Katalon Studio For Automated Testing

Sesi 9

API TESTING

How to do API Testing with Katalon

API Testing is one of the hottest trends of Software Testing in recent years and still keeps growing up till now. Instead of being handled solely by developers, testing API is presently a standard practice among many outsourcing teams. This short article will give you a comprehensive tutorial on how to perform ***REST API & WebServices Testing with Katalon Studio***. Subsequently, in this article REST API & WebServices Testing

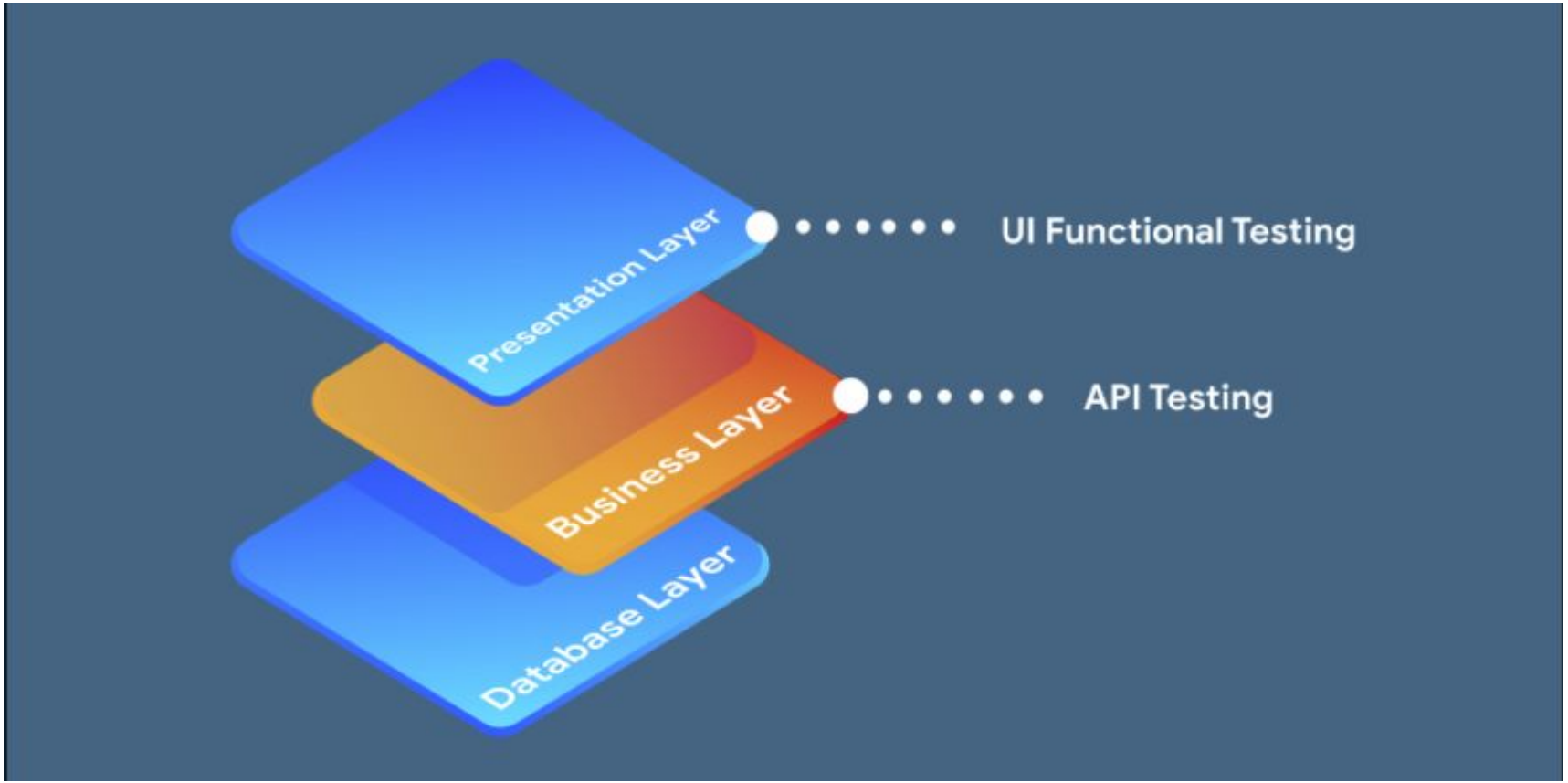
What is API testing?

API testing is a type of software testing that involves testing application programming interfaces (APIs) directly and as part of integration testing to determine if they meet expectations for functionality, reliability, performance, and security.

Where is API testing performed ?

Commonly, applications have three separate layers or tiers including Presentation Layer or user interface, Business Layer or application user interface for business logic processing, and Database Layer for modeling and manipulating data.
API Testing is performed at the most critical layer, the Business Layer, where business logic processing is carried out, and all transactions between User Interface and Database happen.

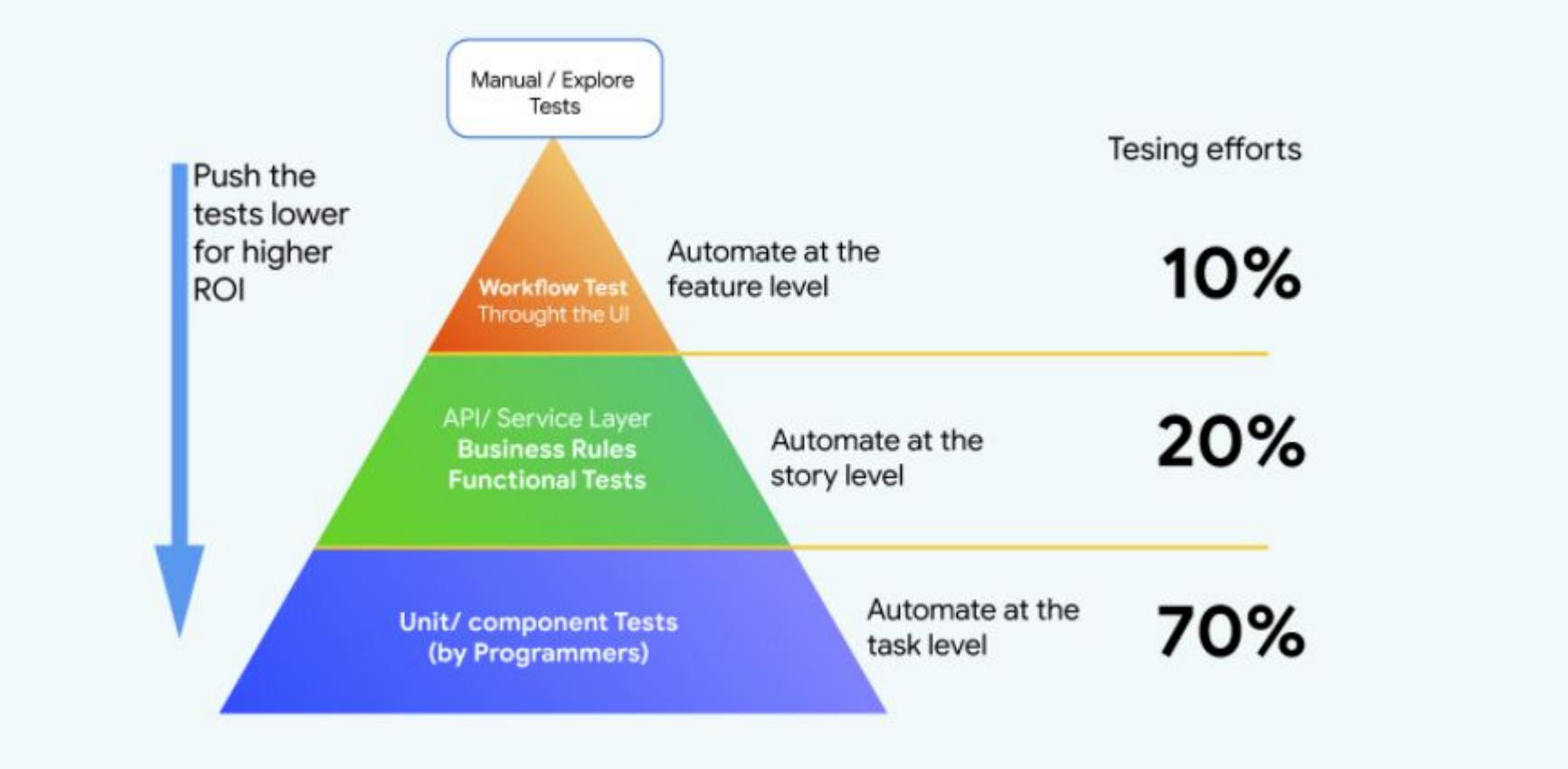
How to do API Testing with Katalon



How to do API Testing with Katalon

Commonly, applications have three separate layers or tiers including Presentation Layer or user interface, Business Layer or application user interface for business logic processing, and Database Layer for modeling and manipulating data.
API Testing is performed at the most critical layer, the Business Layer, where business logic processing is carried out, and all transactions between User Interface and Database happen.

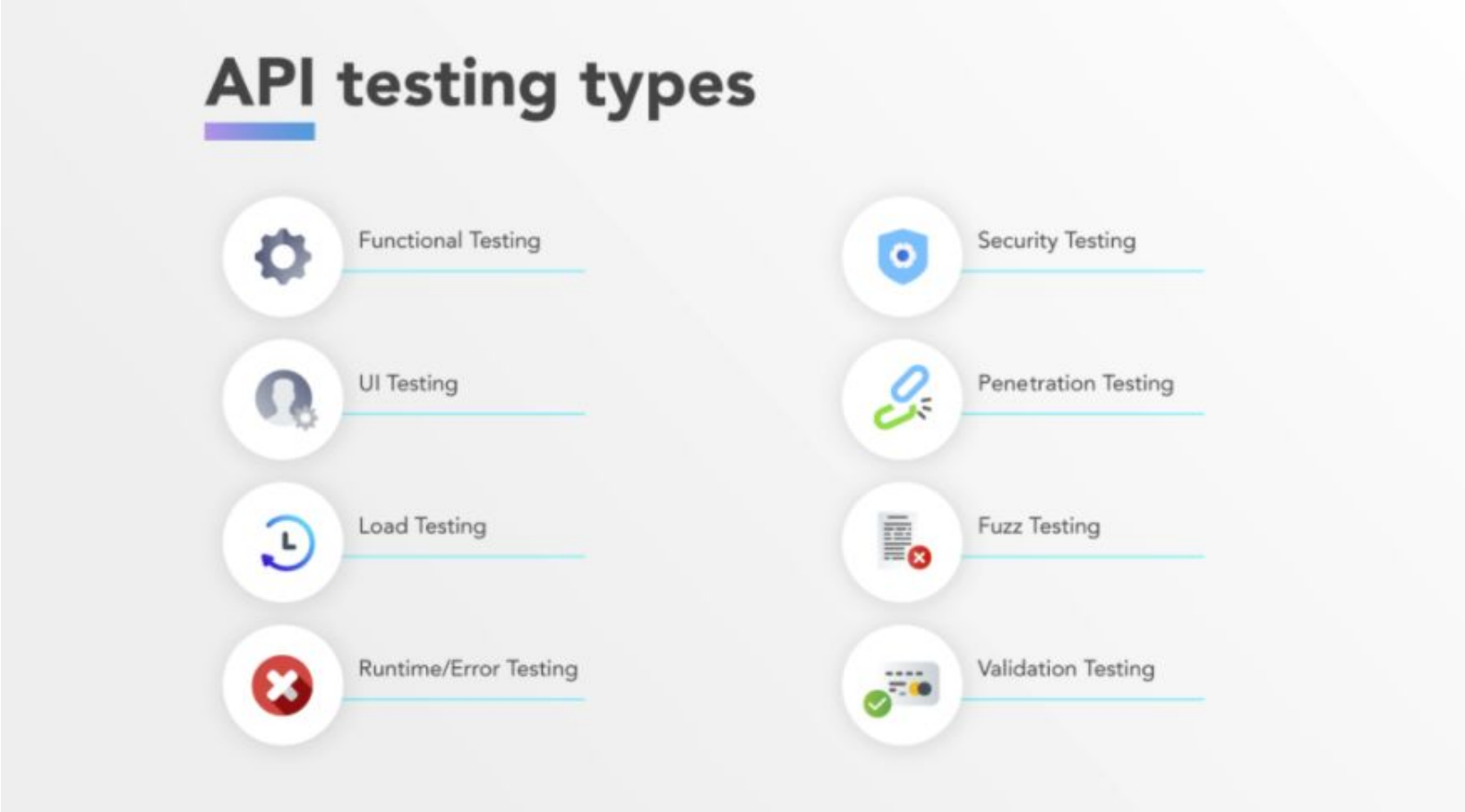
How to do API Testing with Katalon



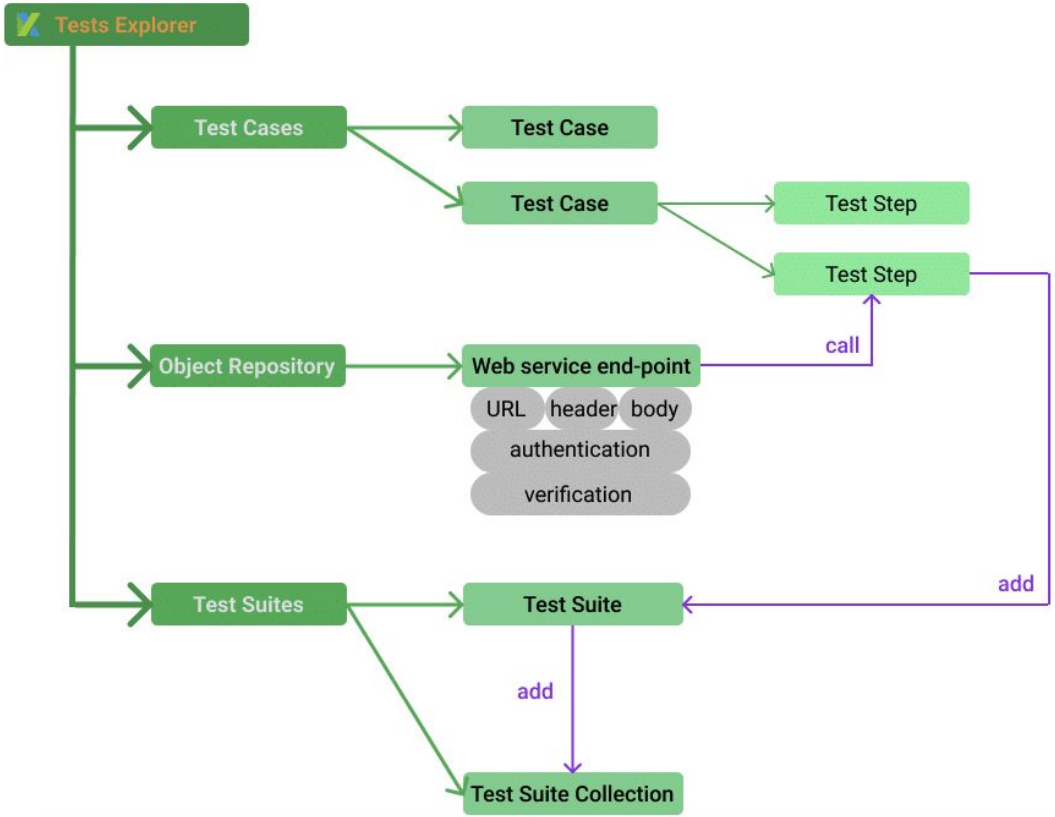
How to do API Testing with Katalon

The figure below shows three different layers of testing called the test pyramid initially coined by Mike Cohn in his book Succeeding with Agile. It has layers representing different types of testing. Despite its being overly simplistic, it offers us a general rule of thumb: it suggests how much testing we should focus on at each layer. As such, API and services tests in the second layer is an important testing activity that we should focus on.

API TESTING TYPES



API AUTOMATION TEST



API AUTOMATION TEST

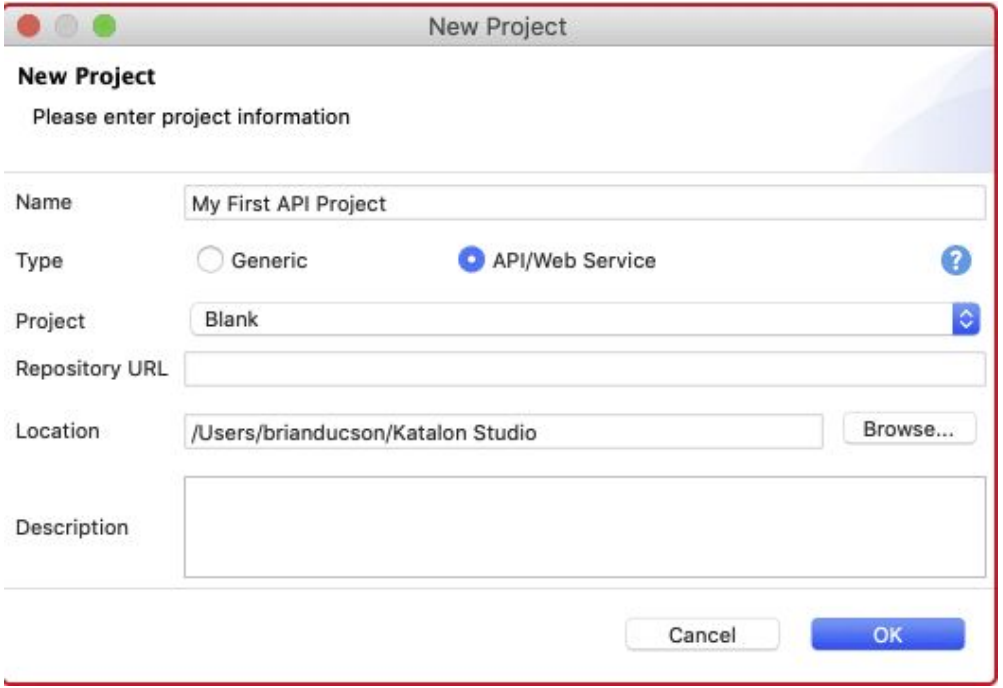
Before the creation of your first API test, let's have a look at the format used to set up a testing project.

- ***Object Repository:*** *Object Repository is a place that stores all Web service endpoints along with all information of request method, URL, header, content, and authentication. A folder system for better management integrates web service test objects in Object Repository.*
- ***Test Cases:*** Test Cases stores/keep all test scenarios and is categorized by a folder system. Each test case includes some steps that illustrate a test scenario. You can execute a test case individually with a specified execution profile.
- ***Test Suites:*** *Test Suites is where all test suites are stored. A test suite is a collection of test cases that verify a specific target. Test cases at the 'test suite' level can execute with a data-driven approach. The Test reports also generate at the 'test suite' level.*
- ***Test Suite Collection:*** *Test Suite Collection is a collection of Test Suites that verifies/ confirms a larger target. Test Suites at 'Test Suite' Collection level has specific Test environments specified.*

API AUTOMATION TEST

Step 1: Create a new project

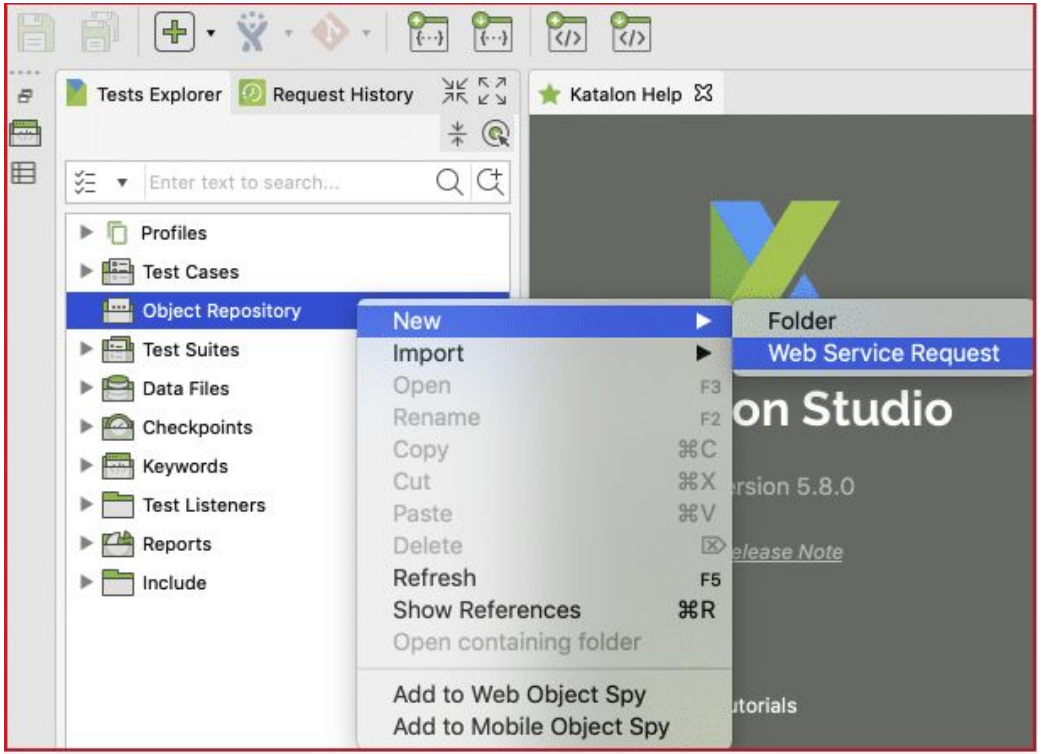
Go to ***File → New → Project***. Enter a project name along with its location to start a new project.



API AUTOMATION TEST

Step 2: Create a new RESTful endpoint at Object Repository

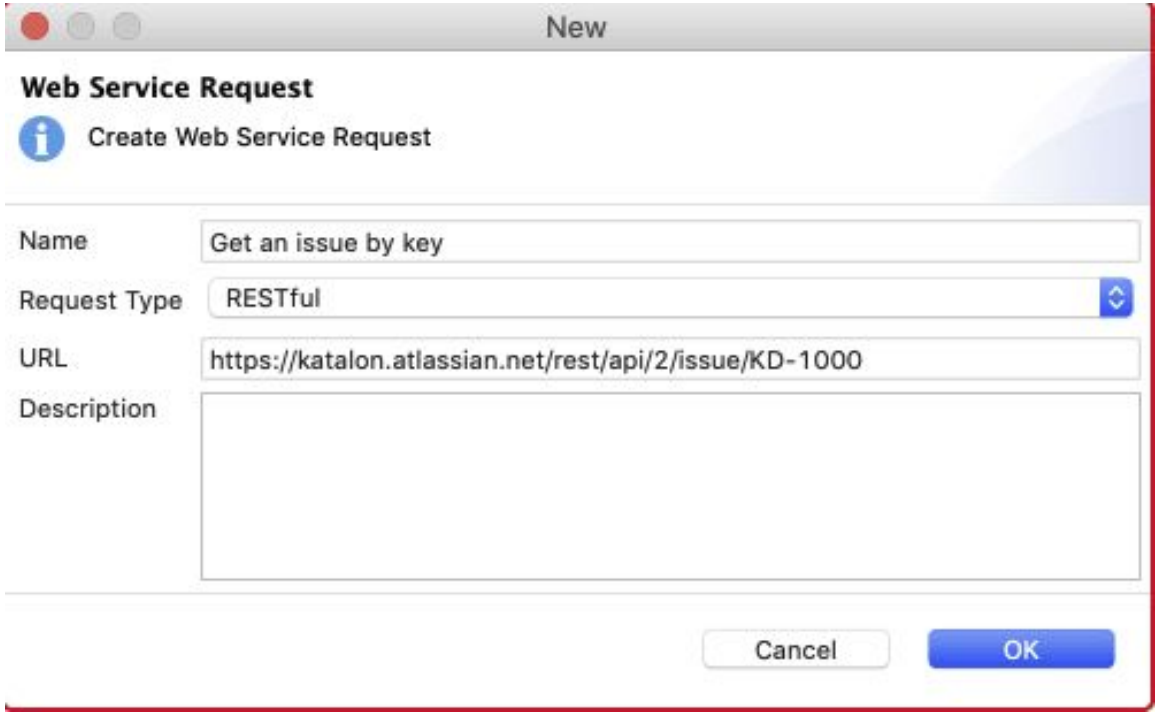
Object Repository → New → Web Service Request



API AUTOMATION TEST

Step 2: Create a new RESTful endpoint at Object Repository

Object Repository → New → Web Service Request



API AUTOMATION TEST

Katalon Studio stores Web service endpoints for testing at Object Repository, which is similar to Test Object in UI Test. At the "*Create New Web Service Request*" dialog, you can either choose to create a RESTful or a SOAP request.

"*Request type*" is a required field. You need to specify it at this very step. In contrast, "*URL*" is not required. You can set this value later in the next step.

API AUTOMATION TEST

Click, “*OK*.” Then we are ready to enter more details to the first *RESTful* test.

Katalon Studio - My First API Project - [Location: /Users/brianducson/Katalon Studio/My First API Project]

default | Keyword | Debug | Forum Search | Community Chat

Tests Explorer | Request History

Enter text to search...

- Profiles
- Test Cases
- Object Repository
  - GET Get an issue by key
- Test Suites
- Data Files
- Checkpoints
- Keywords
- Test Listeners
- Reports
- Include

Katalon Help | GET Get an issue by key | Help

GET (1) https://katalon.atlassian.net/rest/api/2/issue/KD-1000 (2)

Query Parameters

Add | Remove

Name	Value

(3) Authorization | HTTP Header | HTTP Body | (4) Verification | (5) Variables

Type: No Authorization

Update to HTTP Header

Response

Status: Elapsed: Size:

Body | Header

pretty | raw | preview

1

(6) JSON | XML | HTML

API AUTOMATION TEST

There are some critical concepts needed to specify when testing a *RESTful* request:

- ***Request method:*** *You can choose one of these following methods for your first request test: GET, POST, PUT, or DELETE. The method needs to match the URL to have a valid request. For instance, let's assume that your first test is a public API from the Jira Cloud version. In this case, you should choose/ select the GET method to receive information on an existing ticket using its ID.*
- ***Request URL:*** *Along with the request method, "request URL" tells the web server which API to utilize under test. Any mismatch between method and URL will lead to invalid request exceptions at runtime or wrong data response.*
- ***Authorization:*** *Authorization is a vital part of an API. It gets the correct data under permission (unless the data is public). Katalon Studio supports standard authentication methods:*

Basic*: The basic method requires username and password. Don't forget to click 'Update to HTTP Header' so that the authentication can apply to 'HTTP Header.'*

API AUTOMATION TEST

Authorization | HTTP Header | HTTP Body | Verification | Variables

Type: Basic

Username: brian.ducson@gmail.com

Password: ••••••••••

☐ Show Password

Update to HTTP Header

Authorization | HTTP Header | HTTP Body | Verification | Variables

Add Remove

Name	Value
Authorization	Basic YnJpYW4uZHVjc29uQGdtYWlsLmNvbToxMjM0NTY3ODkw

API AUTOMATION TEST

- ***Verification:*** *Verification is where you define assertion to ascertain that the response will contain the expected information.*

Katalon Help | GET *Get an issue by key

GET https://katalon.atlassian.net/rest/api/2/issue/KD-1000

Query Parameters

Authorization | HTTP Header | HTTP Body | Verification | Variables

```
import static org.assertj.core.api.Assertions.*

RequestObject request = WSResponseManager.getInstance().getCurrentRequest()

ResponseObject response = WSResponseManager.getInstance().getCurrentResponse()

// Verify the response
WS.verifyResponseStatusCode(response, 200)

WS.verifyElementPropertyValue(response, 'fields.project.key', 'KD')

WS.verifyElementPropertyValue(response, 'fields.issuetype.name', 'Task')

WS.verifyElementPropertyValue(response, 'fields.priority.name', 'Medium')

WS.verifyElementPropertyValue(response, 'fields.summary', 'REST - Create new RESTf

WS.verifyElementPropertyValue(response, 'fields.description', 'As a Katalon user,
```

Test Request And Verify

Hide snippets

Test scripts are written in Groovy and only run after response is received.

SNIPPETS

- Get current response
- Get current request
- Get a global variable
- Get a variable
- Response body: Contains string
- Response body: Convert to JSON Object
- Response body: Is equal to a String
- Response body: JSON value check
- Response headers: Content-Type header

Response — Elapsed: 2429 ms — Size: 5 KB

Body | Header | Verification Log

pretty | raw | preview


```
{
  "expand":"renderedFields,names,schema,oper
  "id":"10999",
  "self":"https://katalon.atlassian.net/rest
  "key":"KD-1000",
  "fields":{
    "issuetype":{
      "self":"https://katalon.atlassian.net/
      "id":"10002",
      "description":"A task that needs to be
      "iconUrl":"https://katalon.atlassian.n
      "name":"Task",
      "subtask":false,
      "avatarId":10318
    },
    "timespent":null,
    "project":{
      "self":"https://katalon.atlassian.net/
```


JSON | XML | HTML | JavaScript | Wrap Line

API AUTOMATION TEST

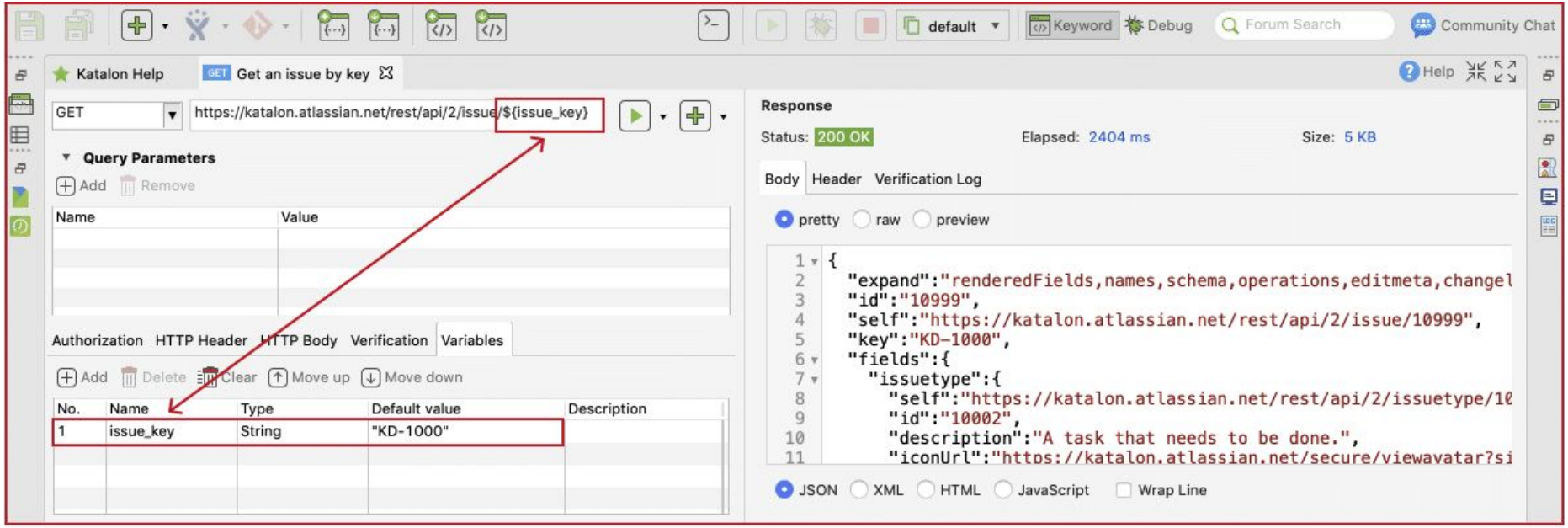
The verification tab of a request is similar to the Script tab of a test case. In other words, you can write custom scripts with built-in keywords or Groovy/Java scripts to check the response data. Besides built-in keywords, Katalon Studio also supports built-in snippets, which help you generate assertions with a single click. This feature is useful for testers who might find it challenging to deal with parsing or to assert with JSON data format.

The right panel of the request consists of the responses automatically displayed in a friendly format, and the verification results in Verification Log. To include verification script when sending the request, you need to choose the 'Test Request and Verify' option from the execution button.

The Verification script gives you quick feedback on the request status rather than an actual test. You can add more assertions at the 'test case' level in the next step.

API AUTOMATION TEST

- ***Variables:*** *Variables make API testing more strong and dynamic with the data-driven approach. In Katalon Studio, every part of the request can parameterize. In other words, we can use dynamic data for URL, authentication, HTTP Header, and HTTP Body to maximize the capability of data-driven testing. Let's have a look at this example:*



- ***Formatter:*** *The response will automatically display in a proper format: JSON, XML, HTML, and JavaScript. It gives you a quick view of the response status.*

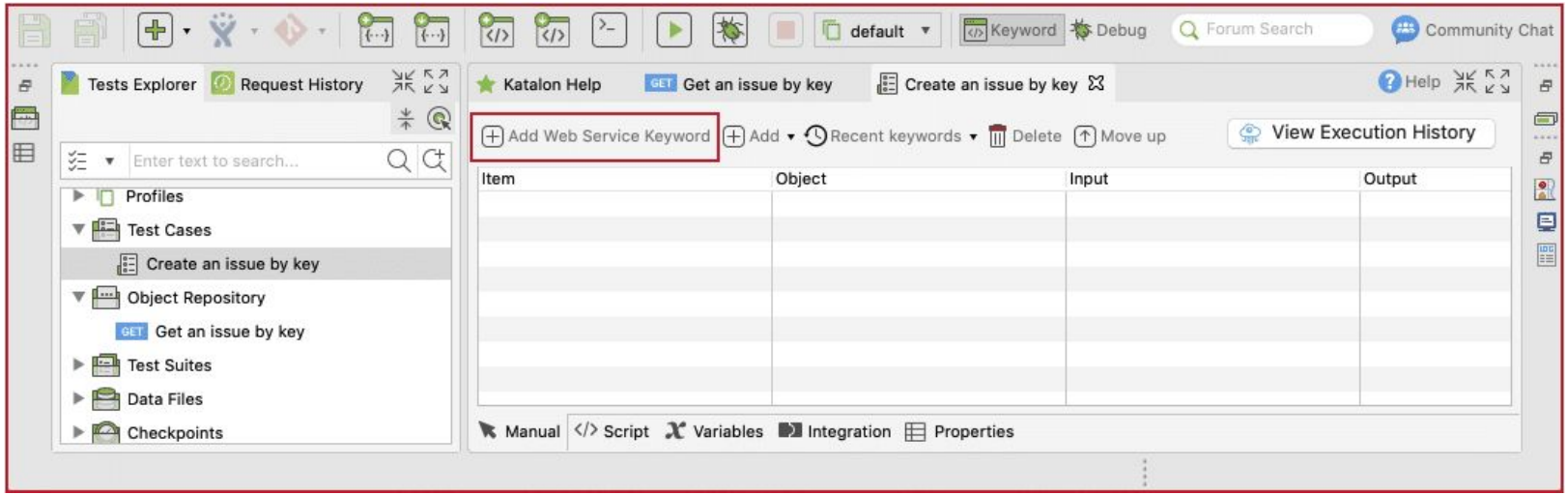
API AUTOMATION TEST

Step 3: Create a new test case with an existing request.

While the request at **Object Repository** is helpful for fast testing, you can add the request verification at the test case level for better managing and reporting.

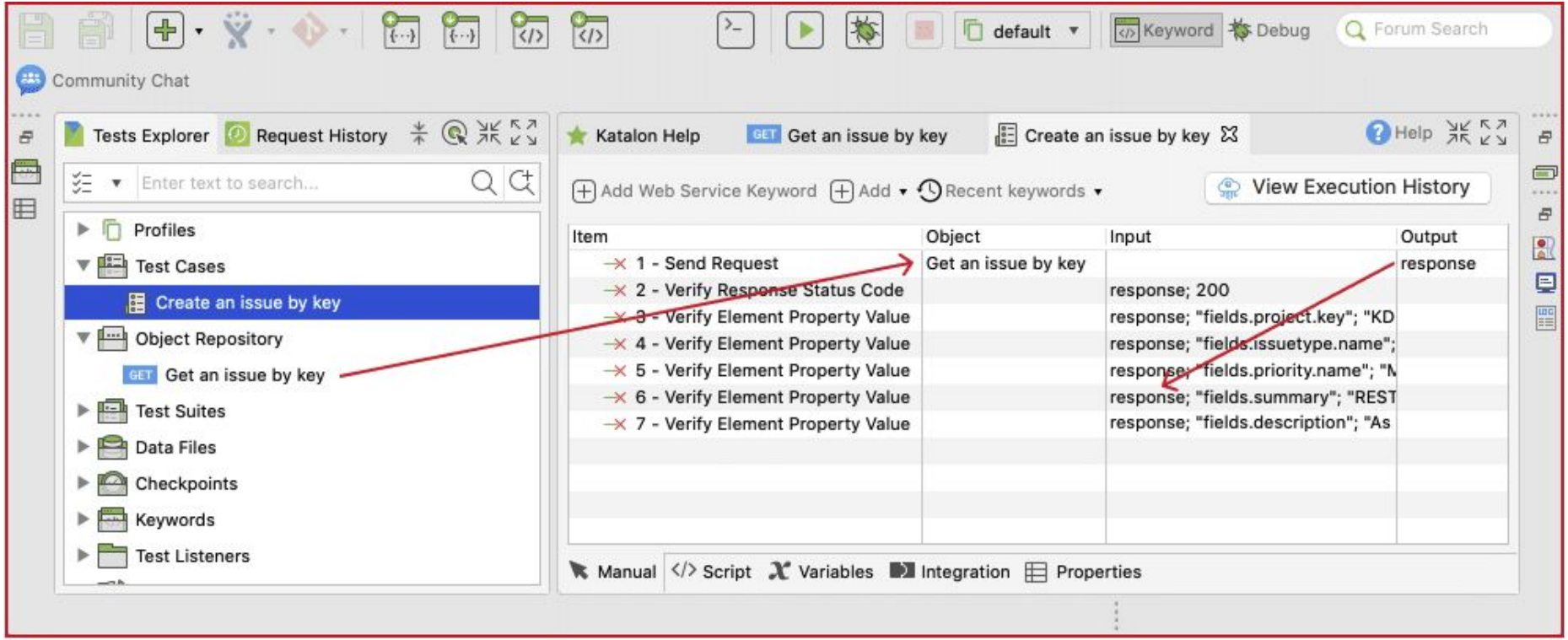
Step 4: Add an existing request to a test case

A request can insert into a test case with Web service built-in keywords. We can use many keywords to send the request, to verify the response, and to make the request as part of a more significant testing flow.



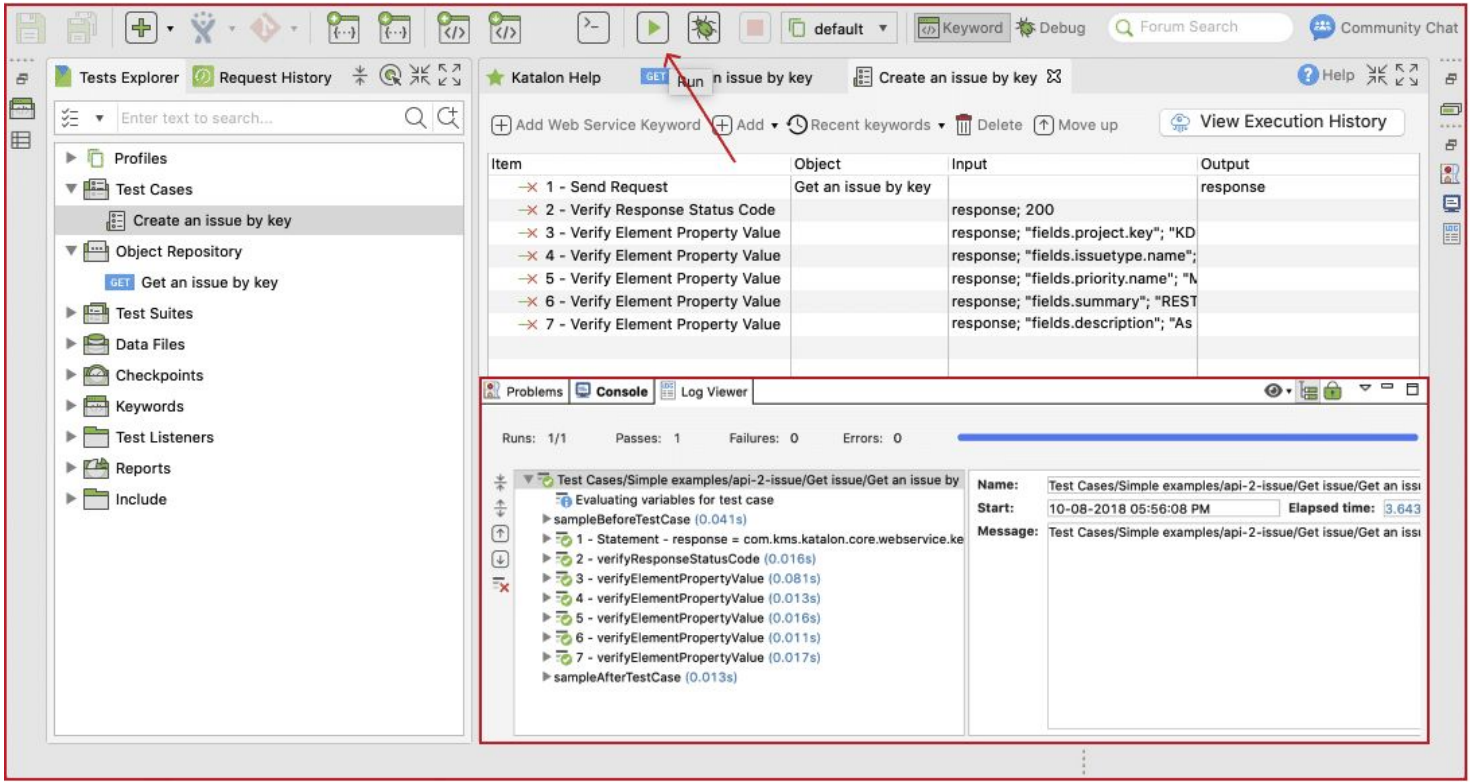
API AUTOMATION TEST

Following test case shows how we can call the request with verification steps from a test case:



API AUTOMATION TEST

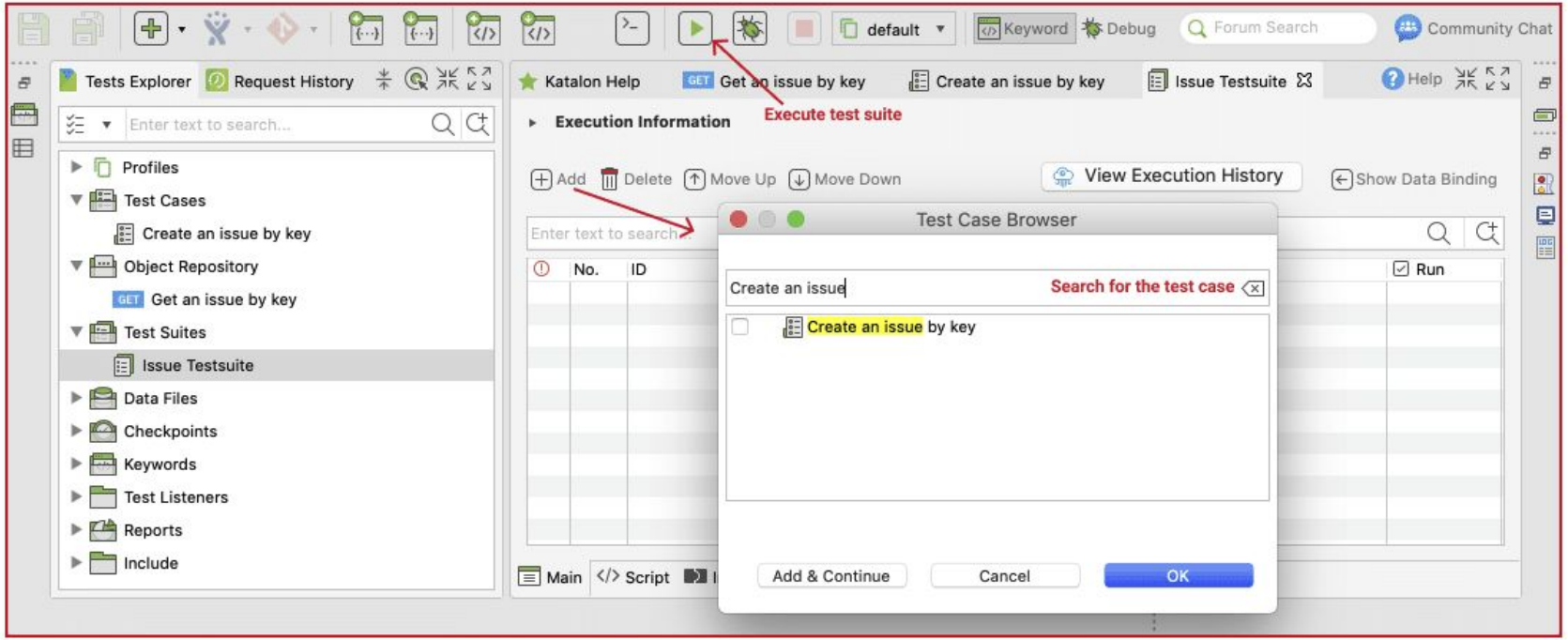
The test case can execute as a standard test case in Katalon Studio. We can view each verification step from the log.



API AUTOMATION TEST

Step 5: Add a test case to the test suite

A test case can add to a test suite via either the drag-and-drop feature or the "*Add test case*" tool. Once the test case adds to the test suite, we can execute the entire test suite with the Run button (*without selecting the browser as in UI testing*)



REST API

1. Buka Web Reqres.in

GET	LIST USERS
GET	SINGLE USER
GET	SINGLE USER NOT FOUND
GET	LIST <RESOURCE>
GET	SINGLE <RESOURCE>
GET	SINGLE <RESOURCE> NOT FOUND
POST	CREATE
PUT	UPDATE

Request
/api/users?page=2

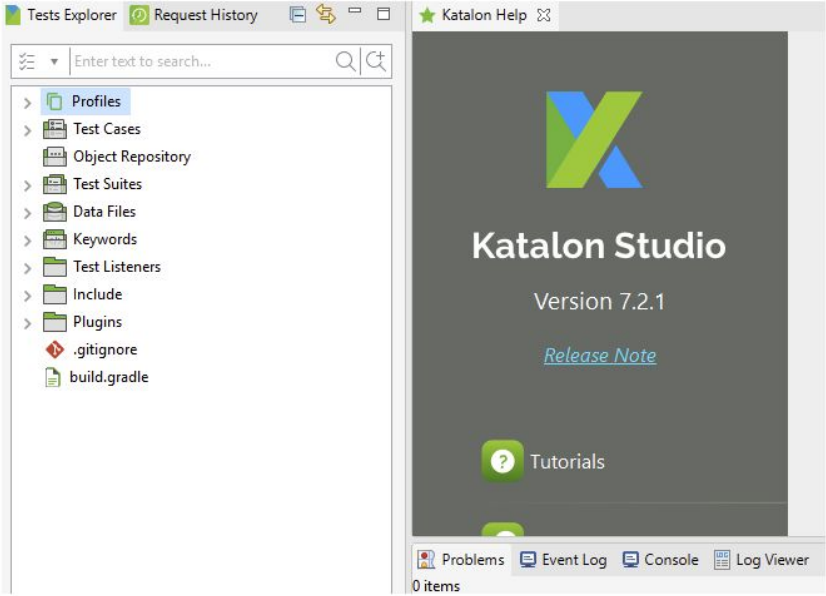
Response
200


```
{
    "page": 2,
    "per_page": 6,
    "total": 12,
    "total_pages": 2,
    "data": [
        {
            "id": 7,
            "email": "michael.lawson@reqres.
            "first_name": "Michael",
            "last_name": "Lawson",
            "avatar": "https://s3.amazonaws.
        },
        {
            "id": 8,
            "email": "lindsay.ferguson@reqre
            "first_name": "Lindsay",
            "last_name": "Ferguson",
```

REST API

Pilih menu FILE > NEW > Project untuk membuat project baru. Lalu akan muncul kotak dialog yang perlu diisi sebagai data project tersebut:

- Name: isikan nama project (bebas)
- Type: isikan API/Web Service
- Lalu klik OK. Tunggu hingga loading selesai. Maka akan terlihat susunan test explorer yang terdiri dari profiles, test cases, object repository, test suites, dan lain-lain.



REST API

Konfigurasi Request

Link URL: https://reqres.in/api/users?page=2
Method: GET

GET	LIST USERS
GET	SINGLE USER
GET	SINGLE USER NOT FOUND
GET	LIST <RESOURCE>
GET	SINGLE <RESOURCE>
GET	SINGLE <RESOURCE> NOT FOUND
POST	CREATE
PUT	UPDATE

Request
/api/users?page=2

Response
200


```
{
    "page": 2,
    "per_page": 6,
    "total": 12,
    "total_pages": 2,
    "data": [
        {
            "id": 7,
            "email": "michael.lawson@reqres.
            "first_name": "Michael",
            "last_name": "Lawson",
            "avatar": "https://s3.amazonaws.
        },
        {
            "id": 8,
            "email": "lindsay.ferguson@reqre
            "first_name": "Lindsay",
            "last_name": "Ferguson",
            "avatar": "https://s3.amazonaws.
```

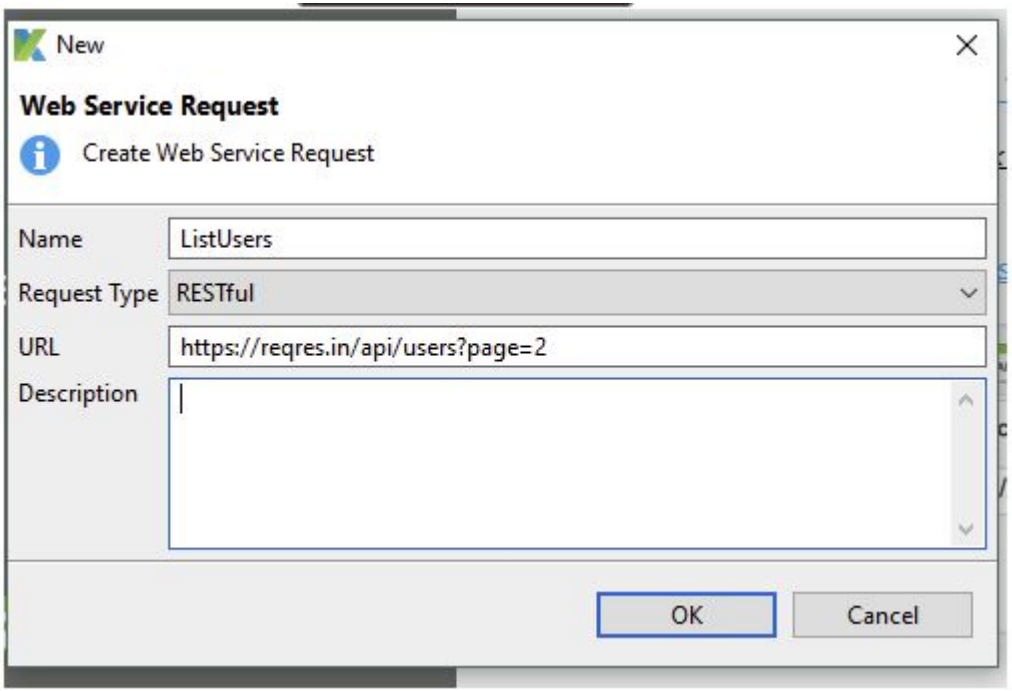
REST API

Langkah berikutnya adalah konfigurasi request. Pertama kita perlu menyusun dulu bagaimana foldering di Katalon agar rapi. Maka pada Object Repository > New > Folder. Beri nama folder tersebut REST Service.

Endpoint yang bakal kita pakai adalah pada website https://reqres.in/ di bagian List User. URL untuk mendapatkan response tersebut menjadi,

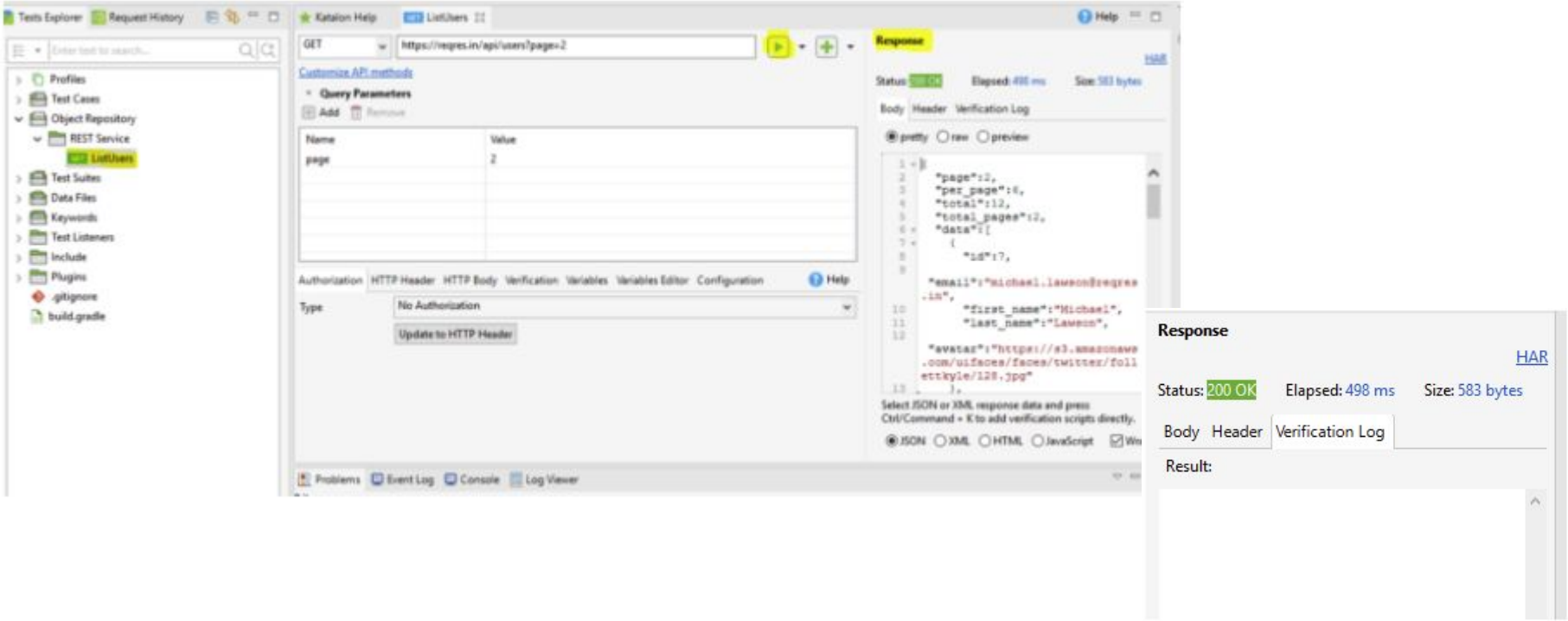
REST API

Selanjutnya, pada folder **REST Service** tadi, ***klik kanan > NEW > Web Service Request***



REST API

- Setelah itu, klik OK dan akan terbentuk object untuk ListUsers.
- Untuk mengirim request, klik tombol test request (tombol play).
- Tunggu beberapa saat dan response akan segera terlihat



Verification log masih kosong

REST API

Mengapa verification log masih kosong? Berarti tidak ada validasi dong?

Iya, karena dia hanya nembak request dan terima respon.

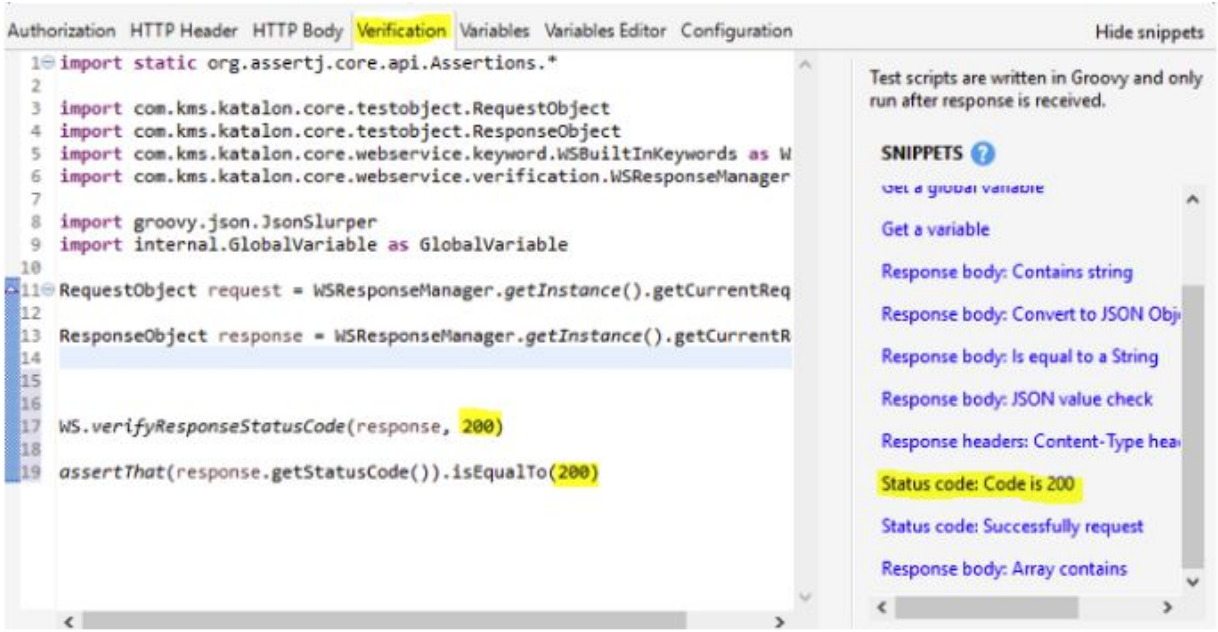
Selain itu tadi kita juga gak menjalankan **test request and verify** karena belum ada langkah verifikasi yang dilakukan. Solusinya ?

memberikan ***verification*** agar respon yang didapatkan dicek status atau isinya. **Pertama,** kita pastikan respon HTTP yang diterima adalah **HTTP 200** yang berarti sukses.

Caranya ??

REST API

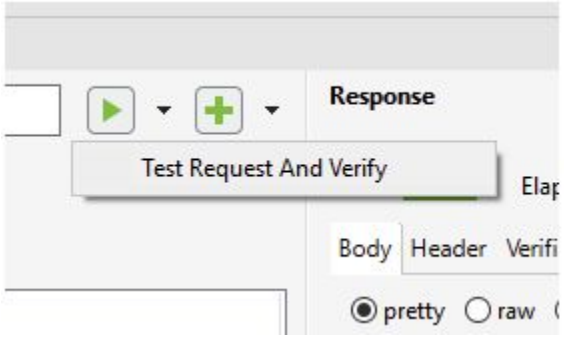
a. klik bar ***verification***. Pilih snippet: ***"Status code: Code is 200"***. Seharusnya ada code baru yang digenerate oleh Katalon secara otomatis. Status code ini juga bisa diganti tergantung kebutuhan user ya.



status code verification di Katalon

REST API

b. Karena kali ini kita menambahkan verification, langkahnya beda. Klik ikon panak kebawah disebelah test request. Kemudian buka verification log lagi.



test request and verify untuk menembak request dan melakukan verifikasi dari script yang dibuat

REST API

Katalon melakukan verifikasi dari snippet HTTP 200 yang sudah kita tambahkan.

Hasil dari verifikasi diatas adalah PASSED yang artinya Katalon sudah menerima respon dan melakukan verifikasi apakah HTTP response nya 200.

Selanjutnya kita akan melakukan verifikasi mengenai isi dari response JSON yang diterima.

REST API


```
{
   "page": 2,
   "per_page": 6,
   "total": 12,
   "total_pages": 2,
   "data": [
      {
         "id": 7,
         "email": "michael.lawson@reqres.in",
         "first_name": "Michael",
         "last_name": "Lawson",
         "avatar":
"https://s3.amazonaws.com/uifaces/faces/twitter/follettkyle/128.jpg"
      },
      {
         "id": 8,
         "email": "lindsay.ferguson@reqres.in",
         "first_name": "Lindsay",
         "last_name": "Ferguson",
         "avatar":
"https://s3.amazonaws.com/uifaces/faces/twitter/araa3185/128.jpg"
      },
      {
         "id": 9,
         "email": "tobias.funke@reqres.in",
         "first_name": "Tobias",
         "last_name": "Funke",
         "avatar":
"https://s3.amazonaws.com/uifaces/faces/twitter/vivekprvr/128.jpg"
      },
      {
         "id": 10,
         "email": "byron.fields@reqres.in",
         "first_name": "Byron",
         "last_name": "Fields",
         "avatar":
"https://s3.amazonaws.com/uifaces/faces/twitter/russoedu/128.jpg"
      },
      {
         "id": 11,
         "email": "george.edwards@reqres.in",
         "first_name": "George",
         "last_name": "Edwards",
         "avatar":
"https://s3.amazonaws.com/uifaces/faces/twitter/mrmoiree/128.jpg"
      },
      {
         "id": 12,
         "email": "rachel.howell@reqres.in",
         "first_name": "Rachel",
         "last_name": "Howell",
         "avatar":
"https://s3.amazonaws.com/uifaces/faces/twitter/hebertialmeida/128.j
pg"
      }
   ],
   "ad": {
      "company": "StatusCode Weekly",
      "url": "http://statuscode.org/",
      "text": "A weekly newsletter focusing on software development,
infrastructure, the server, performance, and the stack end of
things."
   }
```

REST API

Sama seperti langkah sebelumnya, namun kali ini kita pakai script untuk **memvalidasi value dari JSON**.

Katalon sudah menyediakan snippet untuk membantu kita supaya tidak perlu menulis script dari awal. Pilih snippet: ***"Response Body: JSON value check"***.

```
WS.verifyElementPropertyValue(response,
'issues[0].fields.project.key', 'KTP')
```

Bagaimana cari path JSON nya?

- response: script ini mengambil value response yang sudah dideskripsikan sebelumnya.
- 'issues[0].fields.project.key': path yang akan divalidasi
- 'KTP': value apa yang seharusnya ada dalam path tersebut

REST API

CTRL + K (untuk windows).

Pertama, klik dulu pada bagian apa di response yang ingin di validasi.
Klik pada tulisan “Lindsay”.

REST API

Lakukan **CTRL + K** lagi
seharusnya akan ada script yang digenerate di bagian verification, seperti berikut:

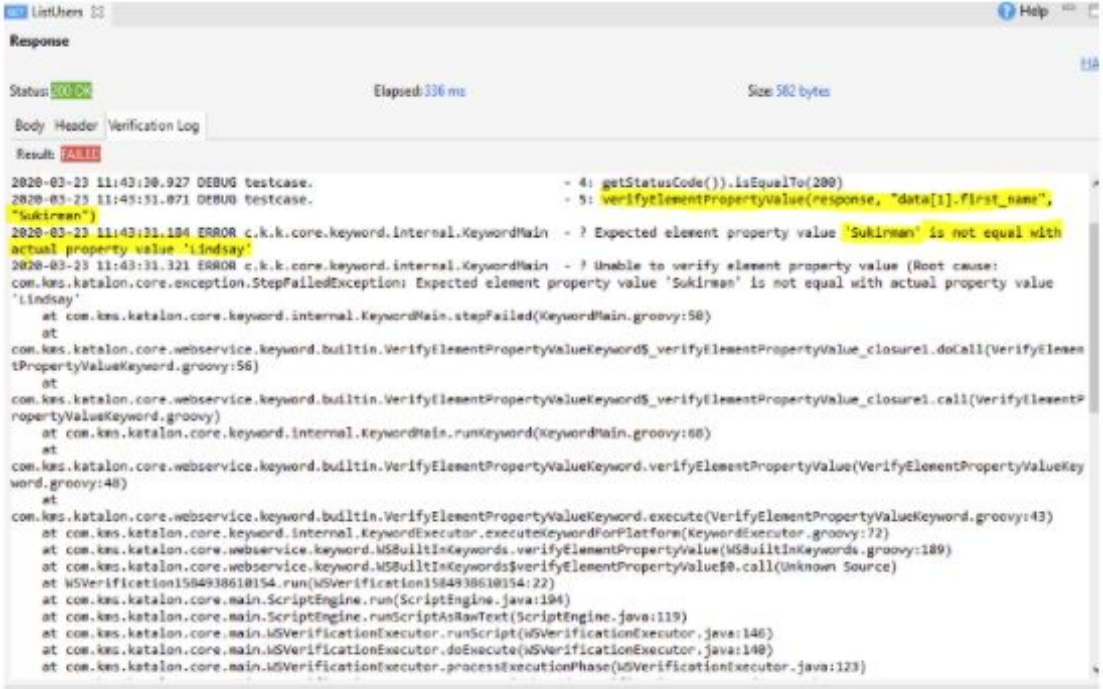
```
WS.verifyElementPropertyValue(response, 'data[1].first_name',
"Lindsay")
```

Mari kita cek di **verification log**nya:

Nah. Hasil dari verification tersebut adalah **PASSED**. Berarti sudah sama dengan apa yang kita harapkan.

CONCLUSION

Memang gimana kalo bukan PASSED? Oke. Kita kembali ke verification dan ganti Lindsay menjadi Amel. Maka setelah di test request and verify, hasilnya adalah:



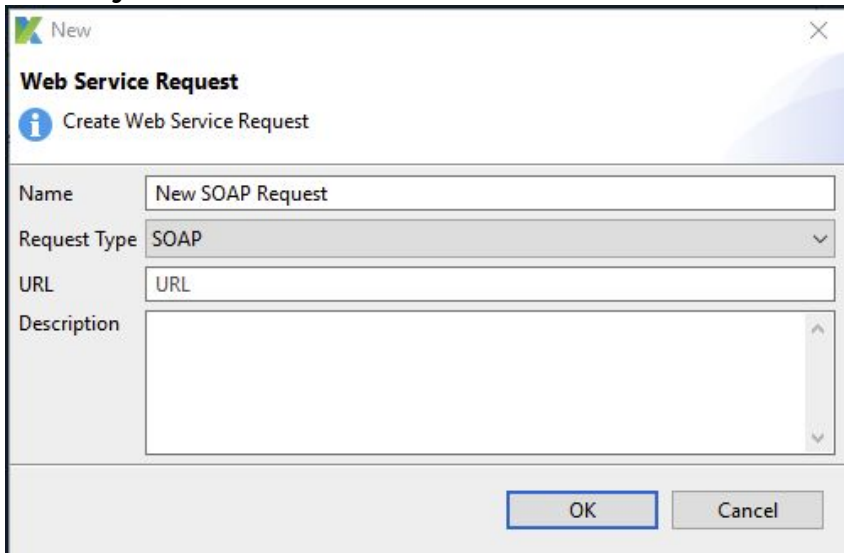
Hasilnya adalah **FAILED** karena valuenya tidak sama seperti yang diharapkan

SOAP API

When sending a SOAP Request in Katalon Studio, you can receive a response from the API server for examination and troubleshooting.

Creating a SOAP-based Request

1. From the main menu, select File > New > Web Service Request.
2. In the New Web Service Request dialog, select SOAP in the Request Type list and click OK to create a new SOAP object.

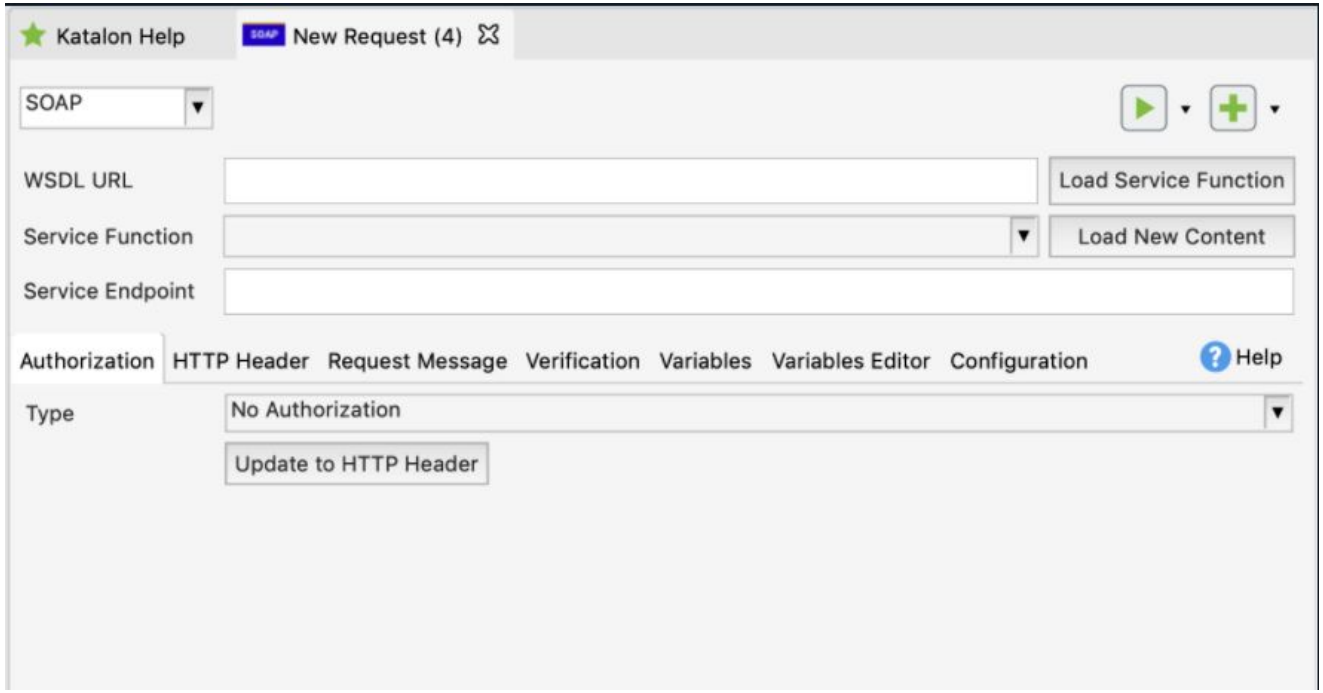


A New request object is created under the Object Repository of Katalon Studio

SOAP API

Adding SOAP Request Details

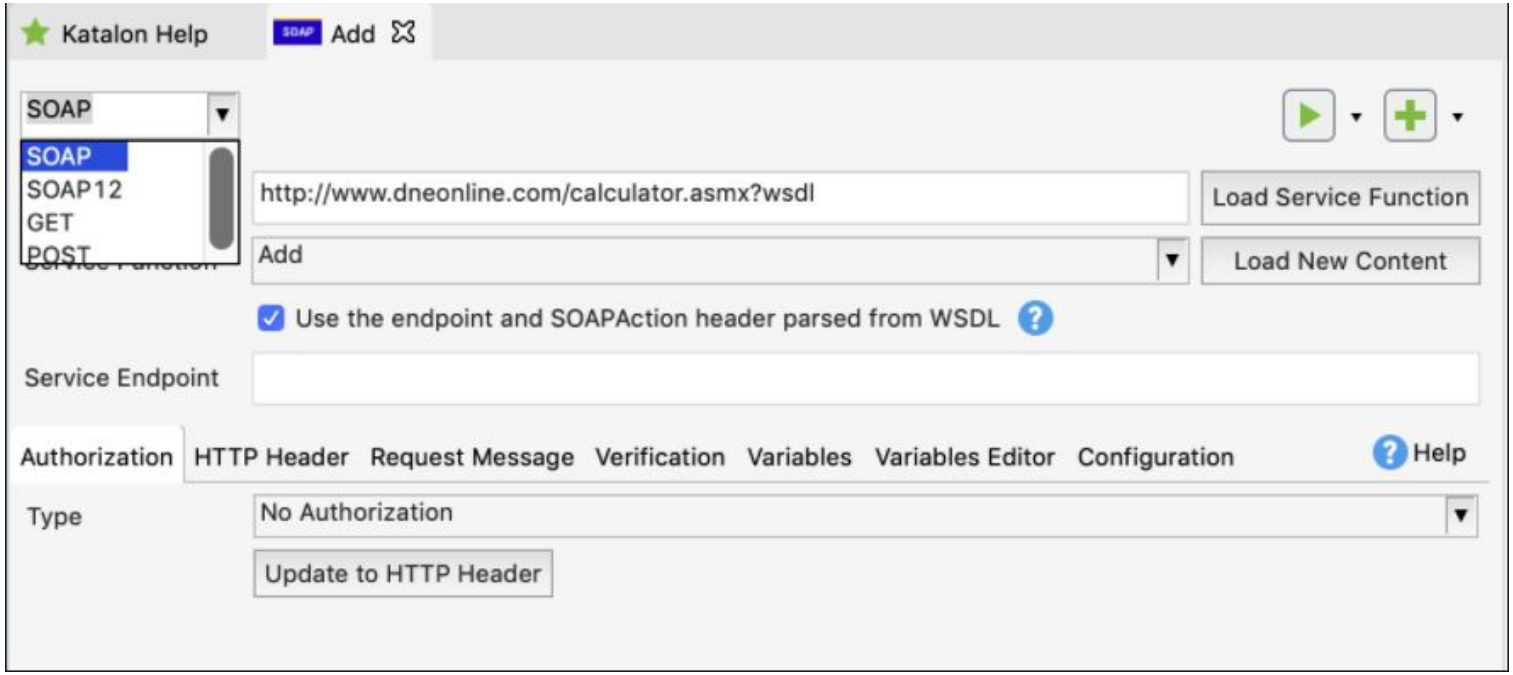
After you've created a request successfully, double-click on the request to open its editor for adding details. In the opened editor of the New Request object, you can see all the required information of a request object.



SOAP API

Request Method

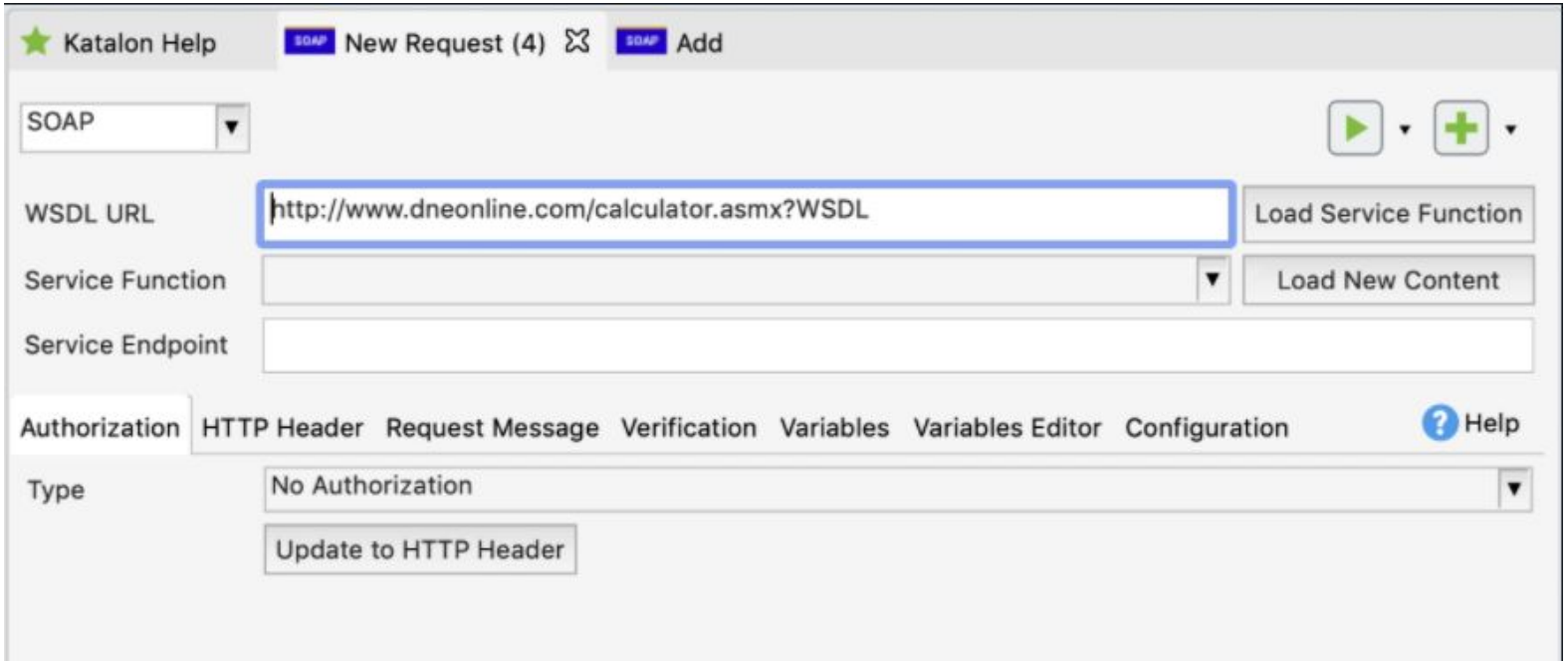
The request method indicates the expected action to be executed on the specified resource. Katalon Studio supports the following SOAP methods: SOAP, SOAP 1.2, POST, GET. By default, Katalon selects SOAP as a method for a new SOAP request.



SOAP API

WSDL URL

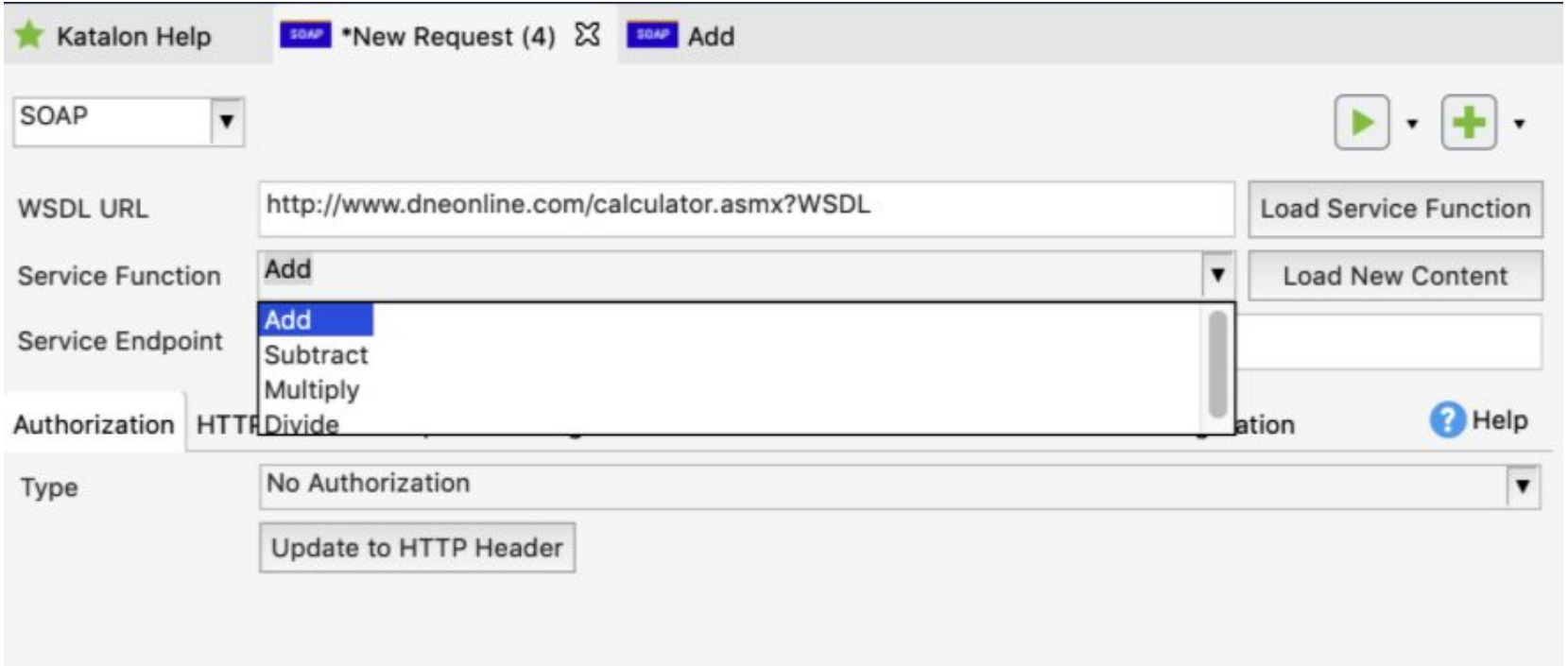
This field is for a WSDL path from which Katalon Studio imports the content to this SOAP request.



SOAP API

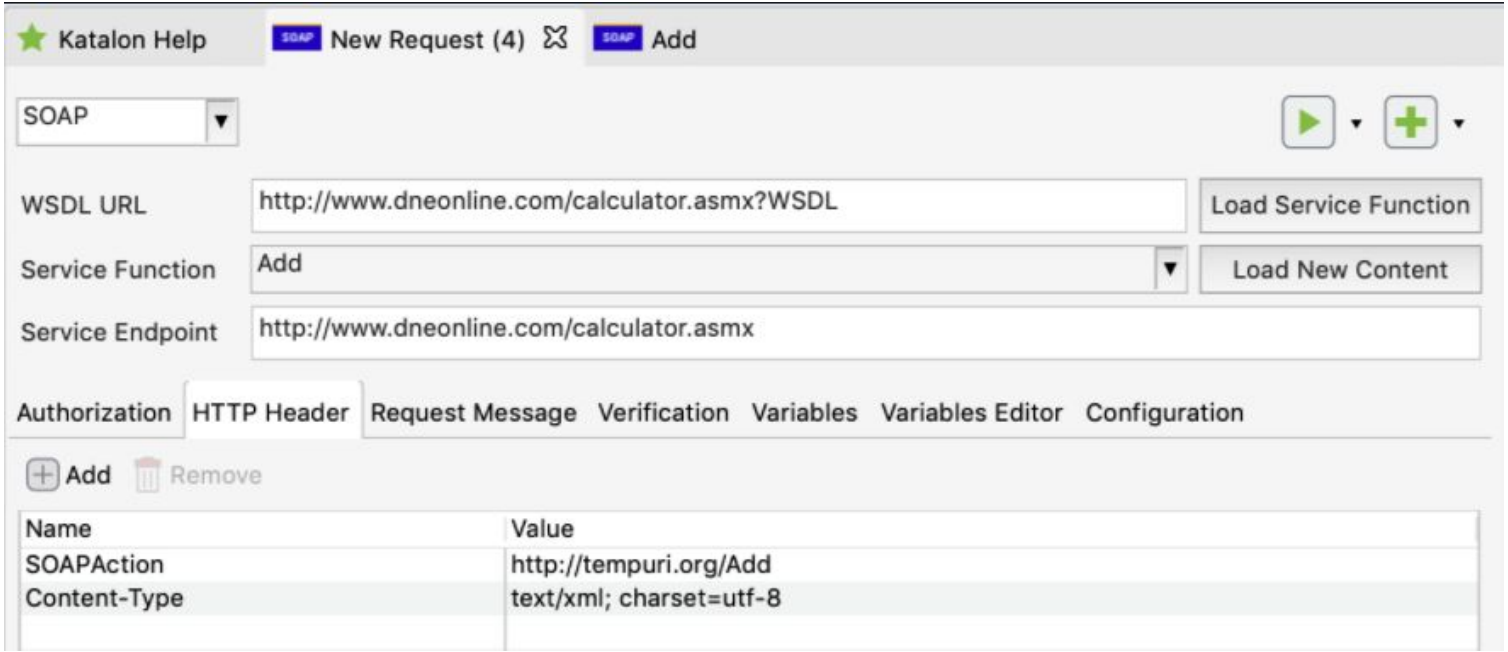
Service Function

The function that you want to use in this SOAP request. When clicking Load Service Function, you can retrieve a list of service functions available from the WSDL file.



SOAP API

Each Service Function carries its own content, including Service Endpoint, SOAPAction Header and Request message.
In Service Endpoint, You can specify another URL indicating the desired service endpoint of this request.

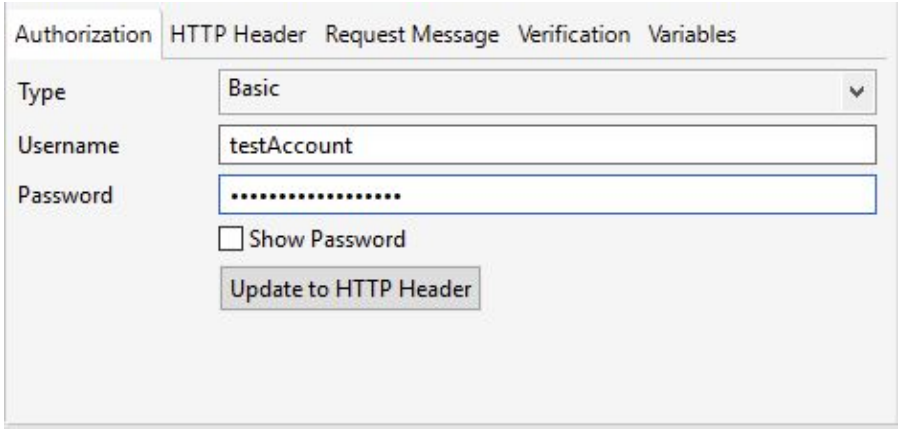


SOAP API

Request Authentication
This part is used for authenticating and authorizing the request, which means to verify if the client is permitted to send the request and to perform the endpoint operation.

For more details on using each type of auth, please see:

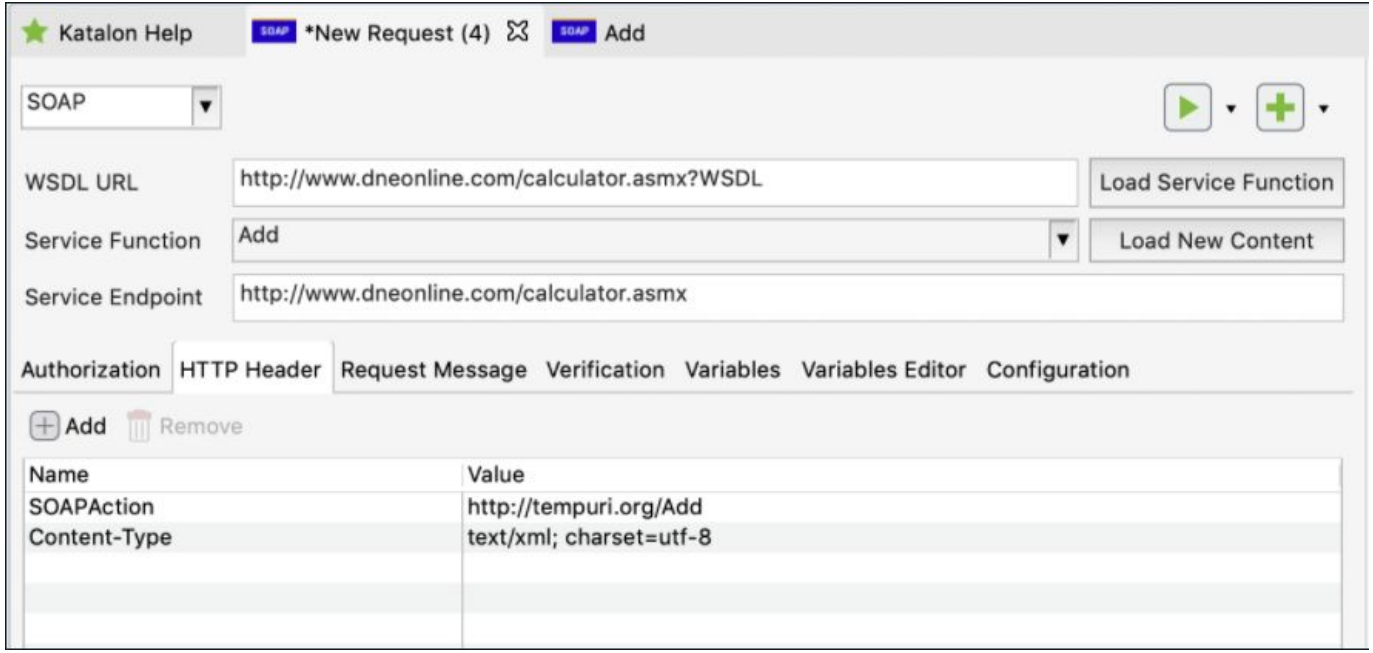
- Basic
- OAuth 1.0
- OAuth 2.0



SOAP API

Request Headers

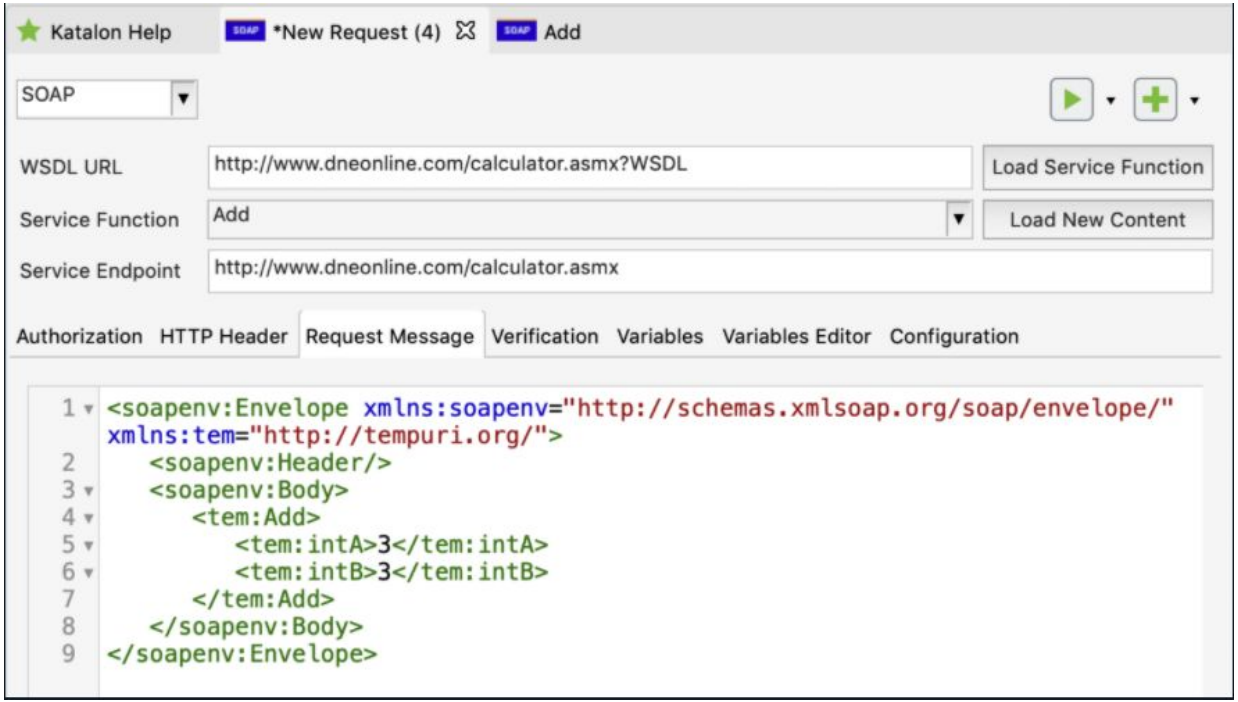
The header information needs sending along with this SOAP request. You can select headers from the list of suggested options (by double-clicking on the Name cell) or enter another header of your interest.



SOAP API

Request Message

The information that you want to transmit in this SOAP request. You can get it after clicking Load New Content of the selected service function.



SOAP API

After sending the service request, Katalon Studio retrieves a message from the server and displays it in the Response view of the request. A service response comprises Status, Elapsed time, and Size fields; Body section, Header, and Verification Log.

- Status: The status code of the response
- Elapsed: The total time that starts from the request is sent until Katalon Studio receives the last byte of the response
- Size: Size of the response package

SOAP API

Response Body
There are 2 viewing formats: pretty and raw. For example, the SOAP's response to http://www.dneonline.com/calculator.asmx?WSDL is shown below.

Response

HAR

Status: 200 OK Elapsed: 664 ms Size: 184 bytes

Body | Header | Verification Log

pretty raw

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
3    <soap:Body>
4      <AddResponse xmlns="http://tempuri.org/">
5        <AddResult>6
6        </AddResult>
7      </AddResponse>
8    </soap:Body>
9  </soap:Envelope>
10
```

SOAP API

Response

HAR

Status: 200 OK Elapsed: 664 ms Size: 184 bytes

Body | Header | Verification Log

pretty raw

```
1  <?xml version="1.0" encoding="utf-
   8"?><soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.o
   rg/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XM
   LSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XM
   LSchema"><soap:Body><AddResponse
   xmlns="http://tempuri.org/">
   <AddResult>6</AddResult>
   </AddResponse></soap:Body>
   </soap:Envelope>
```