API Testing with Robot Framework

Step 1: Install Library

i. Open command prompt (cmd)

ii. Type those two libraries:

```
pip install robotframework-requests
pip install robotframework-jsonlibrary
```

Step 2: Setup Setting on Robot Framework Script

i. Open Python IDE or Robot Framework IDE (RIDE)

ii. Create new test suite

iii. Under ‘Setting’ section, set three libraries

```
*** Settings ***
Library           RequestsLibrary
Library           JSONLibrary
Library           Collections

```

Step 3: Creating Test Case

For this documentation, https://reqres.in/ is based URL of API system.

i. Firstly, create new test case by typing test case name under 'Test Case Section'

ii. Type script as figure below for connecting to API system:

```
create session    API_testing    https://reqres.in/
```

or

```
*** Variables ***
${base_url}       https://reqres.in/

*** Test Cases ***
Get Request Single User
    create session    API_testing    ${base_url}
```

iii. The first is the keyword from library that. The second is the name of session which we give any name. Third is the API system's URL that we want to test.

iv. Or you can type those script on Test Setup under Setting as shown below:

```
*** Settings ***
Test Setup        create session    API_testing    ${base_url}
Library           RequestsLibrary
Library           JSONLibrary
Library           Collections

```

Note: Test Setup is the keywords/step that be run before test cases inside test suit be executed.

v. There are 4 test cases or requests that common be used in API testing: GET, POST, PUT, DELETE

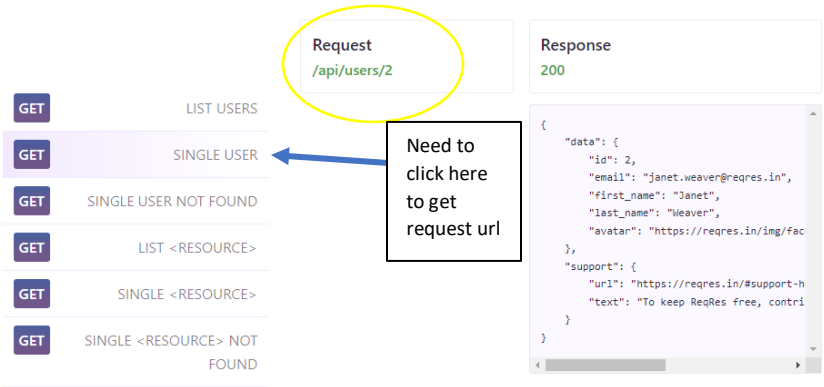
Step 3.1: GET request test cases

i. GET request is a keyword that gets information/data from API system.

ii. To start it, type the script that is shown below:

```
${get_response}=  get on session  API_testing  url=/api/users/2
```

iii. The green word represents a variable that will be used after this. The red words represent GET request keyword. The yellow word represent name of session tha we had named on CREATE session. The blue word represents the URL of user (with id=2) or anything that we will get information or data from this URL. That URL can be found on webpage API as shown in figure below:



iv. That varibale will be used for showing status code, content, and headers that are useful in Step 4 onwards.

v. Status code means the code of the response to tour request action. The script that is shown below can show the status code on log terminal in IDE.

```
log to console  ${get_response.status_code}
```

vi. Content is the information/data that we get from user URL (blue word). The script that is shown below can show the content from that URL on log terminal in IDE.

```
log to console  ${get_response.content}
```

vii. Headers. The script that is shown below can show the header of that URL on log terminal in IDE.

```
log to console  ${get_response.headers}
```

Step 3.2: POST request test case

i. POST request is a keyword that creates user by adding/uploading information/data to API system.

ii. Before writing POST request, we need to create a dictionary that contains information/data that will be uploaded to API system. Firstly, we need to create the body (data) of dictionary as shown below.

```
${body2}=  create dictionary  name=morpheus  job=leader
```

iii. The purple words represent the information/data that we want to upload. The variable (name, job) can be found in *content* that we had found in Step 3.1.

iv. After that, we need to set the header of dictionary as shown below:

```
${header2}=  create dictionary  Content-Type=application/json
```

v. The purple words represent the header information of dictionary. Type of content or other header information can be found in header that we had found in Step 3.1.

vi. Finally, we can start to write POST request. To start it, type the script that is shown below:

```
${post_response2}=  Post Request  API_testing  url=/api/users  data=${body2}
headers=${header2}
```

vii. The red words represent POST request keyword. The yellow word represent name of session tha we had named on CREATE session. The blue word represents the URL that we can post new user (can be found in webpage under POST section). The purple words represent

Step 3.3: PUT request test case

i. PUT request is a keyword that updates existing information/data in API system.

ii. Same as POST request, we need to create body and header of dictionary by writing script as shown below:

```
${body3}=    create dictionary    job=leaderExecutive
${header3}=    create dictionary    Content-Type=application/json
```

iii. We can change value on any variables inside user content (based on API itself). For this case, we change job from leader to leaderExecutive.

iv. We can start to write POST request. To start it, type the script that is shown below:

```
${put_response3}=    Put request    API_testing    /api/users/2    data=${body3}
headers=${header3}
```

v. The script is similar to POST request. The difference is the keyword which is PUT request and it use URL of user that we want to update.

Step 3.4: DELETE request test case

i. DELETE request is a keyword that deletes information/data from API system.
ii. To start it, type the script that is shown below:

```
${delete_response4}=    delete request    API_testing    url=/api/users/2
```

iii. The script is similar to GET request. The difference is it use DELETE request keyword and use URL of user that we want to delete.

Step 4: Status Code and Validation

i. Status/response code means the code of the response to our request action.
ii. The webpage of API system has shown the status that is supposed to be which it can be found on the webpage as shown below:

iii. From webpage of API system, response code for:
- Success GET request: 200
- User does not exist when use GET: 404
- Success POST new user: 201
- User already exits during POST: 200
- Success PUT request: 200
- Success DELETE request: 204

iv. Now we focus on how to validate status code. Firstly, we need to get status_code from request as shown in Step 3.1.

v. Now, we need to change status_code into string. We can convert it by using convert to string as shown below:

```
${status_code}=    convert to string    ${get_respond.status_code}
```

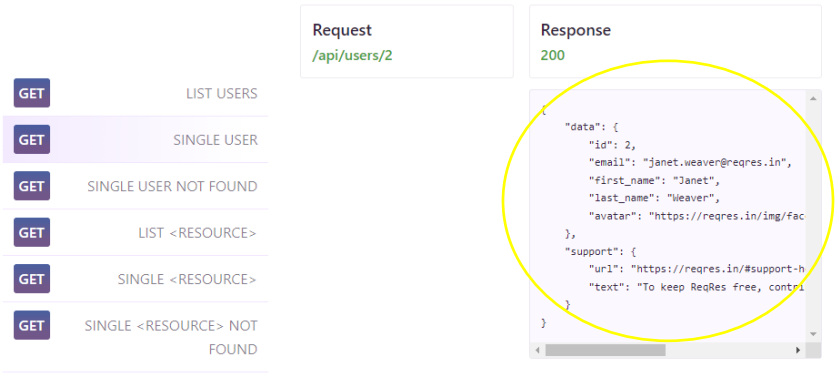
vi. ${status_code} is a varibale that store status_code in string type. Now we try validate ${status_code} by comparing between ${status_code} with actual status code. We compare by typing script as shown below:

```
should be equal    ${status_code}    200
```

vii. should be equal is a keyword that idetifiy whether the the second column (${status_code}) is same as third column (actual response code). If the both are same, it means that the validation is pass. If not, it means that the validation is fail.

Step 5: Content and Its Validation

i. The webpage of API system has shown the body/data of content that is supposed to have which it can be found on the webpage as shown below:



ii. Now we focus on how to validate content. Firstly, we need to get content variable which is ${get_response.content} from request as shown in Step 3.1.

iii. Now, we need to change the content variable into string. We can convert it by using convert to string as shown below:

${content}= convert to string ${get_respond. content}

iv. ${content} is a varibale that store data of content in string type. Now we try validate ${content} checking data inside it. We validate by typing script as shown below:

should contain ${content} Janet

v. should be equal is a keyword that idetifiy whether the the second column (${content}) the value/data of third column (actual data of content). This keyword can checks one or more data inside content. If ${content} contains actual data, it means the validation is pass. If not, it means the validation is fail.