# Development of test automation framework for REST API testing

**Article** *in* Journal of Computational and Theoretical Nanoscience · February 2019

DOI: 10.1166/jctn.2019.7749



**5 authors**, including:

Rajiv Vincent
VIT University Chennai
**12** PUBLICATIONS **4** CITATIONS
SEE PROFILE

Rajesh M
VIT University
**10** PUBLICATIONS **6** CITATIONS
SEE PROFILE

*Journal of*
*Computational and Theoretical Nanoscience*
Vol. 16, 453–457, 2019

# Development of Test Automation Framework for REST API Testing

S. Venkatraj[1], Rajiv Vincent[1,*], V. Vijayakumar[1], K. Vengatesan[2], and M. Rajesh[1]

[1] *School of Computing Science and Engineering, VIT University, Chennai 600127, Tamil Nadu, India*
[2] *The Department of Computer Engineering, Sanjivani College of Engineering, Kopargaon 423603, Maharashtra, India*

This work proposes a generalized excel based test automation framework for REST API testing using groovy script. Testing is an important process in software development process, which helps to find unknown errors, failure and bugs in a software application and increases quality of the software application. With the existing process of testing is manual, where the inputs are given manually and the actual outcome or output is validated for different test case scenarios. This manual testing leads to increase in time and effort of testing, as there will be different scenarios to be tested in which different test data has to be prepared to test a particular functionality. As the functionalities in an application changes or added up, over the software development phase, there is a need to test the old functionalities after and before, integrating the new or changed functionality to the existing system. This results in testing the already tested functionalities to test again manually, which increases effort and time for the testing process. The manual testing process has to be automated, so that the testing time, effort and money can be reduced. The test automation framework is implemented using a groovy compiling tool in which the functionalities such as, read the data from input test data excel file, make a REST API request, validate the data and writing the result to the excel file has been developed. Since groovy is a flavor of java, several libraries have been imported to implement the test automation framework.

**Keywords:** REST API Testing, Test Automation Framework, Manual Testing, JSON Response, API Test Automation Framework, API URL, Response Time, Actual Response Code.

RESEARCH ARTICLE

## 1. INTRODUCTION

The testing process is a vital process for any application before it is going to be deployed or delivered. The testing process is done manually, where a continuous availability of human effort is needed, to visually test the application by giving different inputs for different scenarios. The testers need to prepare test case document, prepare various test case scenarios and prepare input data. For particular a test, the expected and the actual outcome is compared and the results are entered in the document manually. The testers must test again and again after a modification in the application, which also increases time, effort and cost. Having this as a base, the concept of REST API testing to be known where a request to be sent and response is captured and it is verified for errors. Format of request and response is JSON. In REST API testing, the testing process is carried out by sending a request (input data) and observing the response returned from the API and validating it with the expected response. After validation the results are entered for a test. The process of sending request, validating response and observing result is done manually for each test, which eventually increases effort, time and cost. This approach of manual REST API testing becomes inefficient when a change occurs in the API. The testing process must be automated to increase efficiency and also to decrease effort, time and cost.

## 2. REVIEW OF LITERATURE AND RESEARCH GAP

API based testing requires a programming interface to test the application's behavior which is to be tested. The test scripts that are created separately are to be used by the automation framework to execute the validation as specified for a specific test. API automation yields 100% test coverage. Earlier detection of defects, effort put by the manual testers is decreased. The test automation can be scheduled as a nightly build which tests the application automatically if the numbers of test cases are more. The API based test automation framework should produce results at the end with detailed information about passed

*Author to whom correspondence should be addressed.

and failed test cases. The actual values are to be captured and given by the framework for final verification by the testers after the automation [1]. Automating the testing process increases the speed software development process. The API test automation framework developed should be generic as possible, where it should work for any application's developed REST APIs which can be reused for any project in the future, which helps an organization to put minimal effort on API testing [2].

This automated testing can have:

(1) Detailed test cases defined from the test case document and expected results, preconditions for functional validation.

A separate test environment is needed since the test cases may increase and repeating the execution of test cases may increase. Also another advantage of automating the REST API testing is the implemented framework can perform continuous regression testing.

### 2.1. Aim of the Research

The main aim of this work is to automate the integration testing process and reduce time and increase efficiency of testing.

## 3. MATERIALS/METHODS

In the proposed architecture as shown in Figure 1, the framework expects an excel file as an input with the file containing data that are used to do automated testing for validating REST API. The framework is designed in such a way that it exports results in a separate excel file after the automated is executed for a corresponding input file containing test data. The validation functions are created as a separate library file in which the framework calls the appropriate validation function, as mentioned in the input excel file.

## 4. REST API TEST AUTOMATION FRAMEWORK ESSENTIALS

### 4.1. Test Data

This module contains the test data that will be read by the framework to do test automation of each test case scenarios. The columns contained in the input test data excel file used for test automation are as follows:

- Test ID
- API URL
- Method
- Pre-Condition
- Input data
- Action
- Expected Result Description
- Expected Error Code
- Expected Response Code
- Actual Response
- Actual Error Code

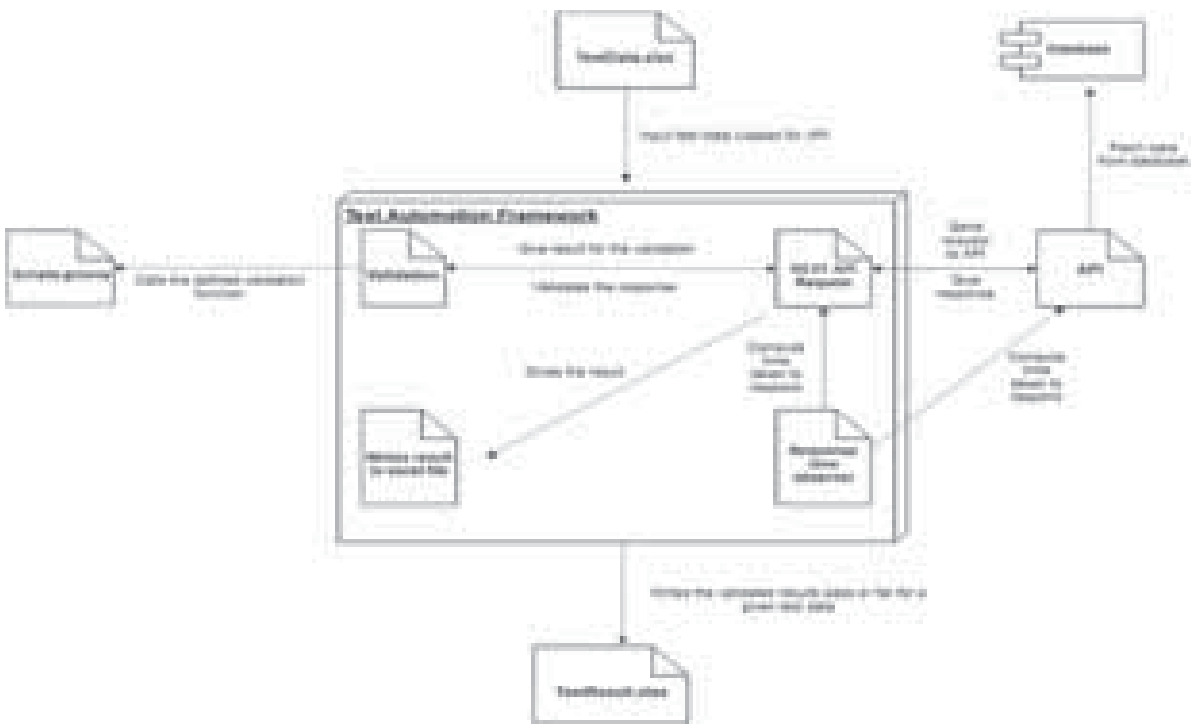

**Fig. 1.** REST API test automation framework architecture.



RESEARCH ARTICLE

- Actual Response Code
- Result.

*Test ID*:

The test id which is a unique identifier for each test case scenario referred from the test case document as per the standard is specified in this field.

*API URL*:

The API URL is used to make REST API request. Each API will be having a specific URL with optional query parameter as input.

*Method*:

An REST API has a specific type of call to perform an operation, the CRUD operations (Create, Read, Update and Delete). Based on the operation the method call is specified in this field. The possible HTTP methods are DELETE, PUT, POST and GET.

*Pre-Condition*:

The pre-condition that are to be satisfied before the test execution to be performed has to be defined in this field.

*Input Data*:

The request data to be sent for a specific API request has to be given in this field. The request data should be in JSON format.

*Action*:

The validation to be performed is given in this field, say "responsecodevalidaton."

*Expected Result Description*:

The description for the expected result, what has to be verified in the response for the test has to be defined in this field.

*Expected Error Code*:

The expected error code, the error code is verified with response, for negative scenario test cases has to be specified in this field.

*Expected Response Code*:

The expected response code for a specific API request, which has to be verified, is specified.

*Actual Response, Actual Error Code, Actual Response Code, Result*:

These fields are initially left empty, which will be filled by the framework to produce results for all the test cases contained in the test data file.

Each row in the excel is a test case which will be having values for all these above-mentioned fields, which serves as an input for the framework to automate each test case.

### 4.2. Framework

The test automation framework is developed using groovy script. The framework is responsible for getting the test data as input and automates the test case and produce test result file. In order to make a REST API request-API URL, method, and request data are needed. The framework takes these data from the input test data file as an input and makes the REST request and gets response and uses it for validation. In addition to that, the framework takes action, expected error code, expected response code data field from the input test data to do the verification in the response returned from that API.

The validation is done based on the value contained in the action column. The validation functions are defined in a separate groovy file. The framework does the validation based on the validation mentioned in the action column of the test data. The framework calls the defined validation function by sending the response returned from the REST API request and expected values from the input test data file, as a parameter. The validation function validates the parameters and returns whether "pass" or "fail" to the framework. The validation function validates by parsing the actual JSON response and compares it with expected value and returns the result, now the framework writes the actual response, actual error code, actual response code returned from the request made and the result returned from the called validation function. The framework creates a copy of the test data excel file and writes the actual response, actual error code and actual response code and the result that are returned.

The Table I shows the necessary inputs that will be read by the framework to do REST API test automation. The framework uses the API URL and method from the above table to make a REST API request. The request body for the particular REST request was taken from the test data field in the Table I. The action will give the type of validation to be done for a particular test. Similarly there will be '*n*' number of test cases in each row, having values for each field. The framework reads row by row and tests automatically till the end of input test data excel file.

### 4.3. Test Result

The test result is the replica of the test data where the left-out columns like actual response, actual error code, actual response code and results columns are filled by the framework after execution of test automation. The result whether passed or failed will be written by the framework for each row containing each test case of different scenarios. The actual response, actual error code and actual response code and result are also written in the test result file. The testers will observe the results written by the framework for each test case in the test result file. By this it helps the testers to observe the results and report the failed test cases to the development team to fix the error or the detected bug.

The Figure 2 shows the step by step process flow of the framework. For each test, the operations in the flow diagram will be carried out and the result will be recorded or

**Table I.** Sample input data for the framework.

| API Url | Method | Test Data | Action |
|---|---|---|---|
| http://apiendpoint/login | POST | { Username: "john", password:"john" } | error code and response code validation |

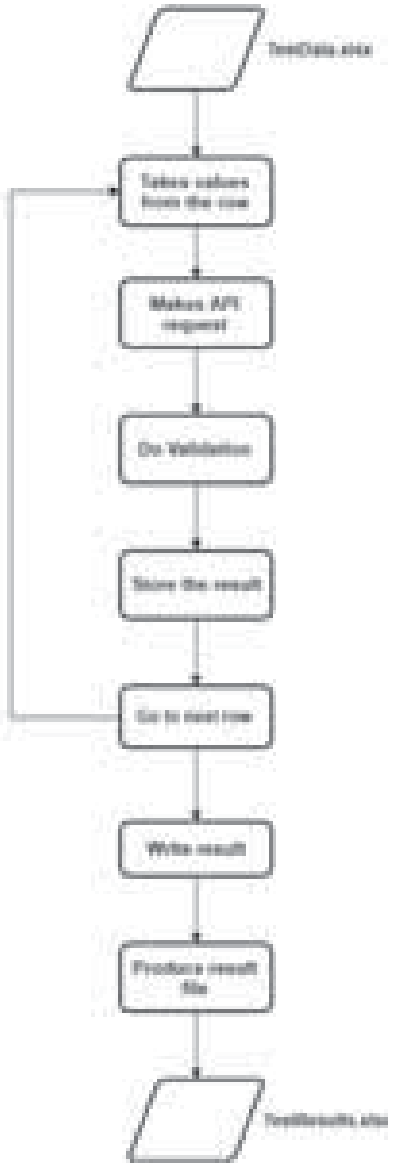

**Fig. 2.** REST API test automation framework process flow.

captured in the result file. The Table III shows the sample expected values for a single test case. The framework will read and use these values to validate with the actual values by passing these values to the test scripts created or

**Table II.** Sample result generated by the framework.

| Actual record | Actual error code | Actual response code | Result |
|---|---|---|---|
| { “errorCode”: 103”, errorMessage”: “Invalid credentials” } | 103 | 401 | Pass |

**Table III.** Sample input expected values for validation.

| Expected result | Expected record Count | Expected error code | Expected response code | Response time |
|---|---|---|---|---|
| Error code 103 and response Code 401 | | 103 | 401 | 00.00.00.7 53 |

**Fig. 3.** Analysis of response time for each REST API request.

developed for validation. The Table II shows the sample results generated by the API test automation framework where actual values are filled by the framework. The actual record field has the JSON response returned by the API. The actual response code is the status code returned by the API and the result column has the “pass” or “fail” value for the sample test specified in the Table I. Similarly the mentioned fields for all the test cases in the input test data file are filled by the framework.

## 5. RESULTS AND DISCUSSION

At the end the result file containing the actual values responded by the API and the validated result will be produced by the framework. The tester or the user can open the file and observe the test cases that are passed and failed. The result file also contains an additional field with response time of each test case that is time taken by the API to respond for a particular REST request (test) which serves as a verification for performance testing. The Figure 3 shows the pictorial representation of captured response time for each test.

## 6. CONCLUSION

Thus the manual regression testing is automated to increase efficiency and to reduce manual testing time. The process is automated with the help of a designed framework, which acts as a core for the entire process to complete. The framework is optimized to make the automation process simple. At the end of automation process the framework produces test results which acts as a detailed



RESEARCH ARTICLE

test report for the user to identify and analyze the failed test cases. Since the test report is generated in a well-known excel format, it is simple and easy to interpret and presented in a tabular format to the user. Since there can occur changes in test data and increase in API tests, it can be simply done by updating or modifying the input test data. Thus increases the maintainability. The framework follows a structured procedure and a flow to run the test automation as shown in Figure 2. Any modification in the test data and the expected result can be made directly in the input test data excel file. This proposed framework performs automation without any manual intervention and at the end produces a result file for error and bug detection. This test automation framework becomes very useful during regression testing where the already tested REST APIs need to be tested again. The framework can be enhanced by building GUI, which acts as a front end, where the testers or the user can able to interact with the framework easily by supplying the directory path of the created test data file and the path in which the tester or the user wishes to store the result file. A descriptive document can be created for the validation on what a particular validation function validates and what are all the parameters it expects, so that the user or the tester can refer the document while preparing the input test data file and give respective validation it should perform. The framework currently handles only JSON response, it can be made to handle XML response also in the future to make it more generic.

## References


1. Asha K.R. and Shwetha, D.J., **2015**. API Testing: Picking the Right Strategy. *Pacific Northwest Software Quality Conference*, p.1–20.
2. Sunil L. Bangare, Seema Borse, Pallavi S. Bangare and Shital Nandedkar, **2012**. Automated API testing approach. *International Journal of Engineering Science and Technology (IJEST)*, *4*, pp.673–676.
3. http://hc.apache.org/httpcomponents-client-4.2.x/httpclient/apidocs/index.html.
4. https://www.tutorialspoint.com/groovy/index.htm.
5. https://support.smartbear.com/readyapi/docs/soapui/tutorial/index.html.


Received: 30 August 2018. Accepted: 17 September 2018.