**Tutorial on Using REST Assured with JSON for API Testing**

### **1. Introduction**

* **What is REST Assured?**  
  REST Assured is a Java library used for testing RESTful APIs. It simplifies writing HTTP requests, validating responses, and automating API testing.  
  Think of REST Assured as a tool that lets us send requests to an API and check if the responses are correct.
* **What is JSON?**  
  JSON (JavaScript Object Notation) is a lightweight format for storing and transferring data. APIs often use JSON to send and receive data because it's easy to read and write.
* **Purpose of this Tutorial:**
  + Learn to test APIs using REST Assured.
  + Understand how to send GET and POST requests.
  + Validate API responses using assertions.

### **2. Setting Up the Project**

#### **2.1 Dependencies Explanation**

We will use the following dependencies in our project:

1. **REST Assured:** This library is used for sending API requests and validating responses.
2. **JSON Library:** This is used for creating and managing JSON objects in our tests.
3. **TestNG:** A popular testing framework for running and managing test cases.

#### **2.2 Add Dependencies in Maven**

Here’s how to set up your project dependencies in the pom.xml file:

<dependencies>

<!-- REST Assured -->

<dependency>

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

<artifactId>rest-assured</artifactId>

<version>5.3.0</version>

</dependency>

<!-- JSON Library -->

<dependency>

<groupId>org.json</groupId>

<artifactId>json</artifactId>

<version>20230618</version>

</dependency>

<!-- TestNG -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.8.0</version>

<scope>test</scope>

</dependency>

</dependencies>

**Why are these dependencies important?**

* **REST Assured:** To send API requests (GET, POST) and verify responses.
* **JSON Library:** To easily create and manipulate JSON data for API requests.
* **TestNG:** To run the tests in an organized way and generate detailed reports.

### **3. Understanding and Writing GET Requests**

#### **3.1 What is a GET Request?**

A GET request is used to retrieve data from an API. For example, if we want to see a list of all available pets, we use a GET request. Visit the petstore swagger for clear understanding.  
Endpoint:

https://petstore.swagger.io/v2/pet/findByStatus?status=available

#### **3.2 Code Example and Explanation**

Here’s how to write and execute a GET request using REST Assured and TestNG:

import io.restassured.RestAssured;

import io.restassured.response.Response;

import org.testng.annotations.Test;

import static org.hamcrest.Matchers.\*;

public class GetRequestExample {

@Test

public void fetchAvailablePets() {

// Set the base URL for the API

RestAssured.baseURI = "https://petstore.swagger.io/v2";

// Send a GET request with a query parameter

Response response = RestAssured.given()

.queryParam("status", "available")

.when()

.get("/pet/findByStatus");

// Print the response body

System.out.println("Response Body: " + response.getBody().asString());

// Assertions to validate the response

response.then().statusCode(200); // Check if the status code is 200

response.then().body("[0].status", equalTo("available")); // Validate the status of the first pet

}

}

**Explanation of Code:**

1. **RestAssured.baseURI:** Sets the base URL for all requests.
2. **given():** Prepares the request.
3. **queryParam:** Adds query parameters (e.g., status=available).
4. **when():** Specifies the HTTP method (GET).
5. **then():** Validates the response using assertions.

### **4. Understanding and Writing POST Requests**

#### **4.1 What is a POST Request?**

A POST request is used to send data to an API. For example, we can use POST to add a new pet to the database. Visit the petstore swagger for clear understanding.  
Endpoint:

https://petstore.swagger.io/v2/pet

#### **4.2 Code Example and Explanation**

Here’s how to write and execute a POST request:

import io.restassured.RestAssured;

import org.json.JSONObject;

import io.restassured.response.Response;

import org.testng.annotations.Test;

import static org.hamcrest.Matchers.\*;

public class PostRequestExample {

@Test

public void addNewPet() {

// Set the base URL

RestAssured.baseURI = "https://petstore.swagger.io/v2";

// Create JSON payload for the request

JSONObject requestParams = new JSONObject();

requestParams.put("id", 12345);

requestParams.put("name", "Tommy");

requestParams.put("status", "available");

// Send a POST request with the JSON body

Response response = RestAssured.given()

.header("Content-Type", "application/json")

.body(requestParams.toString())

.when()

.post("/pet");

// Print the response body

System.out.println("Response Body: " + response.getBody().asString());

// Assertions

response.then().statusCode(200); // Validate status code

response.then().body("name", equalTo("Tommy")); // Validate pet name

}

}

**Explanation of Code:**

1. **JSONObject:** Creates the JSON payload.
2. **header:** Specifies that the request content is in JSON format.
3. **body:** Sends the JSON payload.
4. **Assertions:** Verify that the API response matches the expected output.

### **5. Assertions Based on Responses**

#### **What are Assertions?**

Assertions are checks to ensure that the API response meets the expected criteria.

#### **Examples of Assertions:**

1. **Status Code:**

response.then().statusCode(200);

Validates that the server responded successfully.

1. **Response Body:**

response.then().body("name", equalTo("Tommy"));

Checks if the response body contains the expected values.

1. **Header Validation:**

response.then().header("Content-Type", "application/json");

### **6. Complete Workflow Example**

Here’s a full workflow that adds a new pet and retrieves it:

import io.restassured.RestAssured;

import org.json.JSONObject;

import io.restassured.response.Response;

import org.testng.annotations.Test;

import static org.hamcrest.Matchers.\*;

public class FullWorkflow {

@Test

public void addAndFetchPet() {

// Base URL

RestAssured.baseURI = "https://petstore.swagger.io/v2";

// Step 1: Add a new pet

JSONObject requestParams = new JSONObject();

requestParams.put("id", 12345);

requestParams.put("name", "Tommy");

requestParams.put("status", "available");

Response postResponse = RestAssured.given()

.header("Content-Type", "application/json")

.body(requestParams.toString())

.when()

.post("/pet");

postResponse.then().statusCode(200);

// Step 2: Fetch pets by status

Response getResponse = RestAssured.given()

.queryParam("status", "available")

.when()

.get("/pet/findByStatus");

getResponse.then().statusCode(200);

// Step 3: Validate added pet exists

getResponse.then().body("find { it.id == 12345 }.name", equalTo("Tommy"));

}

}

### **7. Practice Tasks**

1. Create a DELETE request to remove the pet and validate it.
2. Try updating a pet using a PUT request.