**WriteUP PHASEEND PROJECT**

**To automate functionalities using**[**https://petstore.swagger.io/**](https://petstore.swagger.io/)**REST API services**

Rest Assured is a Java library for testing for Rest APIs

Using these libraries you will java code to send a request to a URL and get a response back from the server

All of this will be done on your eclipse Console

You will use the same HTTP methods like GET, PUT, POST, DELETE in order to send request  to server

The Rest Assured library will get the HTTP response from the server and validate

Rest Assured library will help us to validate

 > repsonse code

 > Response message

 > response Header

 > Response Body

Whenever we create a HTTP request we will give details

Examples:

Base URL: https://reqres.in

Resource: /pages/users

Query Parameters : ?q=London

path paramters= :users/:owner

We will test 4 main HTTP methods : CRUD operations

 > GET : This method will be used to read the data form the server

 > Post: This method will be used to create a new resource on the server

> PUT or PATCH: this method will be used to update a resource on the server

> DELETE : this method will be used to delete a resource on the server

==================================

rest Assured also supports BDD (Behaviour driven development)

Rest assured code is written using Gherkin keywords(give, when,then,and,but)

Thereby making it easy to read and write and understand

Rest Assured library gives us Gherkin method to write the Rest api code

======================================

given() : this method is used in rest assured to pass the Base URL, headers, parameters, req body

when()  : HTTP method name

then() : validate and add assettions

The code that we will write today is divided in 3 parts:

=====================

1. request + paramters + body

2. http method

3. Assertion, log output

1. Create a quickstart maven project in eclipse

2. Install Cucumber plugin in Eclipse

3. Select all trusted sites and click on Restart eclipse.

Go to Your cucumber project → POM.xml file and remove the existing depenedecies section. Compelete section remove it.

<dependencies>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-api</artifactId>

<version>2.14.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-core -->

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-core</artifactId>

<version>2.14.1</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.11</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-java</artifactId>

<version>7.10.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-junit</artifactId>

<version>7.10.1</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-core</artifactId>

<version>7.10.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.rest-assured/rest-assured -->

<dependency>

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

<artifactId>rest-assured</artifactId>

<version>5.3.0</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/io.rest-assured/rest-assured-common -->

<dependency>

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

<artifactId>rest-assured-common</artifactId>

<version>5.3.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.rest-assured/rest-assured-all -->

<dependency>

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

<artifactId>rest-assured-all</artifactId>

<version>5.3.0</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/io.rest-assured/json-path -->

<dependency>

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

<artifactId>json-path</artifactId>

<version>5.3.0</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.json/json -->

<dependency>

<groupId>org.json</groupId>

<artifactId>json</artifactId>

<version>20180813</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.hamcrest/hamcrest -->

<dependency>

<groupId>org.hamcrest</groupId>

<artifactId>hamcrest</artifactId>

<version>2.2</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.7.1</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.15.2</version>

</dependency>

</dependencies>

4. For this create 3 packages in the project under src/test/java

5. Create a feature file.

6. Create a file with extension .feature

7. Right click on the file and select the option pretty format.

Now we will change Maven project to Cucumber project

Right click on your project →go to Configure → click on convert to Cucumber project

8. Save the feature file → you steps will be highlighted

That means -> for these steps we have not written selenium code

9. Create Step definition file

**POSTMAN**

1.Create a new collection with name LessonEndProject- demo

2.Add a request:

Give request name as LessonEndProject -demo

And give URL as : <https://api.openweathermap.org/data/2.5/weather?appid=aa9126a7d7b4c20f33602b045a197e3f&q=London>

Save the request

Create a variable local to this request only -> we will use pre-req scripts:

Add the variable in the request

A variable local to the request, should also be part of Environment.

3.Lets create an environment:

Go to the request and select the environment, only then the variable will be resolved and value will be fetched.

4.Now change the environment variable value in the pre-req script to New York

Send the request → you will see response for New york city

Suppose we have to use a variable in all the requests of the collection then create a collection variable:

If the variable value is constant across all the request in that particular collection → we will create collection variable

5.Add the Variable in the request → making a collection variable

6.Global variables and connect with the request

7. Write test case to check the Assertion

8. Monitoring the Collection

Save and execute the request.