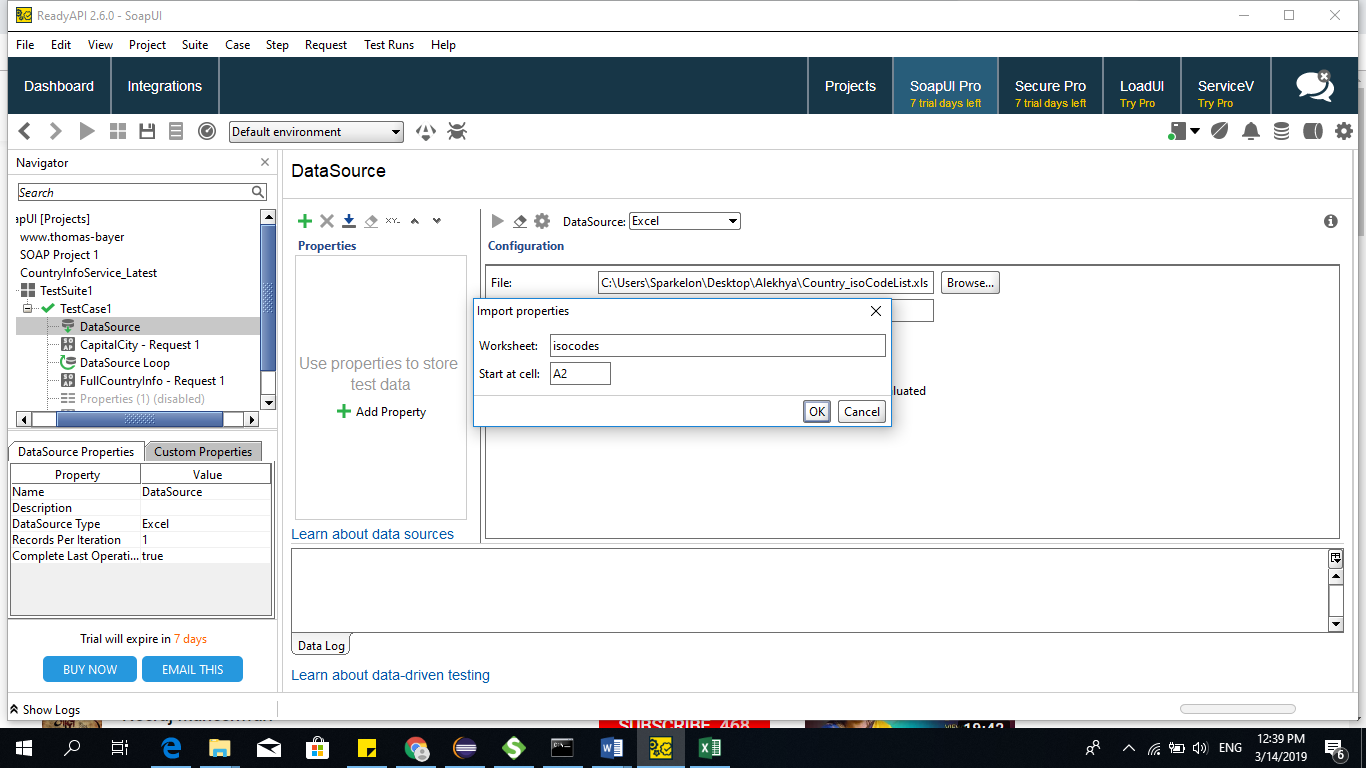


**Importing Excel:**

As shown below, select Data Source as Excel and browse the excel file from local path.

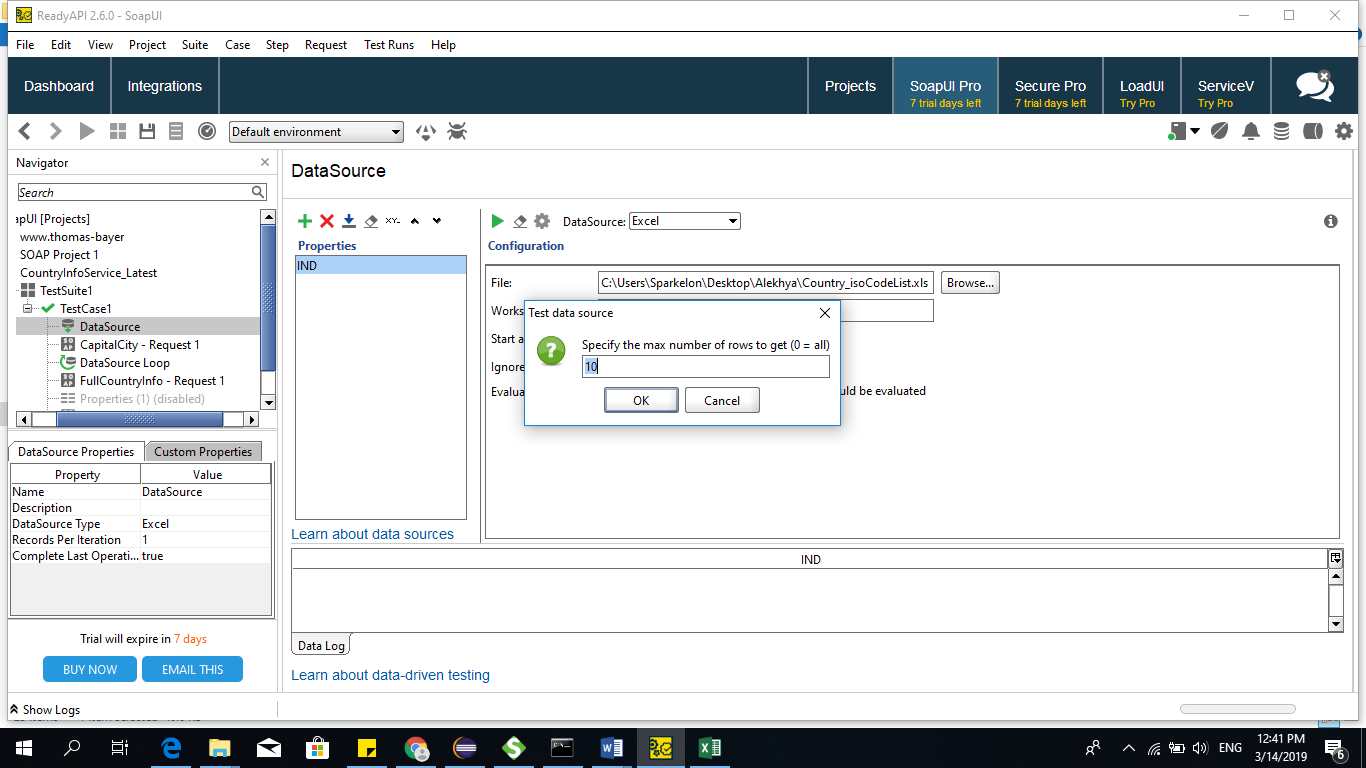
Enter work sheet name and starting cell details



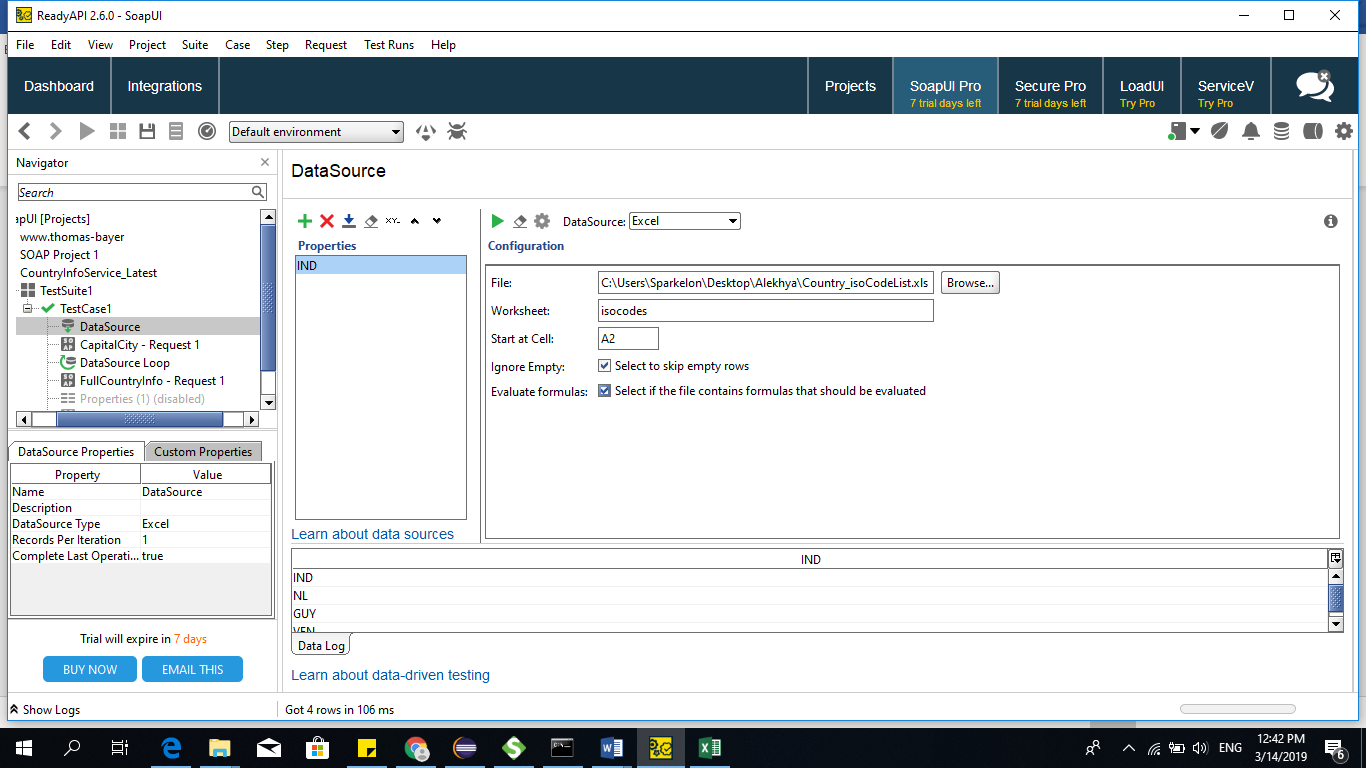


**Run the excel:**

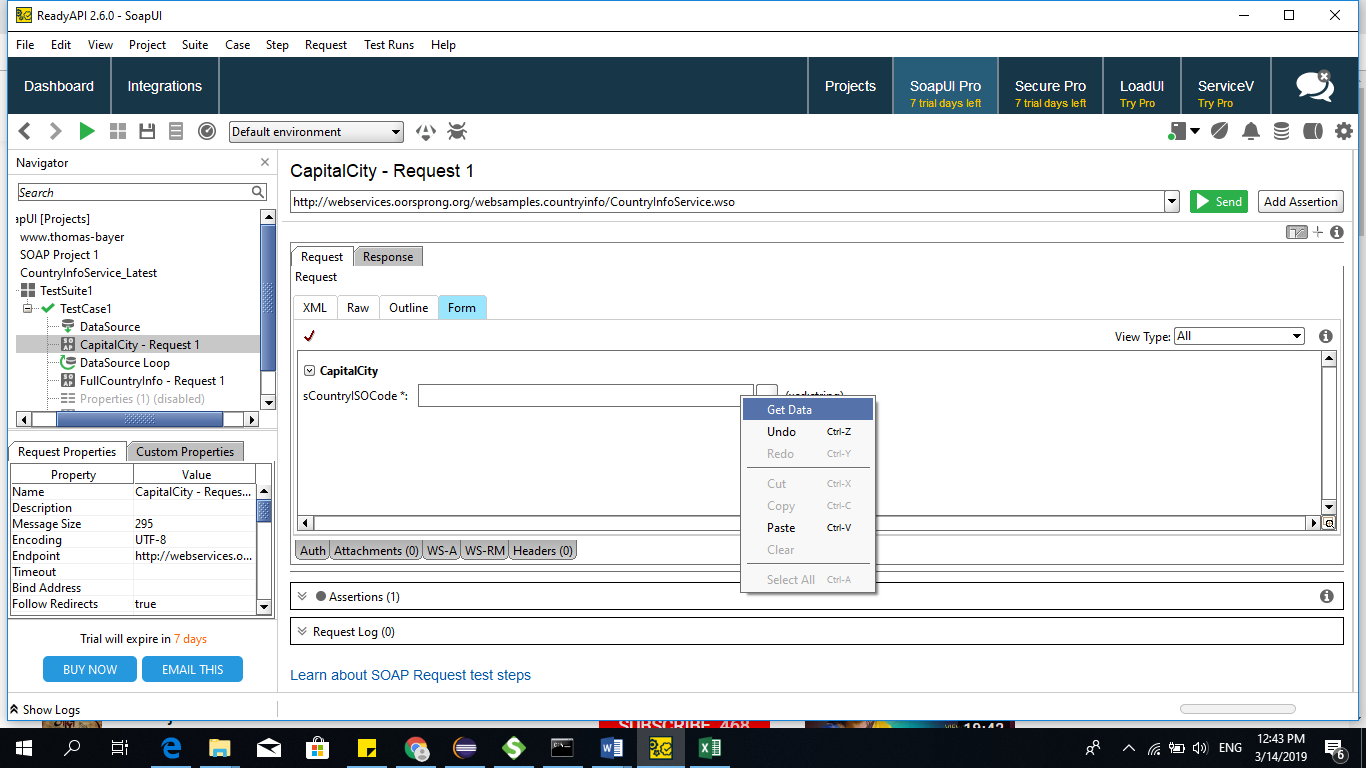
Specify the number of rows

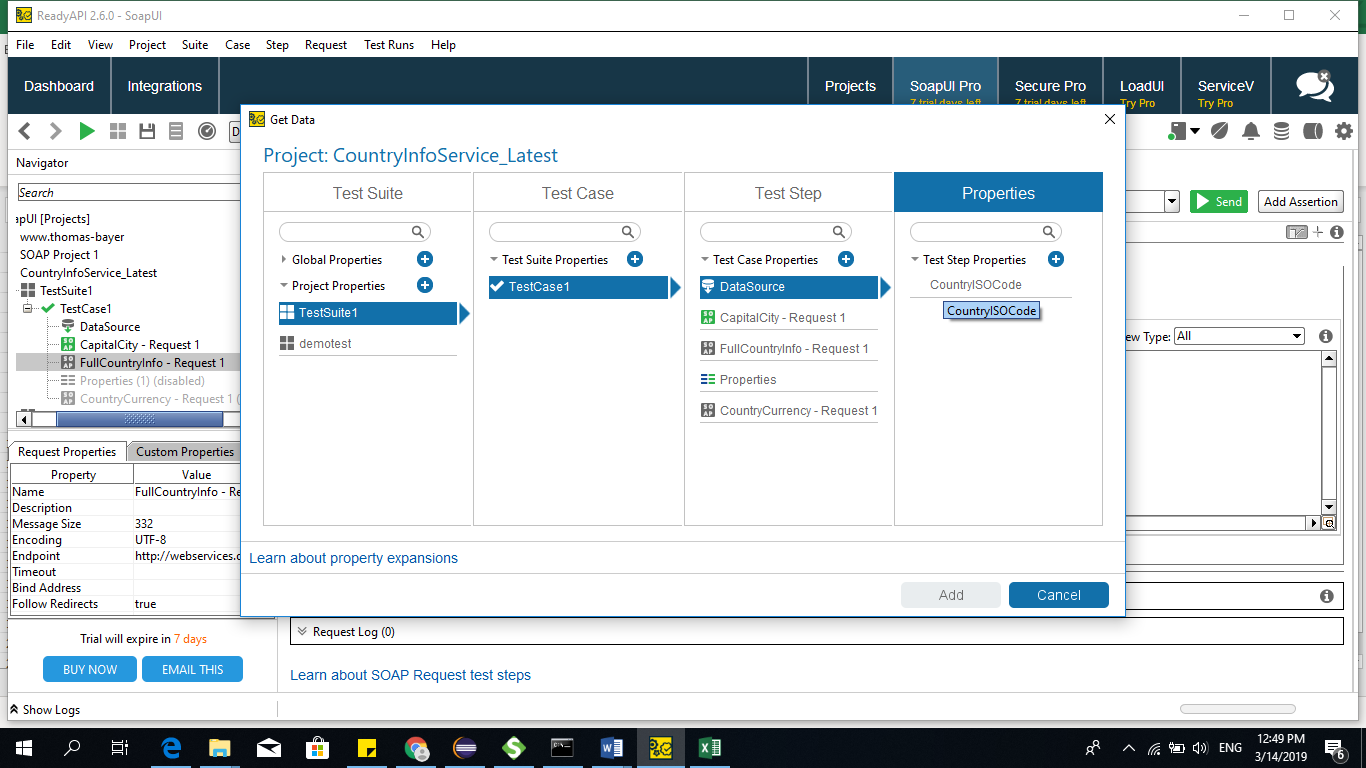


Below is the screenshot for excel data



**Open TestStep**







**CountryISOCode**



**Advantages:**

SOAP is a light-weight protocol that is used for data interchange between applications

**REST Service:**

* REST permits different data format such as Plain text, HTML, XML, JSON, etc. But the most preferred format for transferring data is JSON.

 **POST** – This would be used to create a new employee using the RESTful web service

 **GET** - This would be used to get a list of all employee using the RESTful web service

 **PUT** - This would be used to update all employee using the RESTful web service

 **DELETE** - This would be used to delete all employee using the RESTful web service

* One of the key differences between SOAP and REST is that SOAP is a protocol and REST is an architectural pattern.
* Other key differences between the SOAP and REST protocol is that the requests sent via REST tend to be much lighter than SOAP. Because of this, applications don't require much bandwidth to use REST web services over SOAP.
* **Security** is another major concern with Web services and **SOAP**. REST is good when working with web services open to the public, but if security is required, then the SOAP API has the necessary implementation for the same.
* **REST** has the ability to have **a caching solution** which will help save responses which have been received from the server. In such cases, the client does not need to make the same request to the server and can make use of the cache to get the desired response.

**Response codes:**

* 1xx -means 100 to 199
* 1xx- information based
* 2xx--Success
* 3xx- Redirection
* 4xx-Client error
* 5xx-Server error

**Testing Rest API:**

[**https://github.com/typicode/json-server#install**](https://github.com/typicode/json-server)

**Start JSON Server:**

json-server --watch db.json

**Resources:**

http://localhost:3000/posts

<https://reqres.in/>

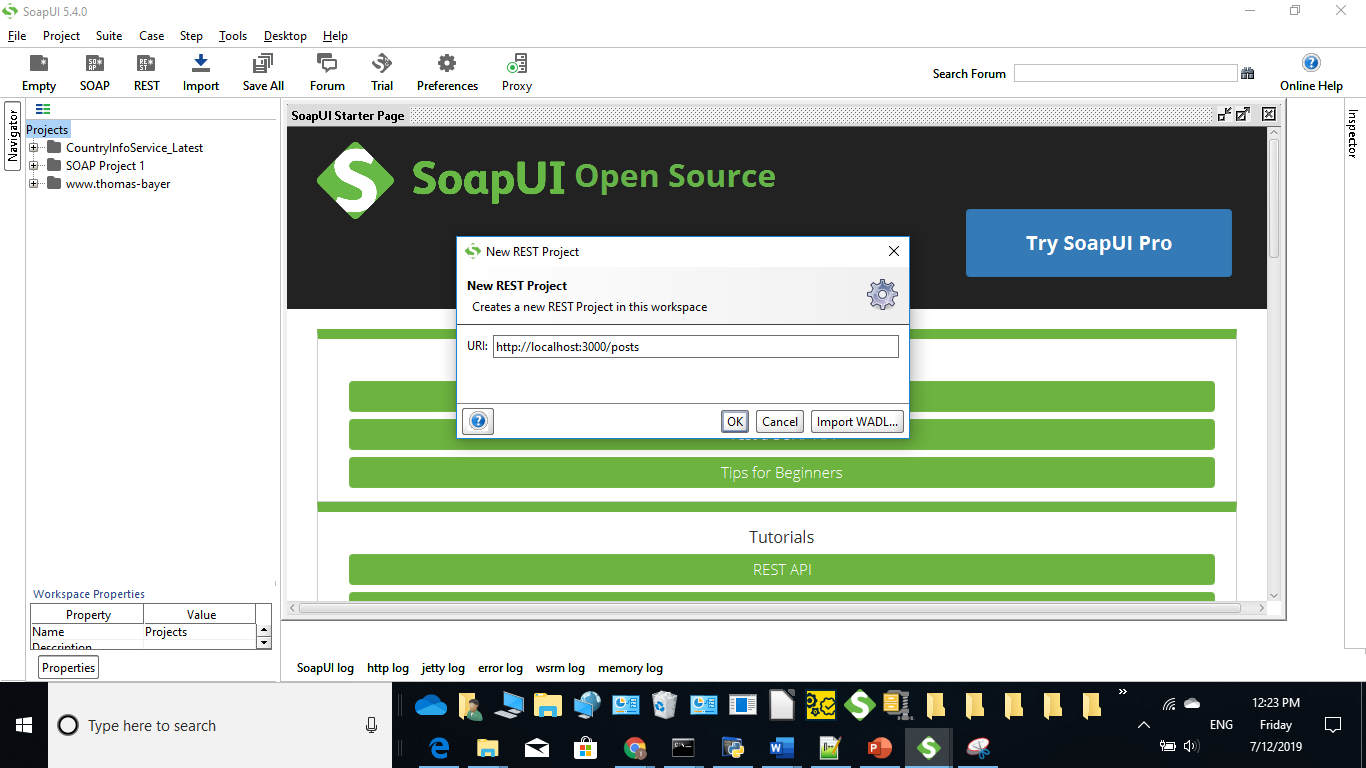
Install JSON Server

npm install -g json-server

Start JSON Server

json-server --watch db.json

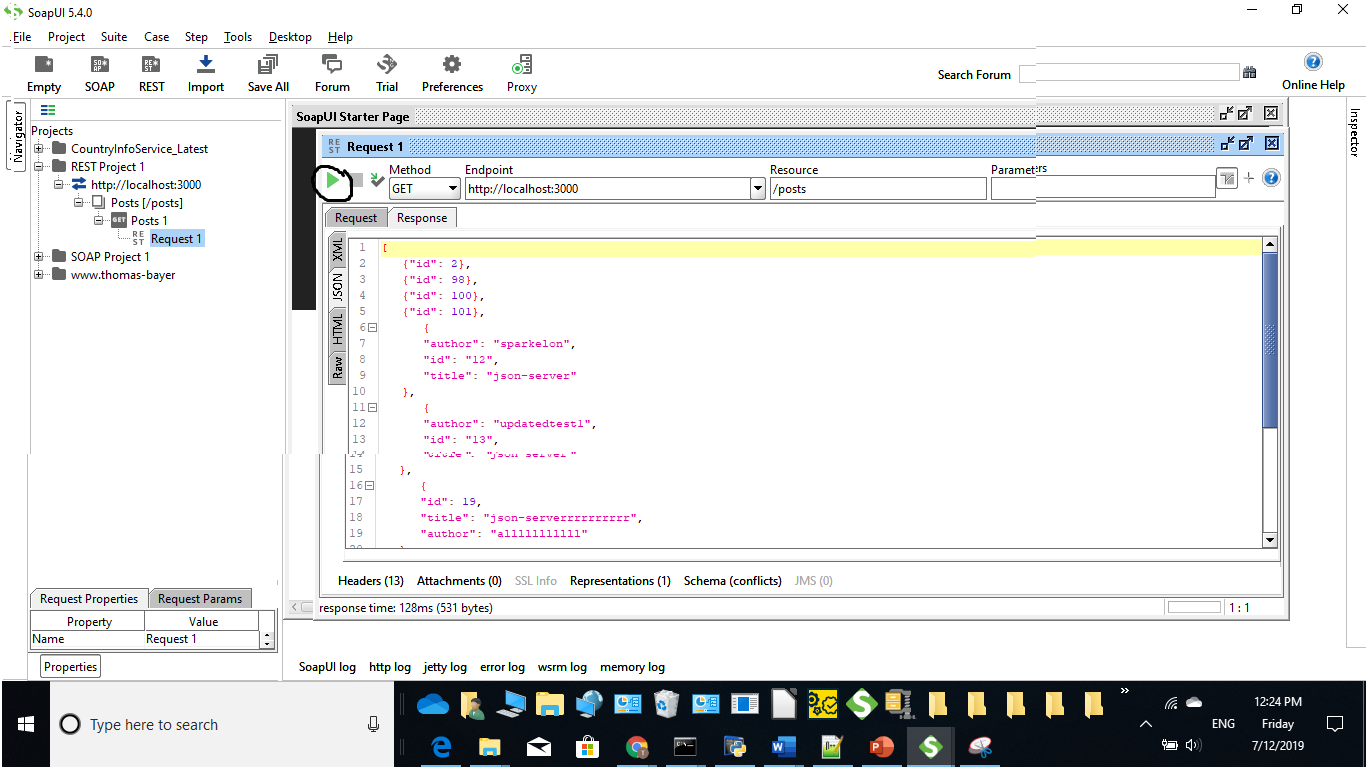
**Rest API testing in SoapUI:**



**Get Method:**

* It will retrieve all the details from webserver for the corresponding input request
* Hit run button as highlighted below

Get method url : http://localhost:3000

****

**Post : To create new resource**

Url: <http://localhost:3000/posts/>

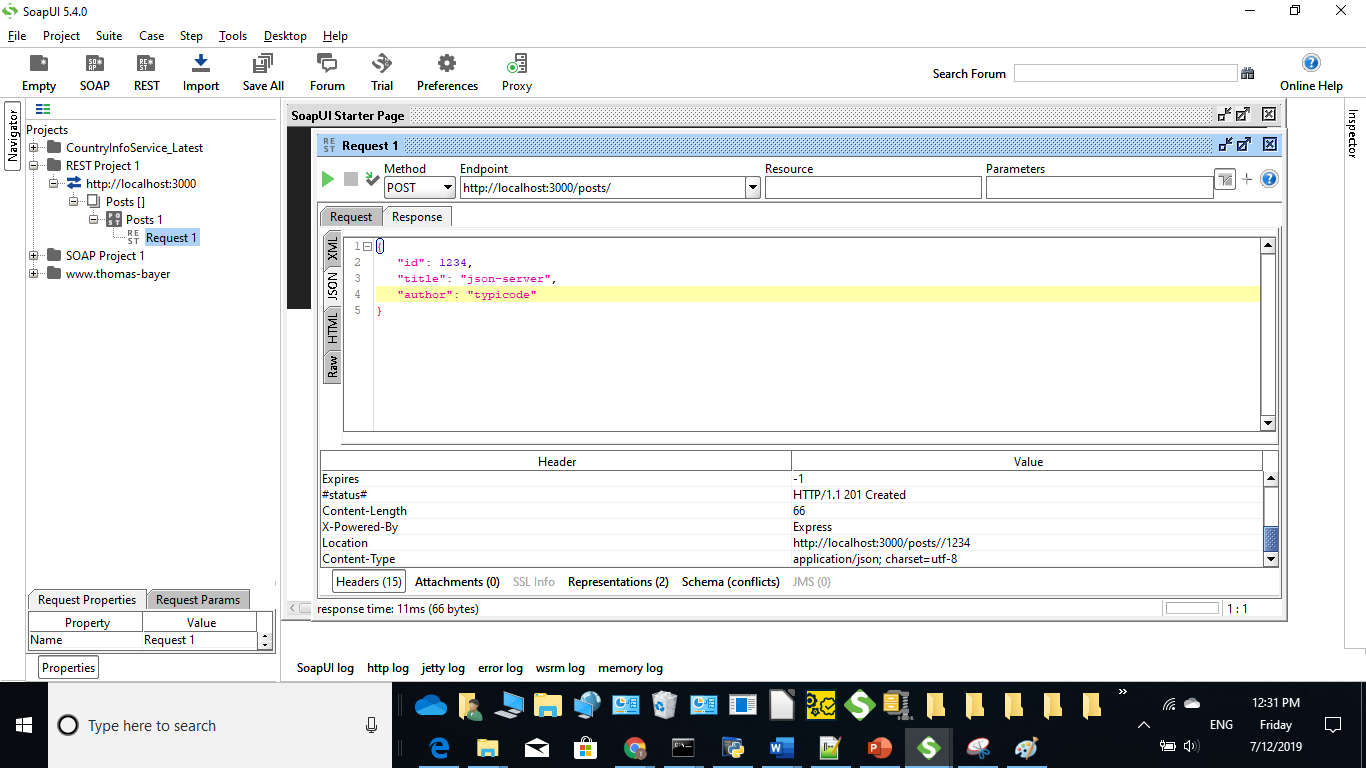
Json request:

{

"id": 1234,

"title": "json-server",

"author": "typicode"}



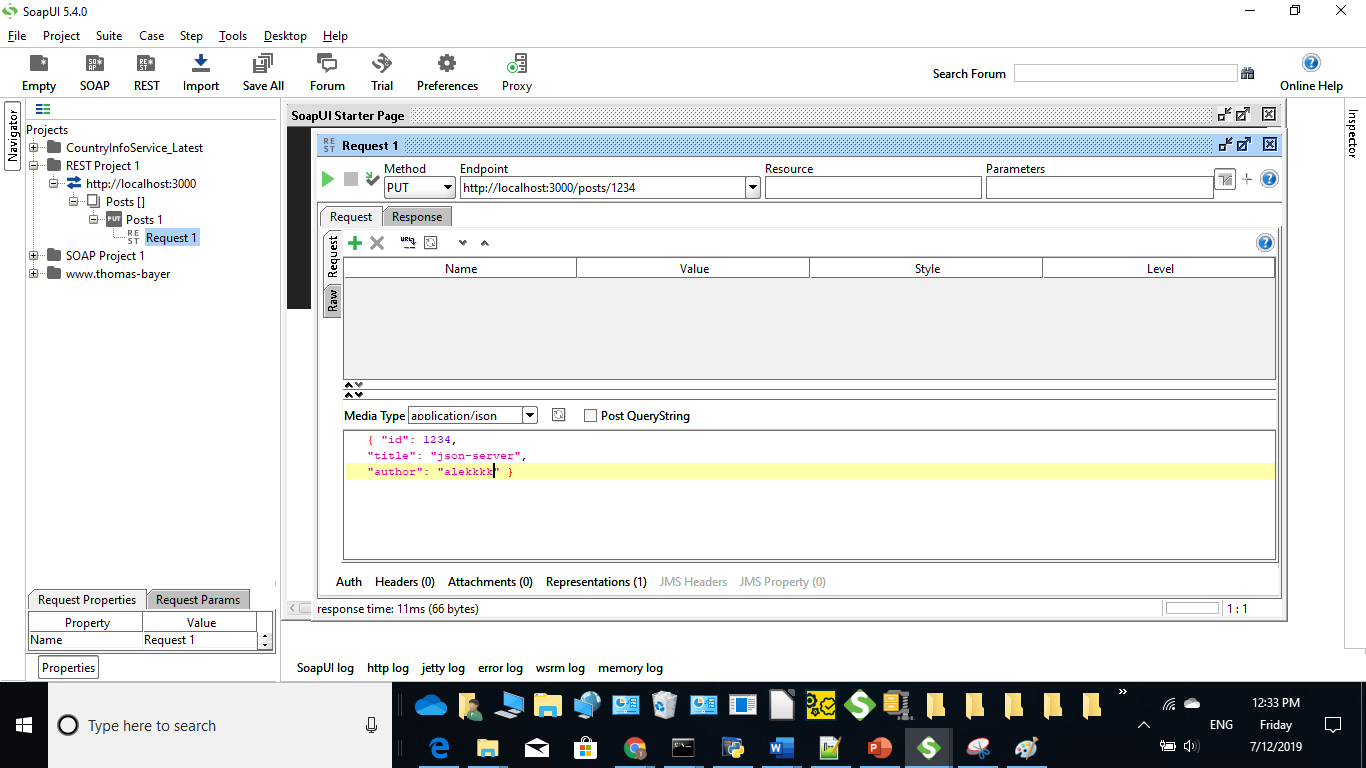
**Put:** To update resource

**Json request:**

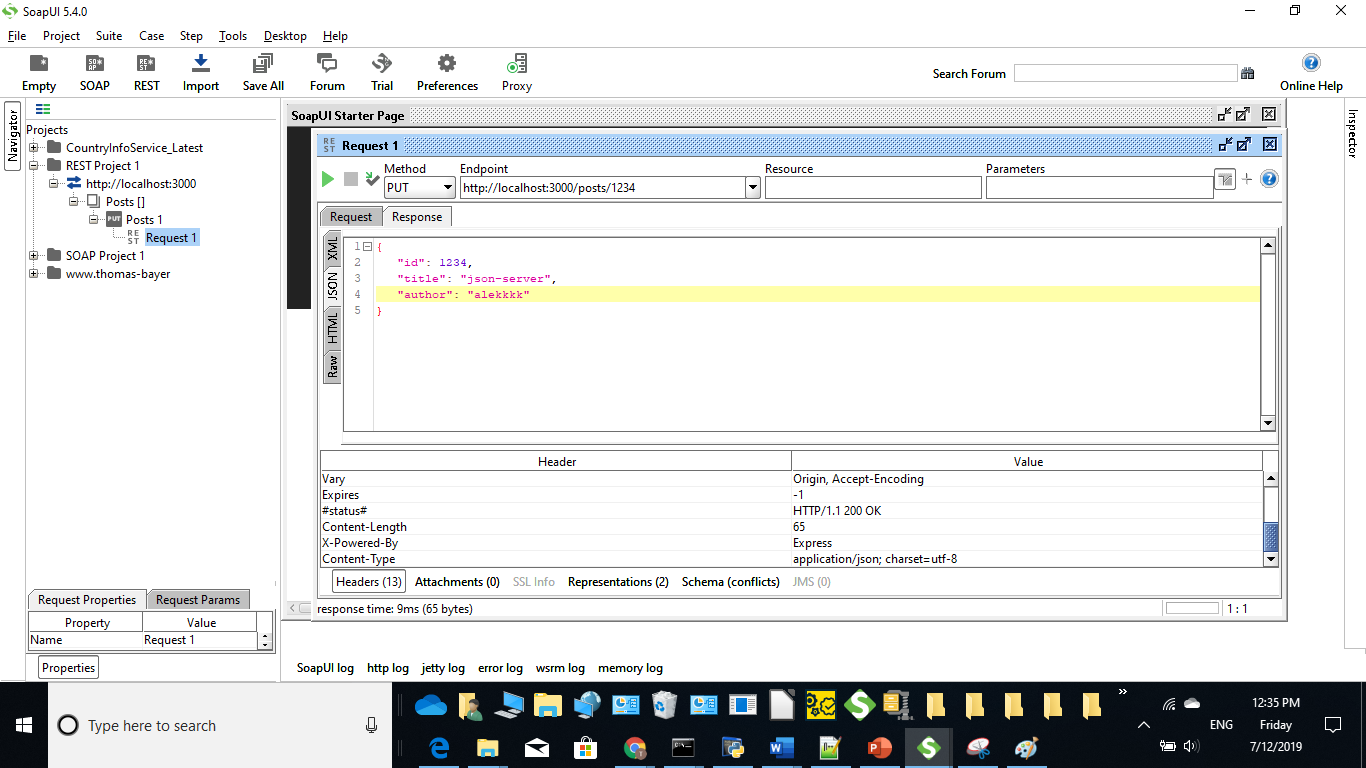
**{ "id": 1234,**

**"title": "json-server",**

**"author": "alekkkk" }**



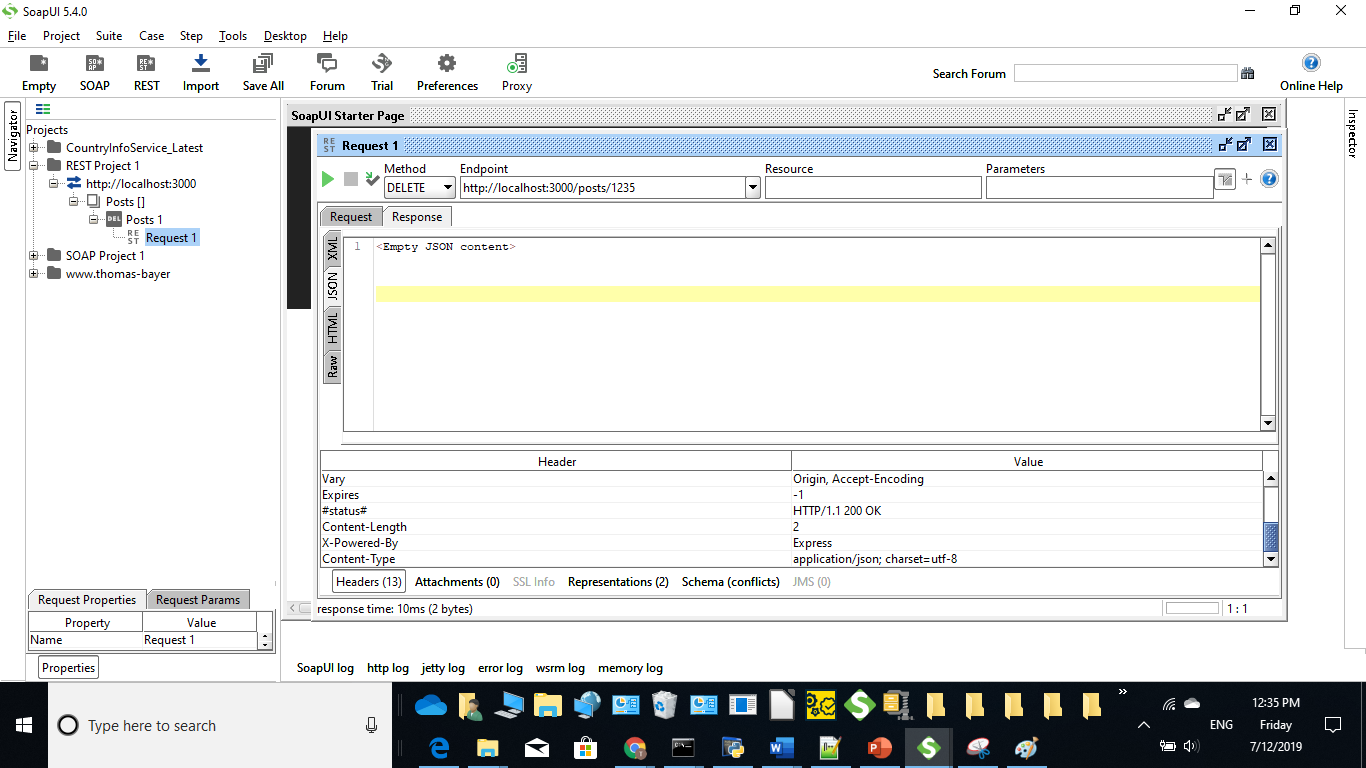
**Response:**



**Delete :**

To delete particular id send url as follows

**url:** [**http://localhost:3000/posts/1235**](http://localhost:3000/posts/1235)



**Rest Api Testing in postman tool:**

http://localhost:3000/posts/

**Post method— will create resource**

**Json format:**

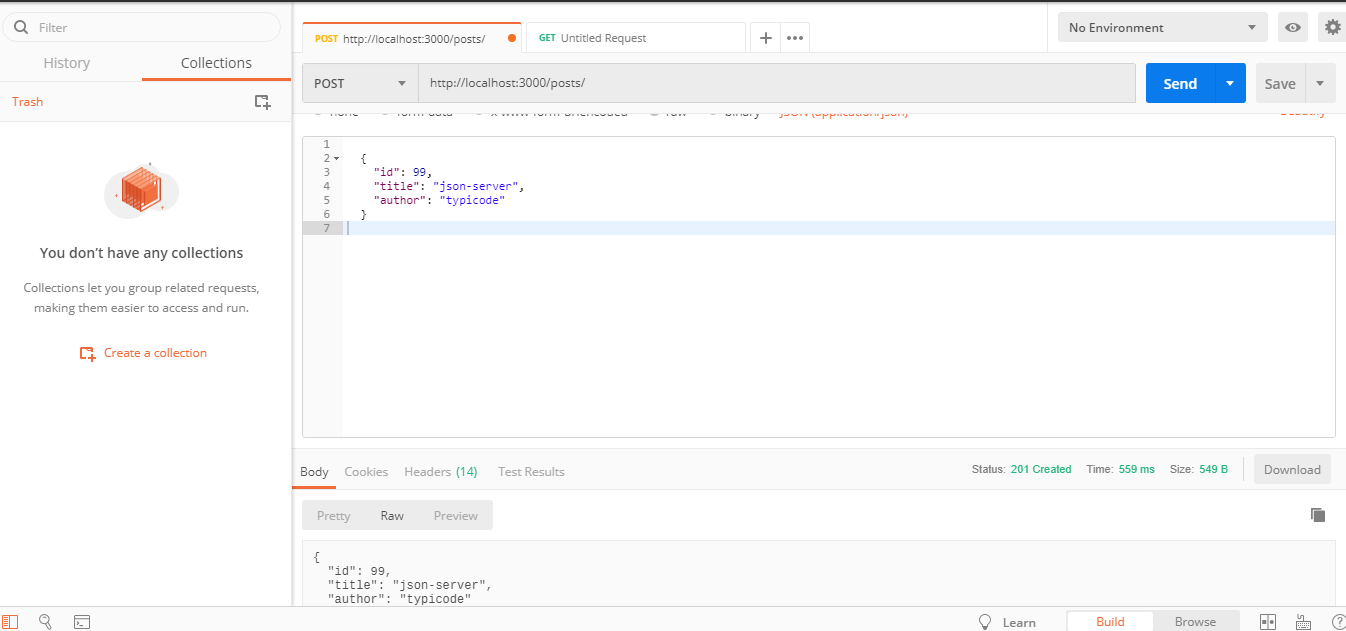
{

Id:”99”

Title:”json-server”

Author:”Typicode”

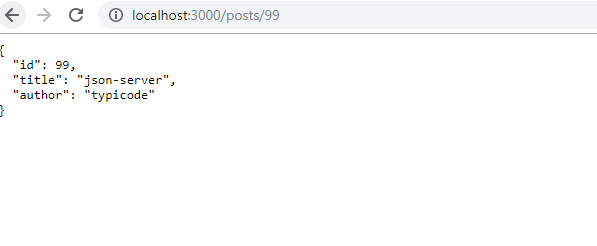
}



In browser we can check as below,



**By passing id number in url we can filter**

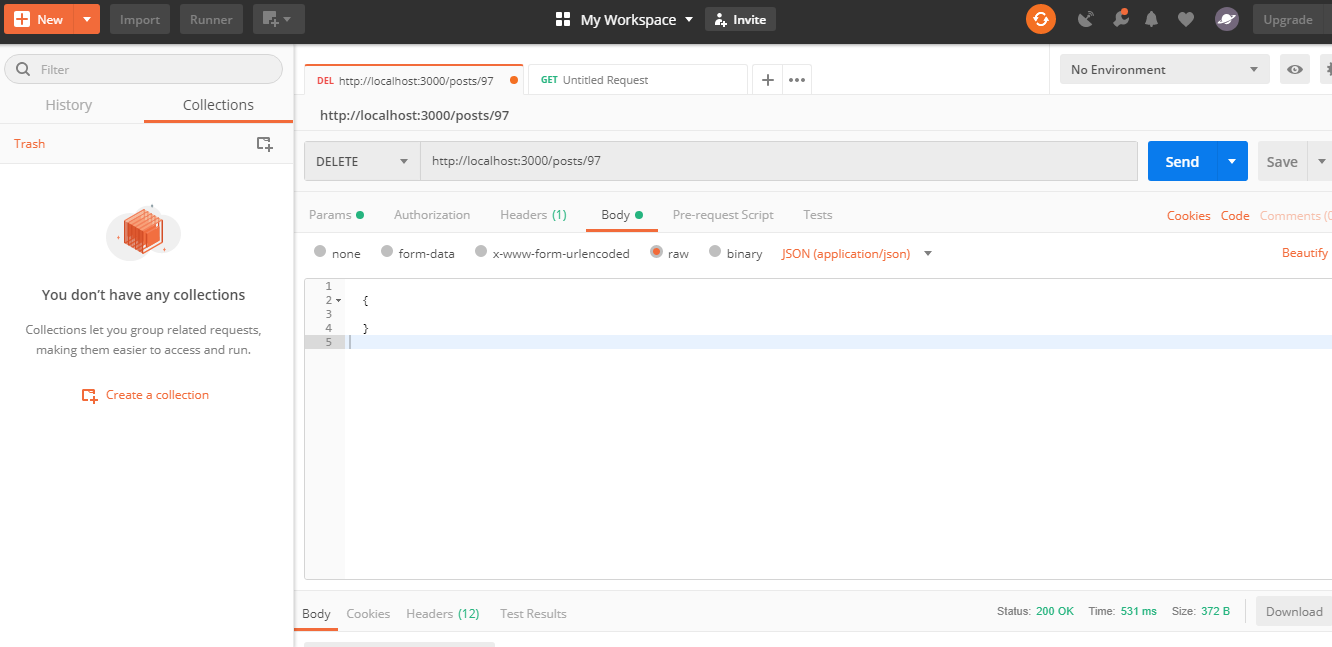


**Delete :**

Planning to delete 97 then send “id” in end point url .

https://localhost:3000/posts/97

Status code – 200 success



**Put –will update data**

**Json request:**

{

"id": 99,

"title": "json-server",

"author": "typicode"

}

to update typicode to ram,then send json request

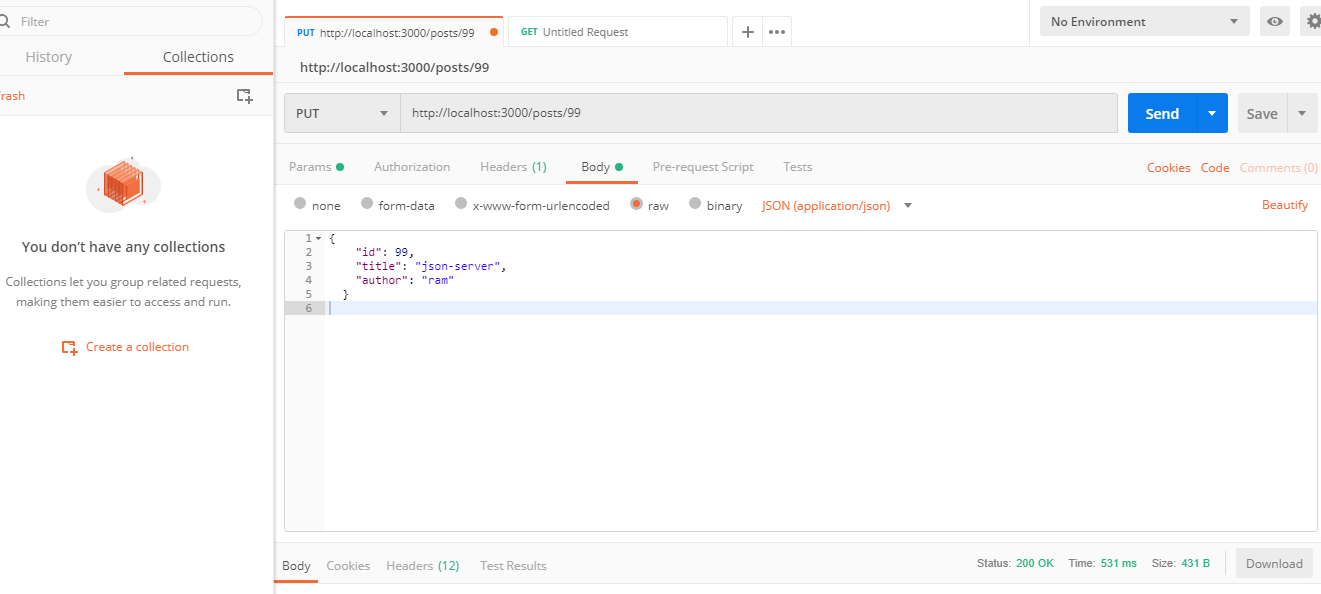
{

"id": 99,

"title": "json-server",

"author": "ram"

}



**In browser**



**Assertions in SOAP UI:**

* Assertion means act of affirming or stating something. It can also be interpreted as check point or a validation point.
* Once a request is sent to a web server a response is received. We need to validate if the response contains the data that we expect. In order to validate the response, we need to use assertions.

**Types of Assertions:**

1. Property Content
2. Compliance Status Standard
3. Script
4. SLA
5. JMS
6. Security

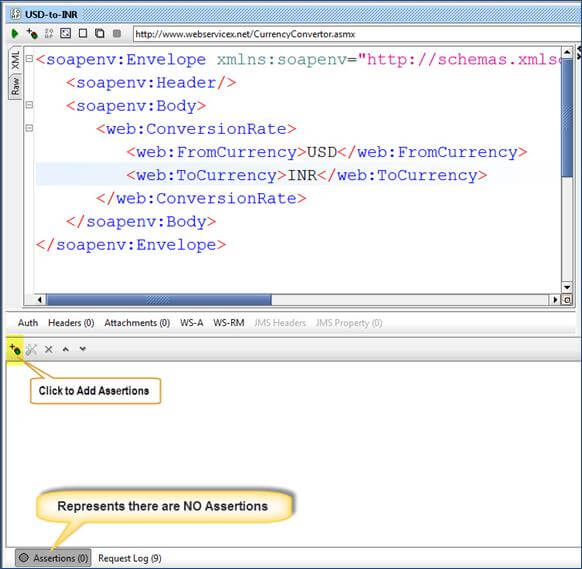
**CONTAINS ASSERTION**

Searches for the existence of the specified string. It also supports regular expression.

We will continue with the same example from the previous tutorial.

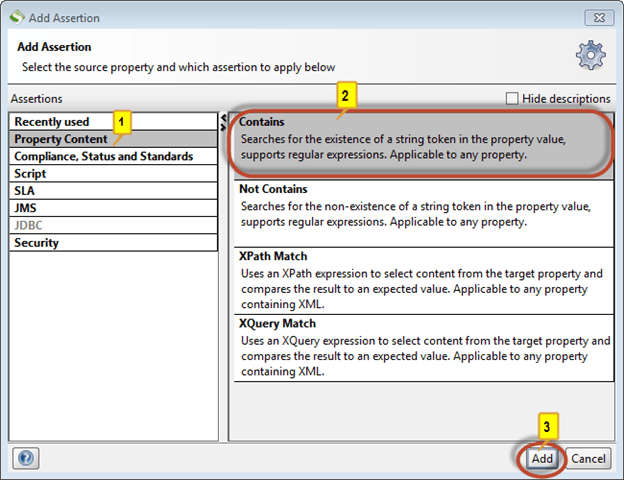
**Step 1:**By Default there are no assertions.

* + The Number of Assertions are shown in the Assertions Tab.
  + To add a new assertion, click on 'Add New Assertion' button.



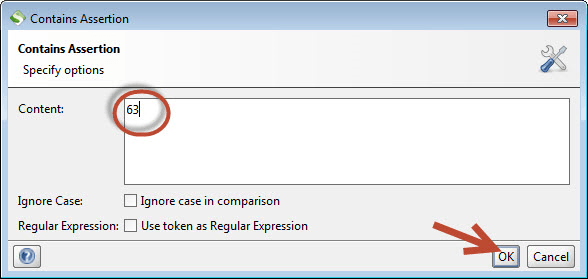
**Step 2:**Now,

* 1. Select the Assertion Category.
  2. Select the Assertion Type.
  3. Click 'Add'

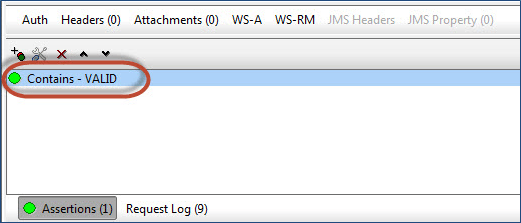


**Step 3:**Let us validate if the string '63' exist in the response. Click 'OK'

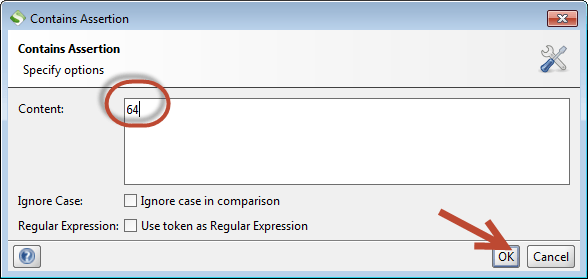
Note : We can also ignore case and add regular expression.



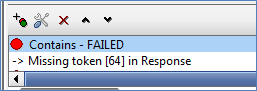
**Step 4:**Upon adding it, immediately assertion is executed and shows if VALID or INVALID.



**Step 5:**Now Let us say we change the content of 'Contains Assertion' to '64' and see what happens.



**Step 6:**The Assertion is executed and the result is thrown to the user. Since we don't have the string "64" within the response, the assertion has failed.

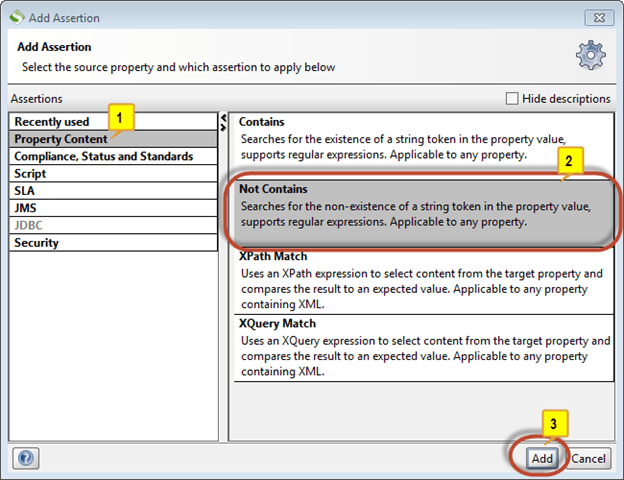


**NOT CONTAINS ASSERTION**

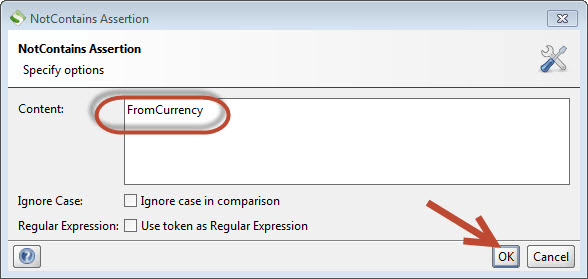
Searches for the Non-existence of the specified string. It also supports regular expression.

**Step 1:**Now after clicking on 'add new assertions' button,

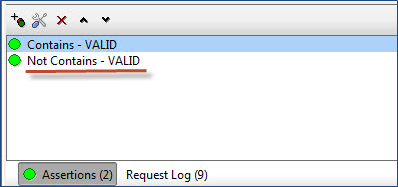
1. Select the Assertion Category.
2. Select the Assertion Type – In this case 'NOT Contains'
3. Click 'Add'



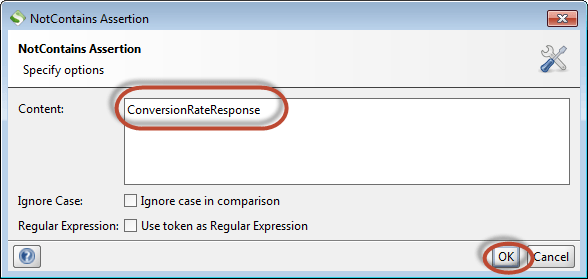
**Step 2:**Let us validate if the string 'FromCurrency' exist in the response. Enter the string 'FromCurrency' and Click 'OK'



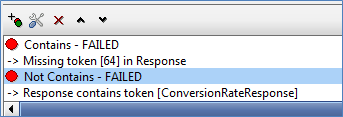
**Step 3:**As soon as an assertion is added, it executes and displays the result. So far we have added two assertions hence both the assertions are executed and displayed the result.



**Step 4:**Now let us change the contents of the 'Not Contains Assertion' and see what happens. We will check for the non-existence of the string "ConversionRateResponse".



**Step 5 :**The string 'ConversionRateResponse' is actually present in the response, hence the 'NOT Contains' assertion will fail as shown below.

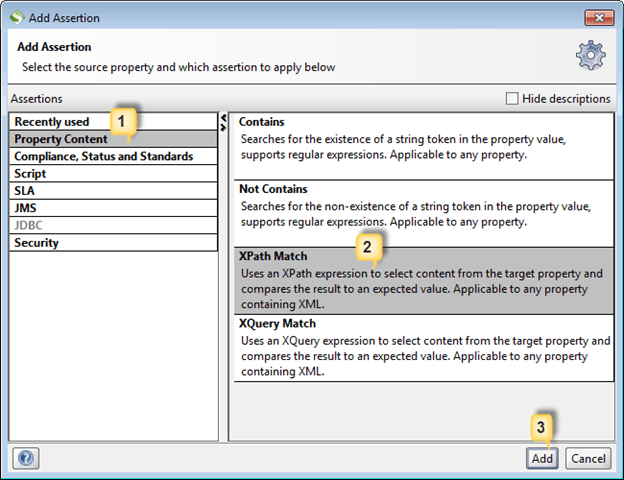


**XPATH MATCH ASSERTION**

Uses[XPath](https://www.guru99.com/xpath-selenium.html)expression to select the target node and its values. XPath, is an XML query language for selecting nodes from an XML document.

**Step 1:**Now after clicking on 'Add New Assertions' button,

1. Select the Assertion Category.
2. Select the Assertion Type – In this case 'XPath Match'
3. Click 'Add'



**Step 2:**Add XPath Window opens.

* + Before Adding XPath, we need to declare the NameSpace. An XML namespace is a collection of names, identified by a Uniform Resource Identifier (URI) reference, which are used in XML documents as element and attribute names.The same is used in SOAP UI XPath Assertion.

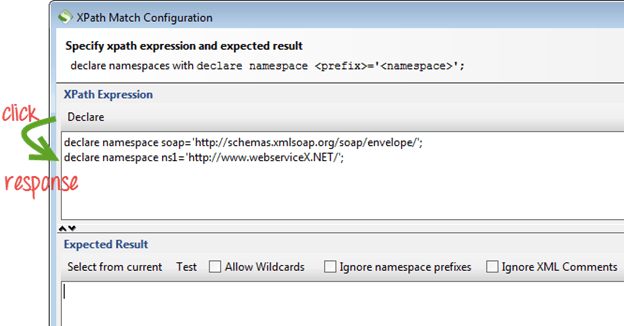
* + For declaring XML Namespace, we just need to click on 'Declare' button which would do the job for us else we can also manually declare a namespace ourselves.

* + After declaring the namespace we need to refer the XPath using the created name space.

* + Upon clicking the 'Declare' button, two namespaces will pop up as we have two URI's. One of them is the schema URL and the other one corresponds to the actual web service URL. We need to use the actual namespace where the web service is located and NOT the schema namespace while referencing XPath.

**declare namespace soap='http://schemas.xmlsoap.org/soap/envelope/';**

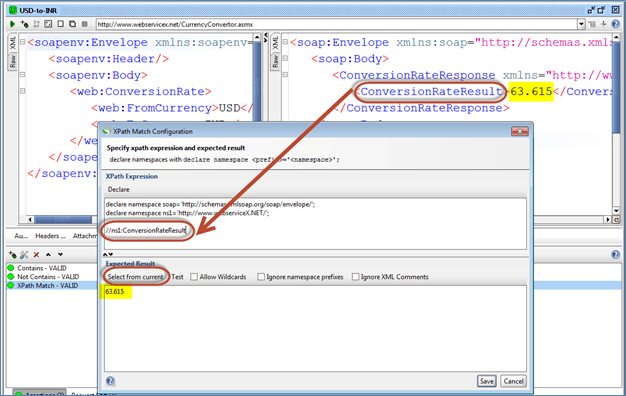
**declare namespace ns1='http://www.webserviceX.NET/';**



**Step 3:**Nowweneed to enter the XPath of the XML node that we need to validate.

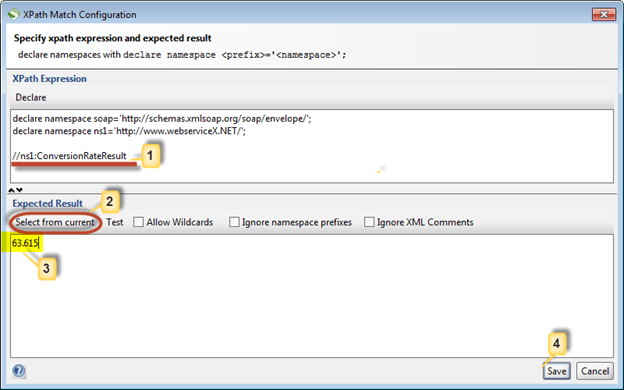
**//ns1:ConversionRateResult**Gives us the Value of the node enclosed between **<ConversionRateResult> & </ConversionRateResult>** and ns1 corresponds to the declared namespace which is pointing to **'http://www.webserviceX.NET/'**

After entering the XML, we need to click on 'Select from current' so that value from the current response would be picked up for comparison going forward.

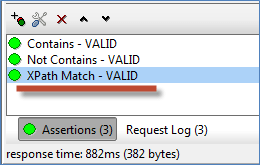


**Step 4:**So far,

* + After declaring the namespaces, we have entered the XPath of XML node that we need to Validate.
  + We Need to click 'Select from Current' to make the current value as the expected value.
  + The current value is shown to the user which we can modify if required.
  + Click 'Save'.



**Step 5:**The added Assertion will be displayed as shown below.



**Comparison Between SOAP and REST:**

|  |  |  |
| --- | --- | --- |
|  | **SOAP** | **REST** |
| Meaning | Simple Object Access Protocol | Representational State Transfer |
| Design | Standardized protocol with pre-defined rules  to follow. | Architectural style with loose guidelines and  recommendations. |  |
| Approach | Function-driven (data available as services,  e.g.: “getUser”) | Data-driven (data available as resources,  e.g. “user”). |  |
| Statefulness | Stateless by default, but it’s possible to make  a SOAP API stateful. | Stateless (no server-side sessions). |  |
| Caching | API calls cannot be cached. | API calls can be cached. |  |
| Security | WS-Security with SSL support. Built-in ACID  compliance. | Supports HTTPS and SSL. |  |
| Performance | Requires more bandwidth and computing power. | Requires fewer resources. |  |
| Message format | Only XML. | Plain text, HTML, XML, JSON, YAML, and others. |  |
| Transfer protocol(s) | HTTP, SMTP, UDP, and others. | Only HTTP |  |
| Recommended for | Enterprise apps, high-security apps, distributed environment,  financial services, payment gateways, telecommunication services. | Public APIs for web services, mobile services,  social networks. |  |
| Advantages | High security, standardized, extensibility. | Scalability, better performance, browser- friendliness, flexibility. |  |
| Disadvantages | Poorer performance, more complexity,  less flexibility. | Less security, not suitable for distributed  environments. |  |