RestAssured

RestAssured is an API/Library through which we can automate RestAPI’s.

Pre-requisites:

1. Java 9+ & Eclipse
2. TestNG Framework
3. Maven (comes with Eclipse)

Install TestNG from Eclipse marketplace.

Specify all dependencies in pom.xml

Create Maven Project.



Src/main/java & src/main/resources => used by developers

Src/test/java & src/test/resources => used by testers

Check TestNG is present:



Dependencies:

RestAssured: <https://rest-assured.io/>

Dependencies: <https://github.com/rest-assured/rest-assured/wiki/GettingStarted>

Add dependencies: Maven, Gradle, XmlPath, etc.

Dependencies to add in pom.xml:

1. io.rest-assured (io is package name) => rest-assured.
2. json-path
3. json
4. gson
5. testng
6. scribejava-apis
7. json-schema-validator
8. xml-schema-validator

Get dependencies from <https://mvnrepository.com/>.

1. io.rest-assured (io is package name) => rest-assured: <https://mvnrepository.com/artifact/io.rest-assured/rest-assured/5.4.0>.
2. json-path: <https://mvnrepository.com/artifact/io.rest-assured/json-path/5.4.0>
3. json: <https://mvnrepository.com/artifact/org.json/json/20240303>.
4. gson: <https://mvnrepository.com/artifact/com.google.code.gson/gson/2.10.1>.
5. testng: <https://mvnrepository.com/artifact/org.testng/testng/7.9.0> .
6. scribejava-apis: <https://mvnrepository.com/artifact/com.github.scribejava/scribejava-apis/8.3.1> .
7. json-schema-validator: <https://mvnrepository.com/artifact/com.github.java-json-tools/json-schema-validator/2.2.14> .
8. xml-schema-validator:

HTTP Requests:

1. get
2. post
3. put
4. delete

Ex: <https://reqres.in/>

Gherkin -keywords

By default, restassured supports BDD Ghekin langusge so no need to add dependencies like cucumber

TestNG style

* given():

Content type, set cookies, add auth i.e. authentication, add param i.e. parameters, set headers info i.e. information, etc.

* when():

get, post, put, delete requests.

* then():

validate status code, extract response, extract headers cookies & response body

Add static Packages: <https://github.com/rest-assured/rest-assured/wiki/GettingStarted#static-imports>

io.restassured.RestAssured.\*

io.restassured.matcher.RestAssuredMatchers.\*

org.hamcrest.Matchers.\*

For which ever method starts first we don’t need to specify dot ‘.’ Else start with dot.

Ex: given()

.when()

.then()

Given ex:

GET List user: <https://reqres.in/api/users?page=2> . Response code: 200

GET user: <https://reqres.in/api/users/2> . Response code: 200

POST Create user: <https://reqres.in/api/users>. Response code: 201

{

"name": "morpheus",

"job": "leader"

}

PUT Update user: <https://reqres.in/api/users/2> . Response code: 200

{

"name": "morpheus",

"job": "zion resident"

}

DELETE user: <https://reqres.in/api/users/2> . Response code: 204

Static packages, ex:

**package** day1HTTPMatehods;

**import** org.testng.annotations.Test;

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

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

**import** **static** org.hamcrest.Matchers.\*;

/\*

given():

Content type, set cookies, add auth i.e. authentication, add param i.e. parameters, set headers info i.e. information, etc.

when():

get, post, put, delete requests.

then():

validate status code, extract response, extract headers cookies & response body

\*/

**public** **class** HTTPRequests {

//using testng test

//for single user

@Test

**void** getUser() {

*given*()

.when()

.get("https://reqres.in/api/users?page=2")

.then()

.statusCode(200)

.body("page", *equalTo*(2))

.log().all();

}

}

**Example:**

**package** day1HTTPMatehods;

**import** org.testng.annotations.Test;

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

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

**import** **static** org.hamcrest.Matchers.\*;

**import** java.util.HashMap;

/\*

given():

Content type, set cookies, add auth i.e. authentication, add param i.e. parameters, set headers info i.e. information, etc.

when():

get, post, put, delete requests.

then():

validate status code, extract response, extract headers cookies & response body

\*/

**public** **class** HTTPRequests {

**int** id;

//using testng test

//for single user

@Test(priority=1)

**void** getUser() {

*given*()

.when()

.get("https://reqres.in/api/users?page=2")

.then()

.statusCode(200)

.body("page", *equalTo*(2))

.log().all();

}

@Test(priority=2)

**void** createUser() {

HashMap data = **new** HashMap(); //hashmap is in key value pair

data.put("name", "rutuja");

data.put("job", "sdet");

id = *given*()

.contentType("application/json")

.body(data)

.when()

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

.jsonPath().getInt("id");

/\*

.then()

.statusCode(201)

.log().all();

\*/

}

@Test(priority=3,dependsOnMethods = {"createUser"})

**void** updateUser() {

HashMap data = **new** HashMap();

data.put("name", "John");

data.put("job", "SDE");

*given*()

.contentType("application/json")

.body(data)

.when()

.put("https://reqres.in/api/users/"+id)

.then()

.statusCode(200)

.log().all();

}

@Test(priority=4)

**void** deleteUser() {

*given*()

.when()

.delete("https://reqres.in/api/users/"+id)

.then()

.statusCode(204)

.log().all();

}

}

**Creating Post Request Payloads in Multiple Ways**

How many ways we create request body:

1. Hashmap
2. Using org.json
3. Using POJO (Plain Old Java Object)
4. Using external json file

Start local server

