Rest Assured

Rest Assured is a group of java libraries which enables us to automate Rest API testing.

Rest Assured is Java-based, and knowledge of core Java suffices for learning it​

It helps fetch values of request and response from complicated JSON structures​

The API request can be customized with a variety of header, query, path param, and any session or cookies to be set.​

It helps set assert statements and conditions.​

While Rest Assured is very helpful when the response is JSON type

Advantages of **REST** Assured

It is an **open-source tool**; you do not need to worry about the licensing cost.​

It supports all the **HTTP methods.**​

It has a **JSON schema validation library** to validate the JSON schema.​

**Parsing** and**validating XML** and **JSON**can be done easily in REST Assured.​

You can integrate the REST Assured with other java libraries like TestNG and Junit as a test framework.​

It supports **multiple authentication mechanisms**available for APIs.

Disadvantages of **REST** Assured​

It does not support the testing of SOAP API, but it can be done with some changes.​

To use REST Assured, you need a strong foundation in Java.

Dependencies we needed

1. Rest assured

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

<dependency>

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

<artifactId>rest-assured</artifactId>

<version>5.3.2</version>

<scope>test</scope>

</dependency>

​

1. Json-path

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

<dependency>

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

<artifactId>json-path</artifactId>

<version>5.3.2</version>

<scope>test</scope>

</dependency>

​

1. Json

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

<dependency>

<groupId>org.json</groupId>

<artifactId>json</artifactId>

<version>20231013</version>

</dependency>

​

1. Gson​

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

<dependency>

<groupId>org.immutables</groupId>

<artifactId>gson</artifactId>

<version>2.10.0</version>

</dependency>

1. TestNG​

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

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.7.1</version>

<scope>test</scope>

</dependency>

1. Json-schema-validation​

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

<dependency>

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

<artifactId>json-schema-validator</artifactId>

<version>5.4.0</version>

</dependency>

Gherkin keywords

**Given** = a context/Getting a system in a particular state​  
​

**When** = something happens/Poke it​  
​

**Then** =we expect some outcome/examine the new state

Static Packages

import static io.restassured.RestAssured.\*;

import static io.restassured.matcher.RestAssuredMatchers.\*;

import static org.hamcrest.Matchers.\*;

HTTP REQUESTS

[GET](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET)​

The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.​

[POST](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST)​

The POST method submits an entity to the specified resource, often causing a change in state or side effects on the server.​

[PUT](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PUT)​

The PUT method replaces all current representations of the target resource with the request payload.​

[DELETE](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/DELETE)​

The DELETE method deletes the specified resource.

Get Request Script

public class RestAssureHTTPreq {  
   
  @Test  
  public void getUser()  
  {  
    given()  
  
    .when()  
       .get("[https://reqres.in/api/users?page=2")](https://reqres.in/api/users?page=2)  
    .then()  
       .statusCode(200)  
       .and()  
       .body("page", equalTo(2))  
       .and()  
       .log().all();  
  }

}

Note:

How many way we can create a request body

1. HashMap
2. JSONObject
3. POJO

Post Request using HashMap

public class RestAssureHTTPreq {

@Test  
  public void createUser()  
  {  
  
    //Create a request using HASHMAP  
  
    HashMap data=new HashMap();  
    data.put("name", "sandeep");  
    data.put("mail", "sandeep.velpula@ojas-it.com");  
  
  
     
    given()  
       .contentType(ContentType.JSON)  
       .body(data)  
  
  
    .when()  
       .post("[https://reqres.in/api/users")](https://reqres.in/api/users)  
        
      
  
     .then()  
         
     
      .statusCode(201)

       .log().all();  
  
  }

}

Post Request using JsonObject

public class RestAssureHTTPreq {

@Test(priority=1)  
  public void createUser()  
  {  
  
  
    //create a request using JSONObject  
  
    JSONObject data=new JSONObject();  
    data.put("name", "sweety");  
    data.put("age", 3);  
  
     
    given()  
       .contentType(ContentType.JSON)  
       .body(data)  
  
  
    .when()  
       .post("[https://reqres.in/api/users")](https://reqres.in/api/users)  
         
   
  
    .then()  
         
     
       .statusCode(201)  
       .log().all();  
  
  }}

Post Request using POJO

public class RestAssureHTTPreq {

@Test(priority=1)  
  public void createUser()  
  {  
  
    //create a request using POJO class  
  
    Pojo data=new Pojo();  
    data.setName("akhilesh");  
    data.setAge(5);  
  
 given()  
       .contentType(ContentType.JSON)  
       .body(data)  
 .when()  
       .post("[https://reqres.in/api/users")](https://reqres.in/api/users)  
       .then()  
       .statusCode(201)  
       .log().all();  
  
  }

}

POJO Class

public class Pojo {  
  String name;

int age;  
  public String getName() {  
    return name;  
  }  
  public void setName(String name) {  
    this.name = name;  
  }  
  public int getAge() {  
    return age;  
  }  
  public void setAge(int age) {  
    this.age = age;  
  }  
  }

How to capture ID using Post

public class RestAssureHTTPreq {

Int id;

@Test  
  public void createUser()  
  {  
  
    //Create a request using HASHMAP  
  
    HashMap data=new HashMap();  
    data.put("name", "sandeep");  
    data.put("mail", "sandeep.velpula@ojas-it.com");  
  
  
     
    id=given()  
       .contentType(ContentType.JSON)  
       .body(data)  
  
  
    .when()  
       .post("[https://reqres.in/api/users")](https://reqres.in/api/users)

.jsonPath().getInt(“id”);  
        
      
  
      }

}

PUT request using Capturing ID

public class RestAssureHTTPreq {

@Test(priority=2,dependsOnMethods = {"createUser"})  
  public void updateUser()  
  {  
    HashMap data=new HashMap();  
    data.put("name", "sweety");  
    data.put("mail", "sandeep.vel@ojas-it.com");  
  
    given()  
     .contentType("application/json")  
     .body(data)  
  
    .when()  
     .put("[https://reqres.in/api/users/"+id)](https://reqres.in/api/users/)  
     //.jsonPath().getInt("id");  
  
    .then()  
    .statusCode(200)  
    .log().all();  
  }  
  
}

Note:

Here we need post request script(captured id) for updating the data.

DELETE request using Capturing ID

public class RestAssureHTTPreq {  
   
  @Test  
  public void getUser()  
  {  
    given()  
  
    .when()  
       .get("[https://reqres.in/api/users/"+id)](https://reqres.in/api/users?page=2)  
    .then()  
       .statusCode(200)  
       .and()  
       .body("page", equalTo(2))  
       .and()  
       .log().all();  
  }

}

Note:

Here we need post request script(captured id) for updating the data.

Path and Query parameters

Example:

<https://reqres.in/api/users?page=2>

in the above example

<https://reqres.in> -----------> Domain

/api/users -------------------> Path parameters

?page=2 ----------------------> Query parameter

Script:

-------

public class PathAndQueryParameters {

   @Test  
   public void path\_QueryParams()  
   {  
     given()  
         .pathParam("path1","api")  
         .pathParam("path2","users")  
         .queryParam("page", 2)  
         .queryParam("id", 8)  
  
     .when()  
         .get("[https://reqres.in/{path1}/{path2}")](https://reqres.in/%7bpath1%7d/%7bpath2%7d)  
     .then()  
         .statusCode(200)  
         .log().all();  
   }

 }

Cookies

Cookies are the information sent by the server and stored in the browser.

Get single Cookies:

----------------

It will give the single single key’s value

Return type of getCookie(“Key”) is String

Get All Cookies:

----------------

It will give all the cookies information.

Return type of getCookies() is Map<String,String>

Note:

------

If we want to get only keys information then we use keySet() method

Headers

Headers are additional piece of information sent along with the request to the server.

Get single Header:

----------------

It will give the single single Headers value

Return type of getHeader(“Key”) is String

Get All Headers:

----------------

It will give all the headers information.

Return type of getHeaders() is Headers.

Get single Cookie

**public** **class** CookiesInfo {

@Test(priority=1)

**public** **void** getSingleCookie()

{

Response res=*given*()

.contentType(ContentType.***JSON***)

.when()

.get("https://www.google.com/");

String getCookieValue = res.getCookie("AEC");

System.***out***.print(getCookieValue);

}

}

Get All Cookies

**public** **class** CookiesInfo {

@Test(priority=2)

**public** **void** getSingleCookies()

{

Response res=*given*()

.contentType(ContentType.***JSON***)

.when()

.get("https://www.google.com/");

Map<String,String> getCookies = res.getCookies();

System.***out***.println(getCookies.keySet());

}

}

Headers Information

public class HeaderText {  
   
  @Test  
  public void header()  
  {  
    given()  
  
    .when()  
       .get("[https://www.google.com/")](https://reqres.in/api/users?page=2)  
    .then()  
      .header(“Content-Type”,”text/html; charset=ISO-8859-1”)

.and()

.header(“Server”,”gws”)  
  }

}

Get single Header

**public** **class** GetHeader {

@Test(priority=1)

**public** **void** getSingleHeader()

{

Response res=*given*()

.contentType(ContentType.***JSON***)

.when()

.get("https://www.google.com/");

String getHeaderValue = res.getHeader("AEC");

System.***out***.print(getHeaderValue);

}

}

Get All Headers

**public** **class** HeadersInfo {

@Test(priority=3)

**public** **void** getAllHeaders()

{

Response res=*given*()

.contentType(ContentType.***JSON***)

.when()

.get("https://www.google.com/");

Headers getHeaders = res.getHeaders();

//System.out.println(getCookies);

**for**(Header hd:getHeaders)

{

System.***out***.println(hd.getName()+" "+hd.getValue());

}

}

}

Parsing JSON response body:

JSONPATH:

{

    "page": 2,

    "per\_page": 6,

    "total": 12,

    "total\_pages": 2,

    "data": [

        {

            "id": 7,

            "email": "michael.lawson@reqres.in",

            "first\_name": "Michael",

            "last\_name": "Lawson",

            "avatar": "https://reqres.in/img/faces/7-image.jpg"

        },

        {

            "id": 8,

            "email": "lindsay.ferguson@reqres.in",

            "first\_name": "Lindsay",

            "last\_name": "Ferguson",

            "avatar": "https://reqres.in/img/faces/8-image.jpg"

        },

        {

            "id": 9,

            "email": "tobias.funke@reqres.in",

            "first\_name": "Tobias",

            "last\_name": "Funke",

            "avatar": "https://reqres.in/img/faces/9-image.jpg"

        },

        {

            "id": 10,

            "email": "byron.fields@reqres.in",

            "first\_name": "Byron",

            "last\_name": "Fields",

            "avatar": "https://reqres.in/img/faces/10-image.jpg"

        },

        {

            "id": 11,

            "email": "george.edwards@reqres.in",

            "first\_name": "George",

            "last\_name": "Edwards",

            "avatar": "https://reqres.in/img/faces/11-image.jpg"

        },

        {

            "id": 12,

            "email": "rachel.howell@reqres.in",

            "first\_name": "Rachel",

            "last\_name": "Howell",

            "avatar": "https://reqres.in/img/faces/12-image.jpg"

        }

    ],

    "support": {

        "url": "https://reqres.in/#support-heading",

        "text": "To keep ReqRes free, contributions towards server costs are appreciated!"

    }

}

Example:

package JsonResponseValidator;

import static org.hamcrest.CoreMatchers.equalTo;

import org.testng.annotations.Test;

import io.restassured.http.ContentType;

import static io.restassured.RestAssured.\*;  
import static io.restassured.matcher.RestAssuredMatchers.\*;

public class ResponseData {  
  
  @Test  
  public void approch1()  
  {  
    given()  
  
      .contentType(ContentType.JSON)  
  
    .when()  
      .get("[https://reqres.in/api/users?page=2")](https://reqres.in/api/users?page=2%22))  
  
    .then()  
      .statusCode(200)  
      .body("data[0].last\_name", equalTo("Lawson"))  
      .log().all();  
  }

}

Example2:

Approch2:

package JsonResponseValidator;

 import static org.hamcrest.CoreMatchers.equalTo;

 import org.testng.annotations.Test;

 import io.restassured.http.ContentType;

import static io.restassured.RestAssured.\*;  
import static io.restassured.matcher.RestAssuredMatchers.\*;

 public class ResponseData {  
  
  @Test  
  public void approch1()  
  {  
    given()  
  
      .contentType(ContentType.JSON)  
  
    .when()  
      .get("[https://reqres.in/api/users?page=2")](https://reqres.in/api/users?page=2%22))  
  
    .then()  
      .statusCode(200)  
      .body("data[0].last\_name", equalTo("Lawson"))  
      .log().all();  
  }

}

Note:

Here we can use JSON PATH FINDER to find the path of the json.