REST Assured for Automating APIs Testing

Rekha[1], Kanchan[2], Neelkanth[3], Vaishali[4]

[1] Lecture, Computer Engineering, Jaihind Polytechnic , Maharashtra, India
[2] Student, Computer Engineering, SPCOE Otur, Maharashtra, India
[3] Student, Computer Engineering, AVCOE Sangmner, Maharashtra, India
[4] Lecture, Computer Engineering, Jaihind Polytechnic , Maharashtra, India

ABSTRACT

Rest Assured is used to verify the REST APIs with the help of the Java library. Java library acts like a headless client to act upon the Rest web services. The libraries based on the Rest Assured library are also capable of validating the HTTP responses from the server. Rest Assured enables you to test REST APIs using java libraries and integrates well with Maven. It has very efficient matching techniques, so asserting your expected results is also pretty straight forward. Rest Assured has methods to fetch data from almost every part of the request and response no matter how complex the JSON structures are.

Rest-Assured library also provides the ability to validate the HTTP Responses received from the server. For e.g. we can verify the Status code, Status message, Headers and even the Body of the response. This makes Rest-Assured a very flexible library that can be used for testing.

Keyword: - *REST APIs, HTTP Request, HTTP Response, JSON,Maven,TestNg.*

1. Why need Rest-Assured?

Imagine you open your google map view and look for a place you want to go, you immediately see close by restaurants, you see options for the commute; from some leading travel providers, and see so many options at your fingertips. We all know they are not google products, then how does Google manage to show it. They use the exposed APIs of these providers. Now, if you are asked to test this kind of setup, even before the UI is built or is under development, testing APIs becomes extremely important and testing them repeatedly, with different data combinations makes it a very suitable case for automation.

1.2 Client Server Basics

1. Client Server Architecture and HTTP Protocol
2. HTTP Request
3. HTTP Response

2.2. REST API Testing – Basics

- Configure Eclipse with Rest-Assured
- REST API Test using Rest Assured
- Validate Response Status using Rest Assured
- Validate Response Header using Rest Assured

- Read JSON Response Body using Rest Assured

In this blog, we'll discuss REST Assured's installation, as well as how to create and execute tests.

API Testing with RestAssured Framework.

During the development of an application, the RestAssured Framework assists in assessing and identifying errors. It is necessary to have knowledge of an IDE (IntelliJ or Eclipse), Maven, and frameworks such as TestNG, or JUnit in order to use RestAssured Framework for API Automation Testing. Rest APIs are tested by Manual Testing and Automation Testing. The most common way to test REST APIs is with automation, as it helps interpret results more effectively.

2. Rest API uses five HTTP methods for requesting commands as listed below:

- GET: Retrieves information from a specific URL.
- PUT: Updates previous resources or creates new information at the specified URL.
- PATCH: Partial updates.
- POST: Develops a new entity and is also used to send information to the server for uploading files or information.
- DELETE: Deletes the present representation at a particular URL.

3. API Rest Assured Features:

- Code Reusability: Since it is a Java client, it can be done as REST-assured in terms of code reuse.
- Customized Reports: It is a lightweight, multilingual test reporting tool that can be customized to specific needs. Graphical reports can be generated.
- Designing a Data-Driven framework: With REST-assured automation, data files are not a limitation for the automation runner.
- CI/CD integration: We can integrate into Jenkins tool.
- Maintenance: Rest-assured provides us with DSL (Domain Specific Language). As a result, we can perform behaviour-driven tests.

Therefore, we can conclude that REST-assured is better than other tools for automating these RESTful web services, as it requires less maintenance and more efficiency.

4. Requirement

1. Jdk 1.8
2. Eclipse / intellij
3. Rest Assured (library) (download the packages)
4. Simple JSON/ Google GSON/ Jackson

5. How to create Maven Project?

Eclipse - File -new - Project -Maven - Maven Project

Click on checkbox on create simple project - next - enter new maven project details.

Artifact - Group id - RESTDEC22

Artifact id - RESTDEC22

Version - 0.0.1-SNAPSHOT

Packaging – jar

Name - RESTDEC22

Description - RESTDEC22

Finish

Project get successfully created in eclipse.

Add the dependencies in POM.xml

Add the dependencies for Rest Assured and JSON with version.

```
<dependencies>

            <dependency>

   <groupId>io.rest-assured</groupId>

   <artifactId>rest-assured</artifactId>

   <version>5.3.0</version>

   <scope>test</scope>

</dependency>

             <dependency>

   <groupId>com.googlecode.json-simple</groupId>

   <artifactId>json-simple</artifactId>

   <version>1.1.1</version>

</dependency>

 </dependencies>
```

6. Rest Assured Automation Testing with TestNG:

The 'TestNG' framework is inspired by junit. This is the most recent Rest Assured framework. You can construct test cases in groups or as dependencies on another function. NG stands for 'NEXT GENERATION'.

7. Characteristics of Test NG:

1. Will get inbuilt report.
2. It Generate test reports as follows.
3. It support Parallel execution.
4. We can execute failed TC only.
5. Test grouping –smoke testing, regression testing.

8. Simple class and object creation using main method in eclipse:

```
    public void TC_01()
    {
        System.out.println(" Welcome to TestNG");
    }
    public void TC_02()
    {
        System.out.println("User can login successfully");
    }
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        rest1 obj= new rest1();
        obj.TC_01();
        obj.TC_02();
    }
}
```

9. Different attribute of @Test:

- @Test(invocationCount=3)
- @Test(enabled=false)
- @Test(dependsOnMethods="TC_01")
- @test (timeOut = 2000)
- @Test(alwaysRun=true)
- @Test(priority=1)
- @Test(groups={“smoke", "regression”})

Example :

```
import org.testng.annotations.Test;

public class WMattribute {

        @Test(timeOut=4000)

        public void TC_04() throws InterruptedException

        {System.out.println("time out and thres.sleep method ");

                Thread.sleep(2000);}

        @Test(alwaysRun=true)

        public void TC_05()

        {System.out.println("any case it will executed");}

        @Test(priority=1)

        public void TC_06()

        {System.out.println("assign priority to execute first and sequence.");}

        @Test(groups= {"Smoke","regression"})

        public void TC_07()

        {System.out.println("Execution of smoke and regression");}

        @Test(groups= {"functional","regression"})

        public void TC_08()

        {System.out.println("Execution of functional and regression");}

        @Test(groups= {"SIT"})

        public void TC_09()

        {System.out.println("Execution of SIT");}}
```

When group attribute can execute then it required TestNg.xml

Right click on java file -testing - convert to testing.

10.Annotation:

- @Test
- @BeforeMethod
- @AfterMethod

- @BeforeClass
- @AfterClass

11. Simple program for Rest Assured to validate the status code:

```
public class Reqresmethods {
  @Test
 //Simple Program for Expected status code=201
public void GetUser() {
                int ExpectedStatusCode=201;
                Response resp =RestAssured.get("https://gorest.co.in/public/v2/users");
                if(ExpectedStatusCode==resp.statusCode())
                {
                              System.out.println("statuscode match");
              }
  }
```

12. Create user using post method:

```
public class ReqresPost {
@Test
  public void CreateUser() {
                String reqBody = "{\n"
                                                        + "    \"name\": \"rekha\",\n"
                                                        + "    \"job\": \"Manager\"\n"
                                                        + "}";
              Response resp = RestAssured.
                                          given().
                                                                header("content-type","application/json").
                                                                body(reqBody).
                                          when()
```

```
                        .post("https://reqres.in/api/users");

                        System.out.println(resp.body().asPrettyString());

  }

@Test(enabled = true)

public void circuitCount() {

   given().

   when().

   get("http://ergast.com/api/f1/2017/circuits.json").

   then().

   assertThat().

   body("MRData.CircuitTable.Circuits.circuitId",hasSize(20)); }
```

given() — We provide all the input details here i.e. Base URI, Headers, Path Parameter, Query Parameter, and Request Body/Payload.

when() — We specify the Resource, HTTP Request method like POST, GET, PUT, PATCH, or DELETE.

then() — We validate the response i.e the response code, response time, response message, response headers, response body, etc.

There are other more Hamcrest matchers/assertors, such as "equalTo()", "lessThan()", "greaterThan()", "hasItem()", and so on. You might find the Hamcrest documentation at http://hamcrest.org/JavaHamcrest/ helpful .

In addition to the response body, other characteristics are being validated.REST Assured checks not only the response body but also technical elements of the response such as HTTP status code, response content type, and headers.

Let's look at an example of a test script for them.

```
@Test

public void headersAndStatus() {

   given().

   when().get("http://ergast.com/api/f1/2017/circuits.json").

   then().

   assertThat().

   statusCode(200).

   and().contentType(ContentType.JSON).
   and().header("Content-Length",equalTo("4567"));
 }
```

A second thing to note is how easy it is to concatenate two assertions using the and() method, which makes the code more understandable.

13. CONCLUSIONS

Since API Testing is a critical part of any development life cycle, REST Assured Framework is one of the most widely used Web Services Testing tools in Java. Advanced features, along with simplicity in implementation, make it a must for any testers to ensure quality in the end product. With its fluent approach and the expressive method names, it makes the call and its output easily understandable. Both JSONPath and Matchers further increase the power and expressiveness.

14. REFERENCES

[1] REST Assured: https://rest-assured.io/

[2]https://github.com/rest-assured/rest-assured/wiki/Usage

[3] TestNG Documents: https://testng.org/doc/

[4]URL: http://ergast.com/mrd/.

[5]RL: http://ergast.com/mrd/.

[6] https://mvnrepository.com/artifact/org.testng/testng/7.7.1

[7]https://gorest.co.in/