Deepa Govindasamy

AUTOMATION OF RESTFUL API – DEMO PROJECT

BY

DEEPA GOVINDASAMY

Version 1.0

13th April 2019

Deepa Govindasamy

Contents

Deepa Govindasamy

1.0 Scope and Objective

The scope of this exercise is to provide automated test solution to perform unit and integration testing of RESTful APIs. It includes the APIs **GET/albums** and **GET/users** from https://jsonplaceholder.typicode.com. The below testing checkpoints are considered the primary objectives during the development and testing phase of this exercise.

- Enabling continuous integration (CI)
- Incremental method of testing
- Command line execution
- Adherence to code quality standards, clear structure and readability

2.0 Approach

The approach towards the development of this automated solution considers the following.

- The automation of API testing involves **continuous integration and enhancement**.
- The automation package should enable easy testing by different stakeholders such as developers and BA, with **zero or minimum dependency on testers**.
- The suite is developed with **BDD approach** with easy readable test cases for the stakeholders, and it can easily fit into CI/CD approach with minimal configuration.

2.1 Choice of Scripting Language

The scripting language chosen for test development is **Javascript**.

2.2 Automation Framework

The automation suite is built using the framework "**Cypress**".

2.3 Pre-requisites

Following are the pre-requisites for the delivered artifact to successfully run and deliver results.

- Cypress tool should be installed and server in running mode.
- Javascripts should be enabled in the target browsers.
- The target machine should have uninterrupted access to internet.

2.4 Test Preparation

Sample feature files (albums.txt and users.txt) are created with test scenarios to demonstrate the test case design approach in BDD framework. Considering the demo project, the testing scope covered includes the following.

- Success code 200 OK
- Response header validation
- Response body content validation
- Response body data structure validation

- Field Level Validation – Datatype validation including regex, Length validation

Please note, there could be multiple other scenarios simulated, but the scope has been limited for the demo project.

2.5 Scripting

Two scripts are developed as part of scripting (albums.js and users.js), to cover 2 APIs respectively. The script covers the scenarios mentioned under the test preparation section. The scripts include necessary assertions to determine the test outcome. The test data for scripting is loaded using fixtures, while the test data itself is created in JSON.

The scripting is designed in such a way, the same script can be used during unit tests, as well as during the integration tests. Individual tests can also be skipped (using .skip function), as the scenario demands.

NOTE: The scripts are delivered with an extension .txt as part of my deliverable, to avoid being retained by mail servers. They should be converted back to .js format, before being executed.

2.6 Test Execution

The Cypress tool can be invoked either through the desktop application or command prompt.

2.6.1 Command Prompt Execution

Here are the commands to execute test cases from command prompt.

- Runs Cypress application until completion of testing exercise

```
$ cypress run
```

- Execute Single Test

```
$ cypress run --spec "cypress/integration/Albums.js"
```

- Execute multiple Test

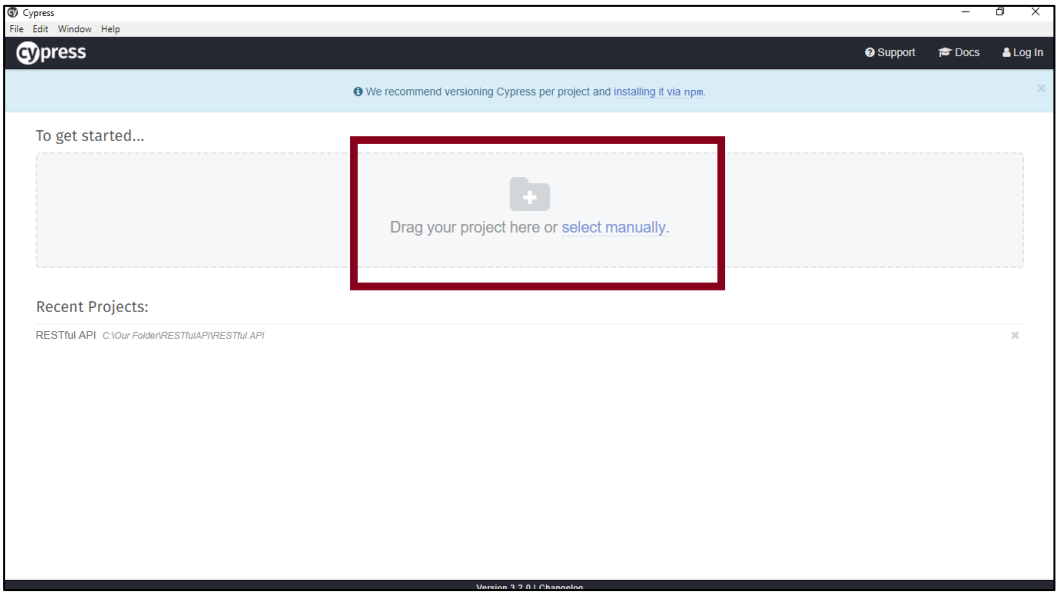
```
$ cypress run --spec "cypress/integration/Albums.js,
cypress/integration/Users.js"
```

2.6.2 UI Execution

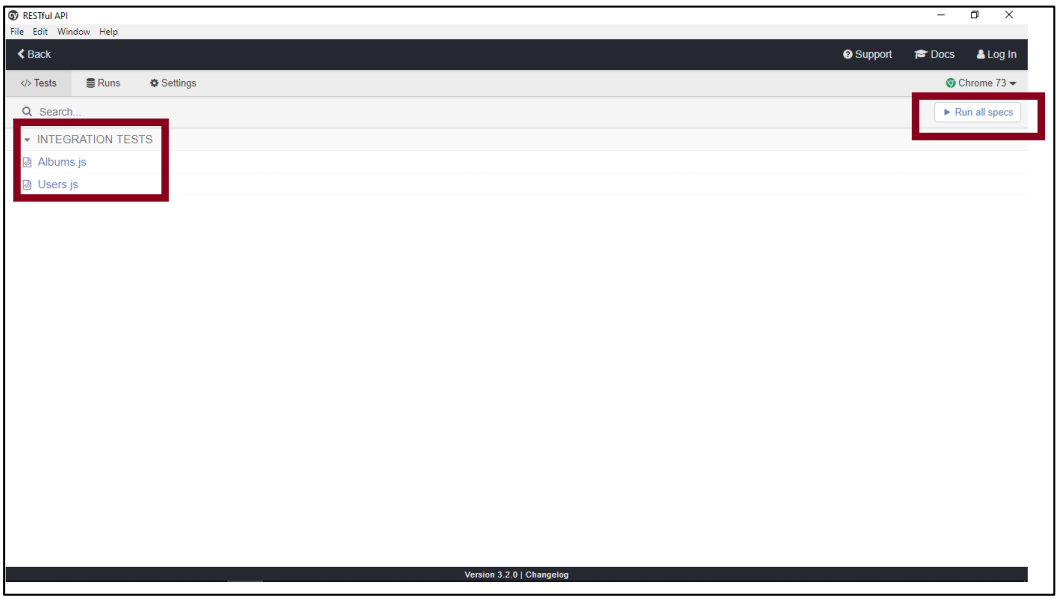
Here are the steps to execute test cases through UI.

Step 1 – Upon opening the tool for first time, we need to select the project manually. The tool provides an option to select from local drive on GITLab. Click on the respective project under recent project sections, to work on the existing project.

Deepa Govindasamy



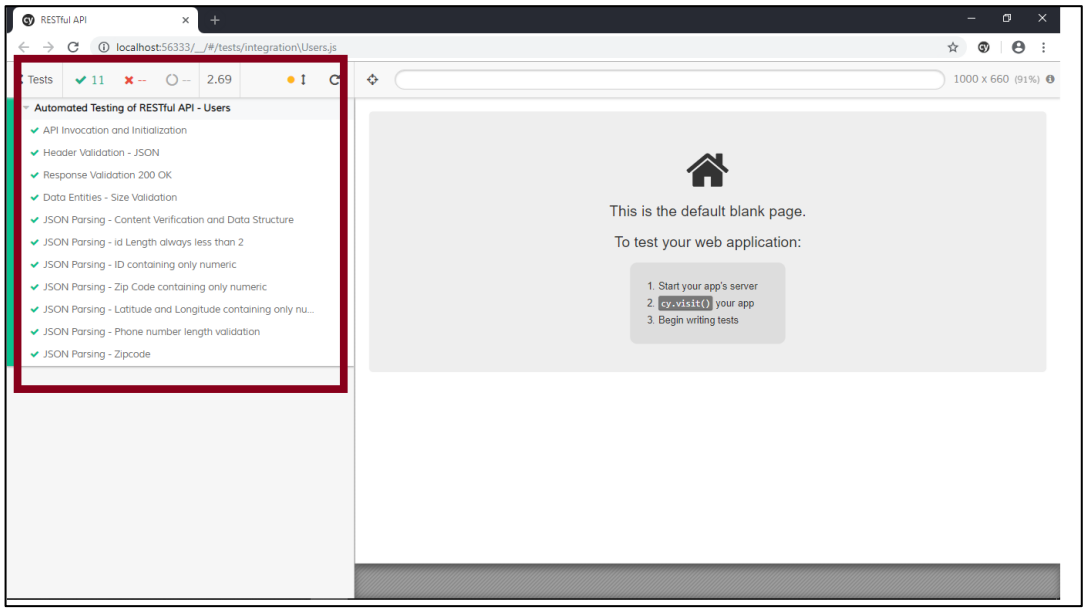
Step 2 - The automated tests created for the project, will get displayed under INTEGRATION TESTS. On clicking the respective test, the execution starts. The complete test suite can be executed on clicking the “Run all specs” button.

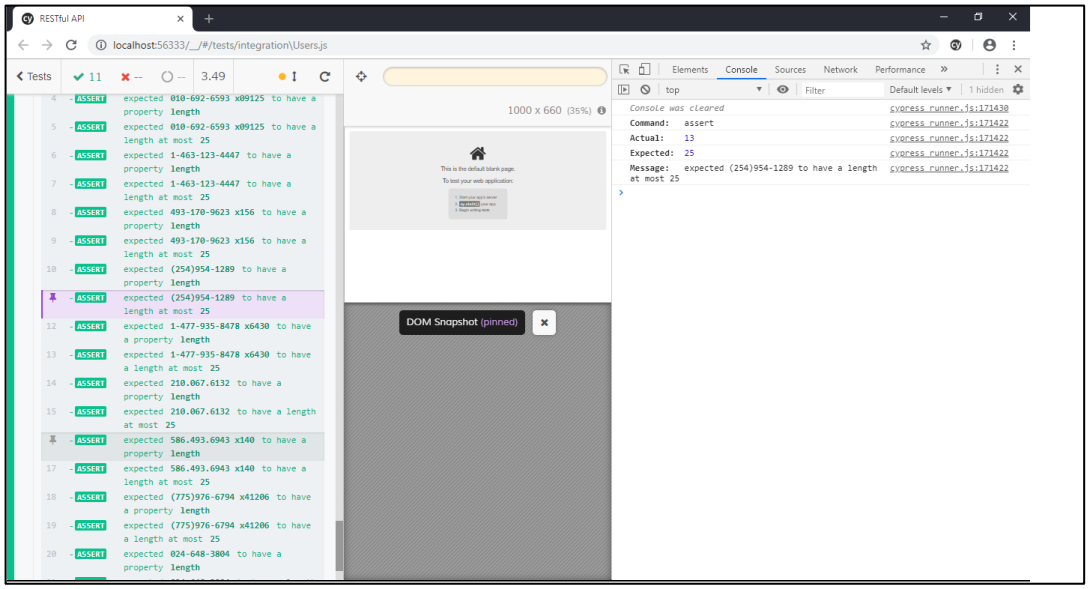


2.7 Reporting

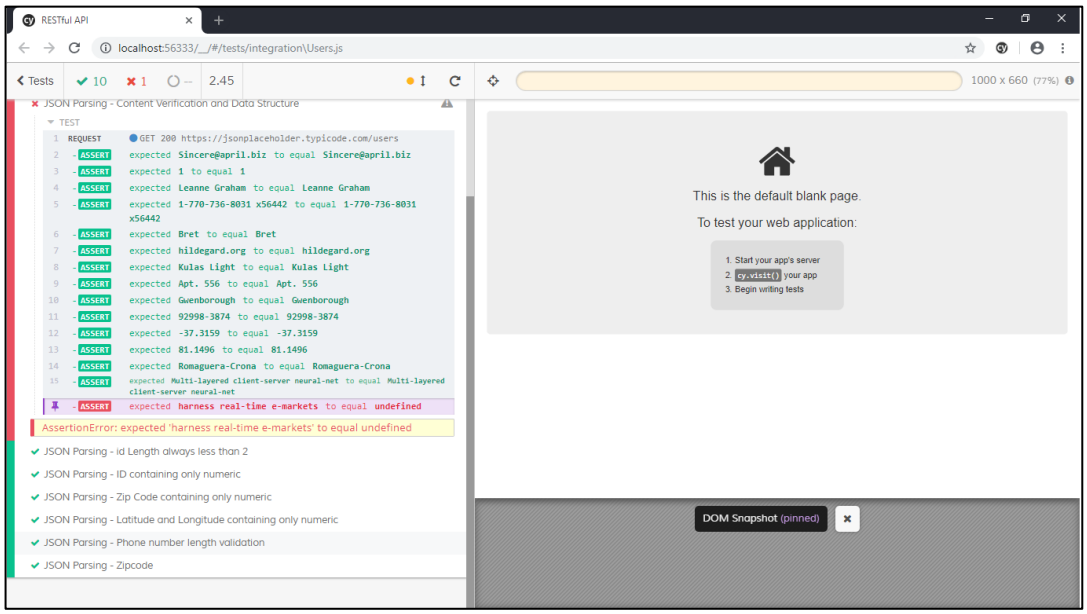
The results from test execution are placed below, highlighting few scenarios. The result includes the below information.

- List of all test suites created within the test, and an indication on its success/failure. We can also obtain a drilldown view on each of the asserts in our tests.
- Number of test suites passed / failed.
- Time taken for the entire test suite to run.





The below result is provided, to indicate the failures, by intentionally making the test case to fail.



3.0 Artifacts delivered for Evaluation

The following artifacts have been delivered for our evaluation.

Javascript files:

- albums.js
- users.js

Code Repository (Project) [like in GIT]:

- Entire project repository

Test data:

- album.json
- users.json

Video capture of test execution:

- Video capture of one full test execution of GET/albums API. This is provided, to get a view on test execution performed.

NOTE: .js and .json files are appended with .txt to ensure, they are not blocked by email servers during the transfer process. They should be renamed back, before execution.

4.0 Revision history

Version	Document Owner	Date Created / Modified	Contents
1.0	Deepa Govindasamy	13/04/2019	First version, after thorough review