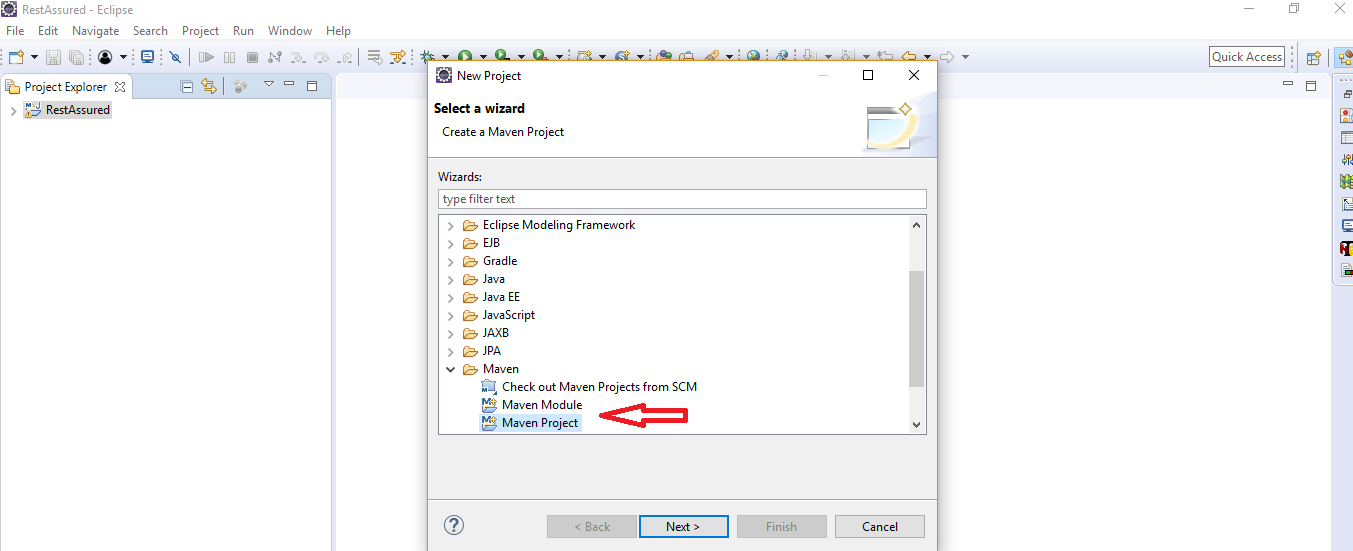
1 Setup or Configure REST Assured

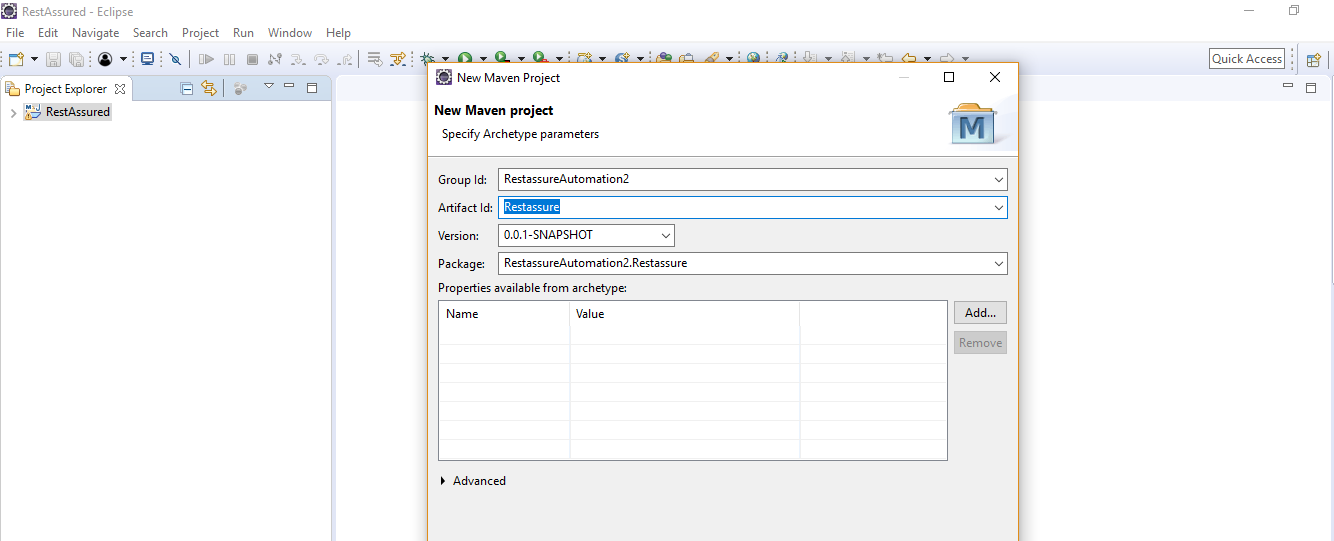
**Step 1.1:** Creating a Maven project in Eclipse

REST Assured is an API designed for automating REST services or REST APIs.

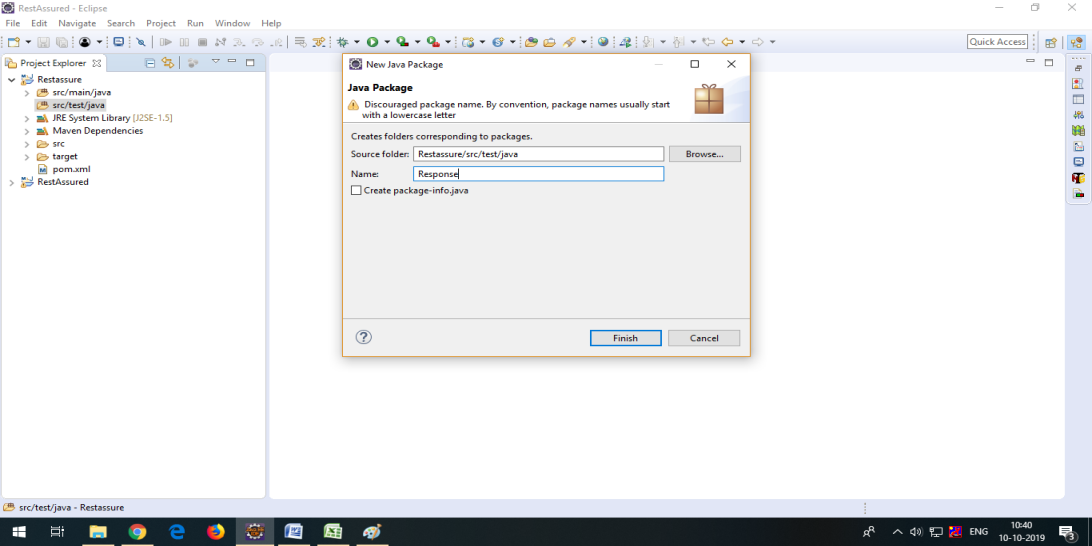
* Open Eclipse.
* Click on File---> click on New--->Project.
* Select the Maven project and click on Next.



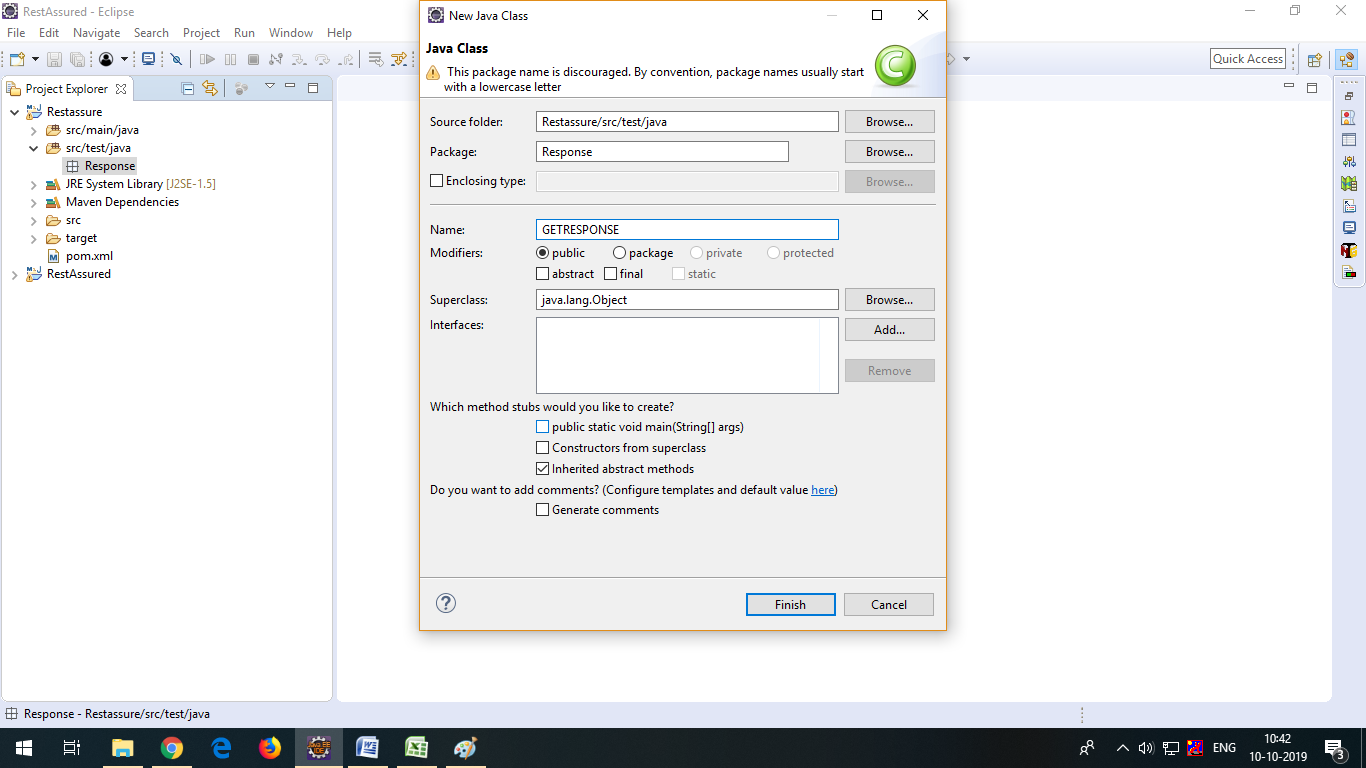
* Enter the Group id, Artifact id, and click on Finish.



* Right click on Project---> src/test/java---> Package.
* Enter the package name and click on Finish.

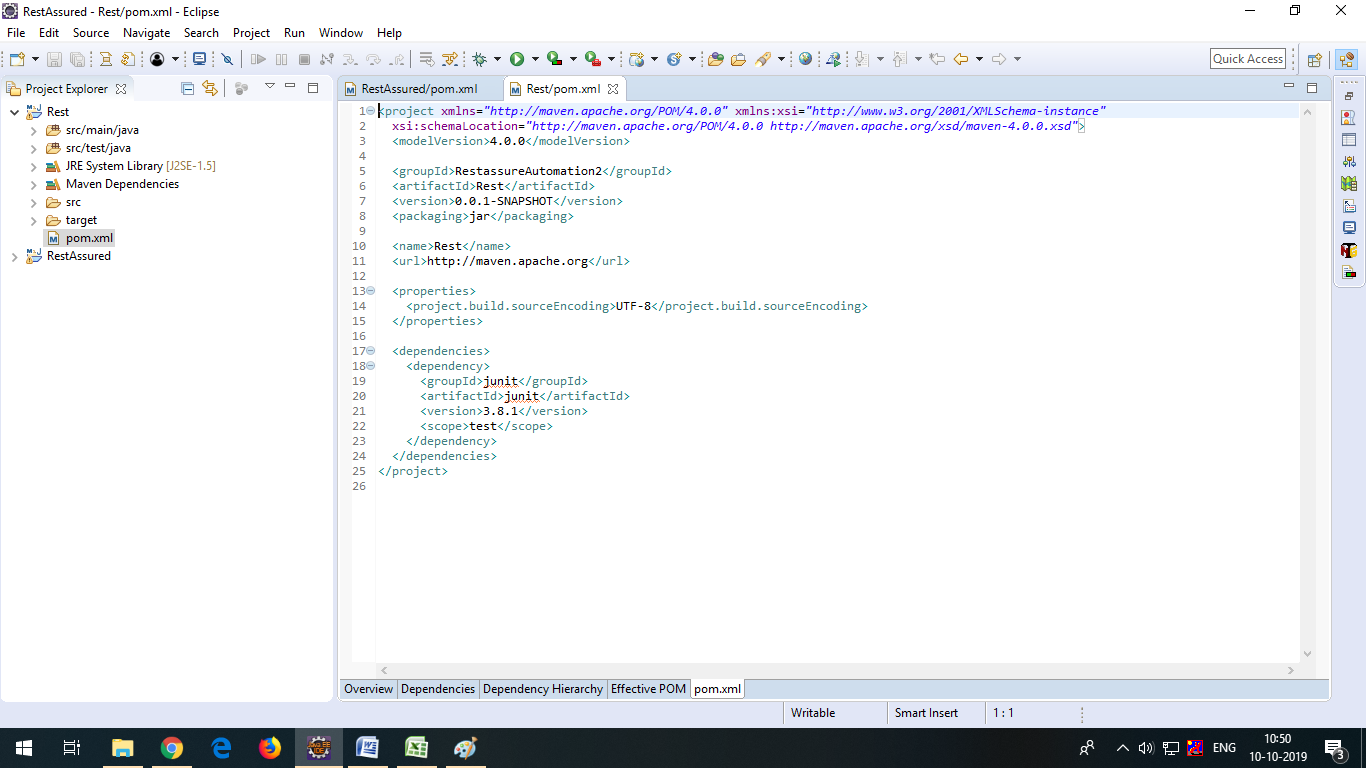


* Right click on Package---> New---> Class.
* Enter the class name and click on Finish.

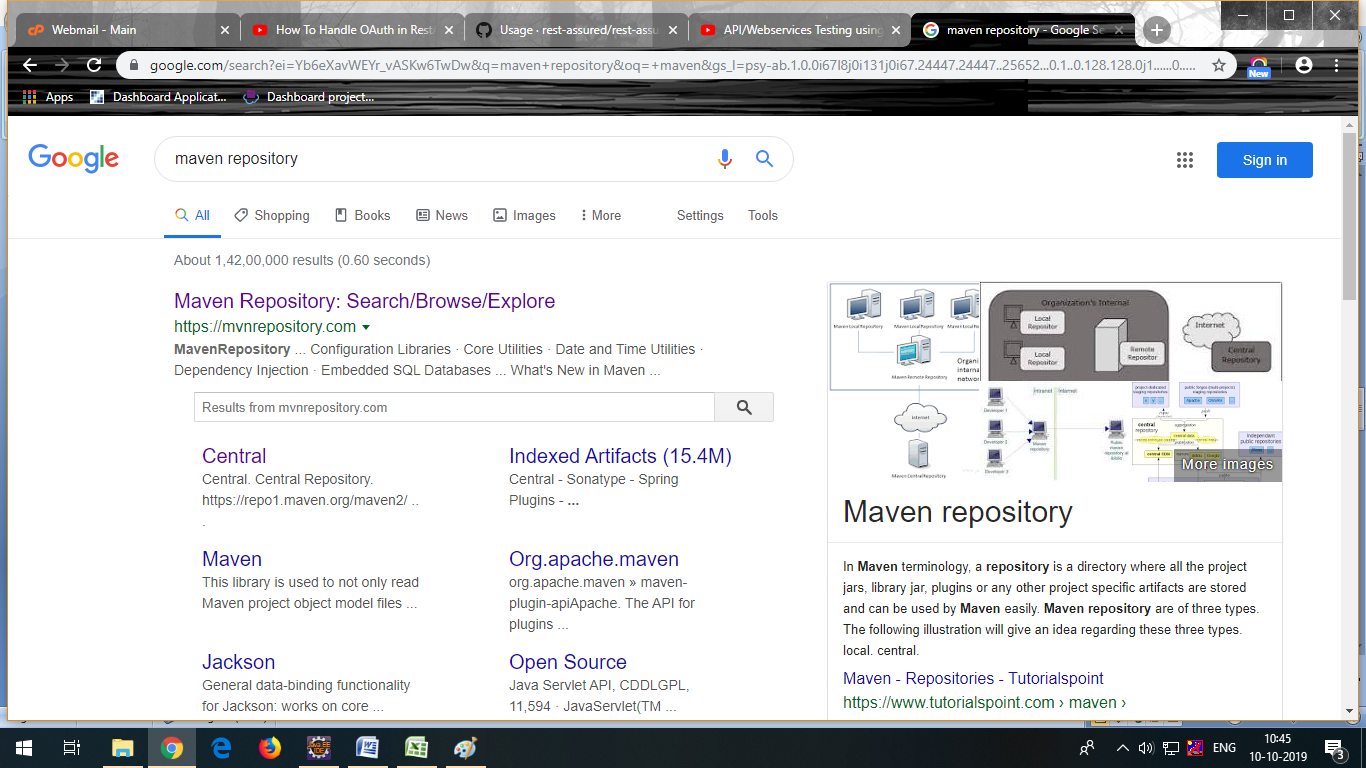


**Step 1.2:** Updating the pom.xml file with the latest stable REST Assured dependencies

* Open the pom.xml file.



* Go to a browser and search for Maven repository.



* Add the dependencies to the pom.xml file, after adding all the dependencies the pom.xml file will look like this:

<dependencies>

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

<dependency>

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

<artifactId>rest-assured</artifactId>

<version>3.3.0</version>

<scope>test</scope>

</dependency>

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

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>6.14.3</version>

<scope>test</scope>

</dependency>

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

<dependency>

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

<artifactId>json-simple</artifactId>

<version>1.1.1</version>

</dependency>

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

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi</artifactId>

<version>4.1.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>4.1.0</version>

</dependency>

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

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-java</artifactId>

<version>2.0.0</version>

</dependency>

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

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-junit</artifactId>

<version>2.0.0</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/info.cukes/gherkin -->

<dependency>

<groupId>info.cukes</groupId>

<artifactId>gherkin</artifactId>

<version>2.12.2</version>

<scope>provided</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/info.cukes/cucumber-jvm-deps -->

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-jvm-deps</artifactId>

<version>1.0.5</version>

<scope>provided</scope>

</dependency>

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

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-jvm</artifactId>

<version>2.0.0</version>

<type>pom</type>

</dependency>

<!-- https://mvnrepository.com/artifact/net.masterthought/cucumber-reporting -->

<dependency>

<groupId>net.masterthought</groupId>

<artifactId>cucumber-reporting</artifactId>

<version>1.0.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.5.3</version>

</dependency><!-- https://mvnrepository.com/artifact/junit/junit -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

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

<dependency>

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

<artifactId>log4j-core</artifactId>

<version>2.12.1</version>

</dependency>

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

<dependency>

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

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

<version>4.1.2</version>

</dependency>

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

<dependency>

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

<artifactId>json-path</artifactId>

<version>4.1.1</version>

</dependency>

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

<dependency>

<groupId>org.hamcrest</groupId>

<artifactId>java-hamcrest</artifactId>

<version>2.0.0.0</version>

<scope>test</scope>

</dependency>

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

<dependency>

<groupId>org.hamcrest</groupId>

<artifactId>hamcrest-core</artifactId>

<version>2.2-rc1</version>

<scope>test</scope>

</dependency>

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

<dependency>

<groupId>org.hamcrest</groupId>

<artifactId>hamcrest-library</artifactId>

<version>2.2-rc1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.codehaus.groovy</groupId>

<artifactId>groovy-all</artifactId>

<version>2.4.5</version>

</dependency>

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

<dependency>

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

<artifactId>xml-path</artifactId>

<version>3.0.0</version>

</dependency>

</dependencies>

* Click on Save.

2 GET Request and Response Automation

**Step 2.1:** Creating a Maven project in Eclipse

Get Method: Get method is used to retrieve data from the server at the specified resource.

* Open Eclipse.
* Click on file ---> click on New ---> Project.
* Select the Maven project and click on Next.
* Enter the Group id, Artifact id, and click on Finish.
* Right click on Project---> src/test/java---> Package.
* Enter the package name and click Finish.
* Right click on Package---> New---> Class.
* Enter the class name and click Finish.
* Add dependencies to pom.xml file.

**Step 2.2:** Executing the GET request and response program

* Write the program GET request using REST Assured and click on Save.

**package** GetResponse;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

import io.restassured.http.Method;

import io.restassured.response.Response;

import io.restassured.specification.RequestSpecification;

import junit.framework.Assert;

public **class** GetResponse1 {

@Test

void getempDetails()

{

//specify baseUrI

RestAssured.baseURI="http://192.168.1.207:8080/api/employee/search";

//Request object

RequestSpecification httpRequest=RestAssured.given();

//Response object

Response response=httpRequest.request(Method.GET,"/8095393564");

//print response in console window

String responseBody=response.getBody().asString();

         System.out.println("Response Body is:" +responseBody);

//status code validation

int statusCode=response.getStatusCode();

System.out.println("status code is :"+statusCode);

Assert.assertEquals(200,statusCode);

}

}

* Click on run and check the output in TestNG.

[RemoteTestNG] detected TestNG version 6.14.3

Response Body is:{"empId":3,"empName":"Sumit Kumar","empAddress":"Bellandur","mobileNumber":8095393564,"department":"development","project":"Cervical cancer application","teamLead":"RamaKrishnan","salary":10000.0,"joiningDate":"24-06-19"}

status code is :200

PASSED: getempDetails

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

Default test

Tests run: 1, Failures: 0, Skips: 0

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

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

Default suite

Total tests run: 1, Failures: 0, Skips: 0

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

3 POST Request and Response Automation

**Step 3.1:** Creating a Maven project in Eclipse

POST Request: POST requests are used to send data to the API server to create or update a resource. The data sent to the server is stored in the [request body](https://stackoverflow.com/questions/22034144/what-does-it-mean-http-request-body) of the HTTP request.

* Open Eclipse.
* Click on File---> click on New--->Project.
* Select the Maven project and click on Next.
* Enter the Group id, Artifact id, and click on Finish.
* Right click on Project---> src/test/java---> Package.
* Enter the package name and click Finish.
* Right click on Package---> New---> Class.
* Enter the class name and click Finish.
* Add dependencies to the pom.xml file.

**Step 3.2:** Executing the POST request and response program

* Write the program POST request using REST Assured and click on Save.

**package** GetResponse;

import org.json.simple.JSONObject;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

import io.restassured.http.Method;

import io.restassured.response.Response;

import io.restassured.specification.RequestSpecification;

import junit.framework.Assert;

public **class** PostResponse {

@Test

void RegistrationSuccessful()

{

//specify base URI

RestAssured.baseURI="http://192.168.1.207:8080/api/employee/";

//Request object

RequestSpecification httpRequest=RestAssured.given();

//contains the information in the json format

//Request payload sending along with post request

JSONObject requestParams=new JSONObject();

requestParams.put("empName","Lavanya");

requestParams.put("empAddress","Bomanahalli");

requestParams.put("mobileNumber","9900321102");

requestParams.put("department","testing");

requestParams.put("teamLead","Aruna");

requestParams.put("salary","10000");

requestParams.put("joiningDate","14-05-19");

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

httpRequest.body(requestParams.toJSONString());//attach above data to the request

//Response object

Response response=httpRequest.request(Method.POST,"/add");

//print response in console window

String responseBody=response.getBody().asString();

System.out.println("Response Body is:" +responseBody);

//status code validation

int statusCode=response.getStatusCode();

System.out.println("Status code is:" +statusCode);

Assert.assertEquals(200,statusCode);

}

}

* Click on run and check the output in TestNG.

[RemoteTestNG] detected TestNG version 6.14.3

Response Body is:58

Status code is:200

PASSED: RegistrationSuccessful

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

Default test

Tests run: 1, Failures: 0, Skips: 0

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

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

Default suite

Total tests run: 1, Failures: 0, Skips: 0

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

}

4 Send XML Payload in REST Assured,  
Parse XML Response in REST Assured,   
Parse JSON Response in REST Assured,   
and Explore Native Logging of REST Assured

**Step 4.1:** Creating a Maven project in Eclipse

* Open Eclipse.
* Click on File---> click on New--->Project.
* Select the Maven project and click on Next.
* Enter the Group id, Artifact id, and click on Finish.
* Right click on Project---> src/test/java---> Package.
* Enter the package name and click Finish.
* Right click on Package---> New---> Class.
* Enter the class name and click on Finish.
* Add dependencies to pom.xml file.

**Step 4.2:** Sending XML Payload in REST Assured

* Write the program for XML Payload in REST Assured and click on Save.

**package** Response;

import static io.restassured.RestAssured.given;

import org.hamcrest.Matchers;

import org.testng.Assert;

import org.testng.annotations.Test;

import io.restassured.http.ContentType;

import io.restassured.response.Response;

public **class** XMLResponse

{

@Test

public void post\_xml\_test() {

Response response = given().contentType(ContentType.XML).accept(ContentType.XML).body("<Employee>" +

"<empName>Lavanya Gowda</empName>" + "<empAddress>abc</empAddress>"

+ "<mobileNumber>1592211560</mobileNumber>" + "<department>abc</department>" + "<project>abc</project>"

+ "<teamLead>abc</teamLead>" + "<salary>10000</salary>" + "<joiningDate>11-10-19</joiningDate>"

+ "</Employee>").when().post("http://192.168.1.207:8080/api/employee/add/xml");

System.out.println("POST Response\n" + response.asString());

// tests

int statusCode = response.getStatusCode();

System.out.println("Status code is:" + statusCode);

Assert.assertEquals(200, statusCode);

}

}

* Click on run and check the output in TestNG

[RemoteTestNG] detected TestNG version 6.14.3

POST Response

78

Status code is:200

PASSED: post\_xml\_test

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

Default test

Tests run: 1, Failures: 0, Skips: 0

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

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

Default suite

Total tests run: 1, Failures: 0, Skips: 0

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

**Step 4.3:** Parsing XML Response in REST Assured

* Write the program for XML Response in REST Assured and click on Save.

**package** Response;

import org.testng.annotations.Test;

import org.testng.Assert;

import io.restassured.RestAssured;

import io.restassured.http.ContentType;

import io.restassured.response.Response;

import static io.restassured.RestAssured.\*;

public **class** ParseXML

{

@Test

public void parse\_xml\_test() {

Response response = given().contentType(ContentType.XML).accept(ContentType.XML).body("<Employee>" +

"<empName>Lavanya Gowda</empName>" + "<empAddress>abc</empAddress>"

+ "<mobileNumber>1591111560</mobileNumber>" + "<department>abc</department>" + "<project>abc</project>"

+ "<teamLead>abc</teamLead>" + "<salary>10000</salary>" + "<joiningDate>11-10-19</joiningDate>"

+ "</Employee>").when().post("http://192.168.1.207:8080/api/employee/add/xml");

System.out.println("POST Response\n" + response.asString());

// tests

int statusCode = response.getStatusCode();

System.out.println("Status code is:" + statusCode);

Assert.assertEquals(200, statusCode);

}

}

* Click on run and check the output in TestNG.

[RemoteTestNG] detected TestNG version 6.14.3

POST Response

85

Status code is:200

PASSED: parse\_xml\_test

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

Default test

Tests run: 1, Failures: 0, Skips: 0

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

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

Default suite

Total tests run: 1, Failures: 0, Skips: 0

**Step 4.4:** Parsing JSON Response in REST Assured

Write the program for Parse JSON Response in REST Assured and click on Save.

**package** Response;

import static org.testng.Assert.assertEquals;

import java.util.List;

import org.json.simple.JSONObject;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

import io.restassured.http.Method;

import io.restassured.path.json.JsonPath;

import io.restassured.response.Response;

import io.restassured.specification.RequestSpecification;

import junit.framework.Assert;

public **class** JsonParse {

    @Test

    public void testJsonParsing() {

        //specify base URI

RestAssured.baseURI="http://192.168.1.207:8080/api/employee/";

//Request object

RequestSpecification httpRequest=RestAssured.given();

//contains the information in the json format

//Request payload sending along with post request

JSONObject requestParams=new JSONObject();

requestParams.put("empName","Lavanya");

requestParams.put("empAddress","Bomanahalli");

requestParams.put("mobileNumber","9109320002");

requestParams.put("department","testing");

requestParams.put("teamLead","Aruna");

requestParams.put("salary","10000");

requestParams.put("joiningDate","14-05-19");

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

httpRequest.body(requestParams.toJSONString());//attach above data to the request

//Response object

Response response=httpRequest.request(Method.POST,"/add");

//print response in console window

String responseBody=response.getBody().asString();

System.out.println("Response Body is:" +responseBody);

//status code validation

int statusCode=response.getStatusCode();

System.out.println("Status code is:" +statusCode);

Assert.assertEquals(200,statusCode);

}

}

* Click on run and check the output in TestNG.

[RemoteTestNG] detected TestNG version 6.14.3

Response Body is:81

Status code is:200

PASSED: testJsonParsing

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

Default test

Tests run: 1, Failures: 0, Skips: 0

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

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

Default suite

Total tests run: 1, Failures: 0, Skips: 0

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

**Step 4.5:** Exploring Native Logging of REST Assured

* Write the program for Native Logging of REST Assured and click on Save.

**package** Response;

import org.testng.annotations.Test;

import static io.restassured.RestAssured.given;

import static io.restassured.RestAssured.when;

public **class** Logging

{

@Test

public void testLogging1()

{

given().get("http://192.168.1.207:8080/employee/api/search/8970922880").

then()

//.log().headers();

//.log().body();

//.log().cookies();

.log().all();

}

//Logs only in case of errors

@Test

public void testLogging2()

{

given().

get("http://192.168.1.207:8080/employee/api").

then().

log().ifError();

}

//conditional logging

@Test

public void testLogging3()

{

given().

get("http://192.168.1.207:8080/employee/api/search/8970922880").

then().

log().ifStatusCodeIsEqualTo(200);

}

}

* Click on run and check the output in TestNG.

[RemoteTestNG] detected TestNG version 6.14.3

HTTP/1.1 404

Content-Type: application/hal+json;charset=UTF-8

Transfer-Encoding: chunked

Date: Fri, 11 Oct 2019 07:01:31 GMT

{

"timestamp": "2019-10-11T07:01:31.720+0000",

"status": 404,

"error": "Not Found",

"message": "No message available",

"path": "/employee/api/search/8970922880"

}

HTTP/1.1 404

Content-Type: application/hal+json;charset=UTF-8

Transfer-Encoding: chunked

Date: Fri, 11 Oct 2019 07:01:31 GMT

{

"timestamp": "2019-10-11T07:01:31.961+0000",

"status": 404,

"error": "Not Found",

"message": "No message available",

"path": "/employee/api"

}

PASSED: testLogging1

PASSED: testLogging2

PASSED: testLogging3

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

Default test

Tests run: 3, Failures: 0, Skips: 0

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

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

Default suite

Total tests run: 3, Failures: 0, Skips: 0

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

5 GET, POST, XML, and JSON

**Step 5.1:** Problem statement for GET, POST, XML, and JSON

* **Objective:** As a part of developing a functionality, get the list of employees in a particular organization based on the number of parameters passed.
* **Following requirements should be met for problem statement:**
* Create a Maven project.
* Create a JSON package inside the Maven project.
* Create a class inside the package.
* Create a method for GET, POST, and XML to validate the response code from API for using REST Assured.
* Create a JSON Object.
* Create an HTTP request.

**Step 5.2:** Solution for the problem statement

Write the program for GET request, POST request, and XML payload using REST API in REST Assured and click on Save.

package Response;

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

**import** org.json.simple.JSONObject;

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

**import** io.restassured.RestAssured;

**import** io.restassured.http.ContentType;

**import** io.restassured.http.Method; **import** io.restassured.response.Response;

**import** io.restassured.specification.RequestSpecification;

**import** junit.framework.Assert;

public class PostResponse {

@Test

**void** RegistrationSuccessful()

{

//specify base URI

RestAssured.baseURI="http://192.168.1.207:8080/api/employee/";

//Request object

RequestSpecification httpRequest=RestAssured.given();

//contains the information in the json format

//Request payload sending along with post request

JSONObject requestParams=new JSONObject();

requestParams.put("empName","Lavanya");

requestParams.put("empAddress","Bomanahalli");

requestParams.put("mobileNumber","9119321102");

requestParams.put("department","testing");

requestParams.put("teamLead","Aruna");

requestParams.put("salary","10000");

requestParams.put("joiningDate","14-05-19");

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

httpRequest.body(requestParams.toJSONString());//attach above data to the request

//Response object

Response response=httpRequest.request(Method.POST,"/add");

//print response in console window

**String** responseBody=response.getBody().asString();

System.out.println("Response Body is:" +responseBody);

//status code validation

**int** statusCode=response.getStatusCode();

System.out.println("Status code is:" +statusCode);

Assert.assertEquals(200,statusCode);

}

@Test

**void** getempDetails()

{

//specify baseUrI

RestAssured.baseURI="http://192.168.1.207:8080/api/employee/search";

//Request object

RequestSpecification httpRequest=RestAssured.given();

//Response object

Response response=httpRequest.request(Method.GET,"/8970922880");

//print response in console window

**String** responseBody=response.getBody().asString();

System.out.println("Response Body is:" +responseBody);

//status code validation

**int** statusCode=response.getStatusCode();

System.out.println("status code is :"+statusCode);

Assert.assertEquals(200,statusCode);

}

@Test

public **void** post\_xml\_test() {

Response response = given().contentType(ContentType.XML).accept(ContentType.XML).body("<Employee>" +

"<empName>Lavanya Gowda</empName>" + "<empAddress>abc</empAddress>"

+ "<mobileNumber>1592210060</mobileNumber>" + "<department>abc</department>" + "<project>abc</project>"

+ "<teamLead>abc</teamLead>" + "<salary>10000</salary>" + "<joiningDate>11-10-19</joiningDate>"

+ "</Employee>").when().post("http://192.168.1.207:8080/api/employee/add/xml");

System.out.println("POST Response\n" + response.asString());

// tests

**int** statusCode = response.getStatusCode();

System.out.println("Status code is:" + statusCode);

Assert.assertEquals(200, statusCode);

}

}

* Click on run and check the output in TestNG.

[RemoteTestNG] detected TestNG version 6.14.3

Response Body is:88

Status code is:200

Response Body is:{"empId":5,"empName":"Lavanya Gowda","empAddress":"bomanahalii","mobileNumber":8970922880,"department":"testing","project":"Cervical cancer application","teamLead":"Aruna","salary":10000.0,"joiningDate":"14-05-19"}

status code is :200

POST Response

89

Status code is:200

PASSED: RegistrationSuccessful

PASSED: getempDetails

PASSED: post\_xml\_test

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

Default test

Tests run: 3, Failures: 0, Skips: 0

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

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

Default suite

Total tests run: 3, Failures: 0, Skips: 0

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

6 Handle OAuth Authorization in REST API Using REST Assured

**Step 6.1:** Creating a Maven project in Eclipse

OAuth: OAuth is a protocol that allows a user to grant limited access to their resources on one site to another, without having to expose their credentials.

* Open Eclipse.
* Click on File---> click on New--->Project.
* Select the Maven project and click on Next.
* Enter the Group id, Artifact id and click on Finish.
* Right click on Project---> src/test/java---> Package.
* Enter the package name and click Finish.
* Right click on Package---> New---> Class.
* Enter the class name and click on Finish.
* Add dependencies to pom.xml file.

**Step 6.2:** Executing the program for OAuth Authorization in REST API using REST Assured

* Write the program for OAuth Authorization in REST API using REST Assured and click on Save.

**package** Response;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

import io.restassured.response.Response;

public **class** Oauth {

@Test

public void OAuth()

{

Response resp=RestAssured.given()

.auth()

.oauth2("a2c46473d65826bb118e5ae7e260d4cf604c8e982")

.post("http://192.168.1.207:8080/api/employee/search/1597534560");

System.out.println("code" +resp.getStatusCode());

System.out.println("code" +resp.getBody().asString());

}

}

* Click on run and check the output in TestNG.

[RemoteTestNG] detected TestNG version 6.14.3

Body is:{"empId":5,"empName":"Lavanya Gowda","empAddress":"bomanahalii","mobileNumber":8970922880,"department":"testing","project":"Cervical cancer application","teamLead":"Aruna","salary":10000.0,"joiningDate":"14-05-19"}

status code is :200

PASSED: OAuth

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

Default test

Tests run: 1, Failures: 0, Skips: 0

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

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

Default suite

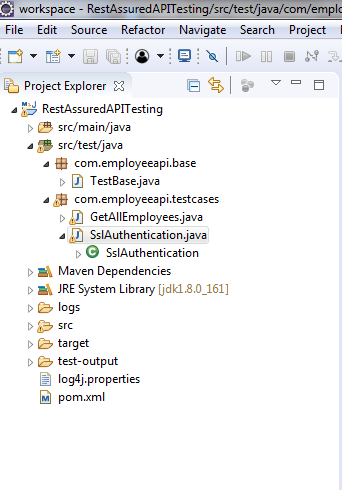
Total tests run: 1, Failures: 0, Skips: 0

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

7 SSL Authentication

**Step 7.1:** Creating aMaven Project

* The project structure looks like the screenshot below:



* Open Eclipse.
* Click on File---> click on New--->Project.
* Select the Maven project and click on Next.
* Enter the Group id, Artifact id and click on Finish.

**Step 7.2:** Updating the pom.xml file with required dependencies

* Open the pom.xml file.
* Add the given dependencies to the pom.xml file.

<dependency>

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

<artifactId>rest-assured</artifactId>

<version>3.3.0</version>

<scope>test</scope></dependency>

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>6.14.3</version>

<scope>test</scope></dependency>

**Step 7.3:** Handling SSLPeerUnverifiedException using REST Assured

* Create a package “com.employeeapi.testcases” inside the src/test/java directory.
* Create a class“SslAuthentication.java” inside the package “com.employeeapi.testcases”.
* Write the below code:

**package** com.employeeapi.testcases;

**import** org.testng.annotations.BeforeClass;

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

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

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

**import** io.restassured.RestAssured;

**public** **class** SslAuthentication {

/\*Suppose has invalid certificate and throwing an

SSLPeerUnverifiedException

\* so to handle this case we can relax certificate condition and now SSL

Exception will not come

\*

\* Do not have any proper url to test this feature\*/

**@Test**

**public** void testSsl() {

given().relaxedHTTPSValidation()

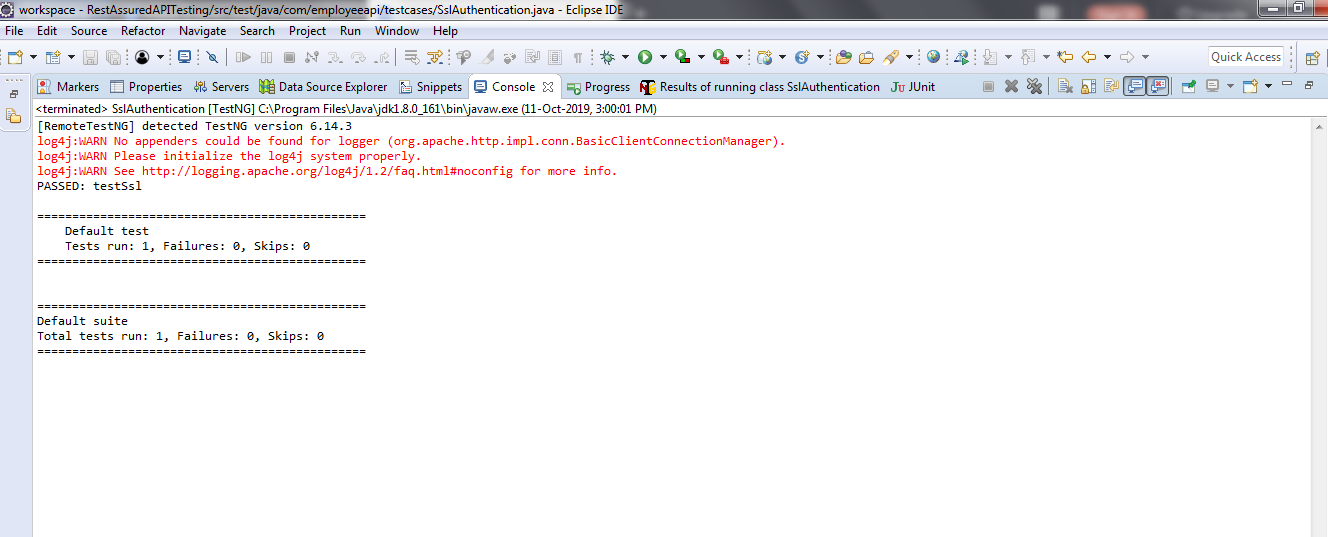
.when().get("http://www.bupa.com.au/")

.then().statusCode(200);

}

}

* Suppose the url has an invalid certificate then it will throw an SSLPeerUnverifiedException.
* To handle this case we can relax the certificate condition using relaxedHTTPSValidation() and then SSL Exception will not occur.
* Note: Do not have any proper url to test this feature.
* Right click on SSAuthentication class --> Run As --> TestNg Test.
* Verify the output from the Console:



* Note: testSsl is Passed.

8 Configure Log4j in Eclipse Maven Project

**Step 8.1:** Creating a Maven Project

* Open Eclipse.
* Click on File---> New--->Project.
* Select the Maven project and click on Next.
* Enter the Group id and Artifact id and click on Finish.

**Step 8.2:** Updating the pom.xml file with log4j dependencies

* Open the pom.xml file.
* Add the dependencies given below to pom.xml file:

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version></dependency>

**Step 8.3:** Testing the application with BasicConfigurator

* Create a package **com.employeeapi** inside the src/main/java directory.
* Create a class **LogTest.java** inside the package **com.employeeapi.**
* Write the code given below to test the logger:

**package** com.employeeapi;

**import** org.apache.log4j.BasicConfigurator;

**import** org.apache.log4j.Logger;

**public** **class** LogTest {

**static** **final** **Logger** logger =

**Logger**.getLogger(LogTest.class);

**public** **static** void main(**String**[] args)

{

//Configure logger

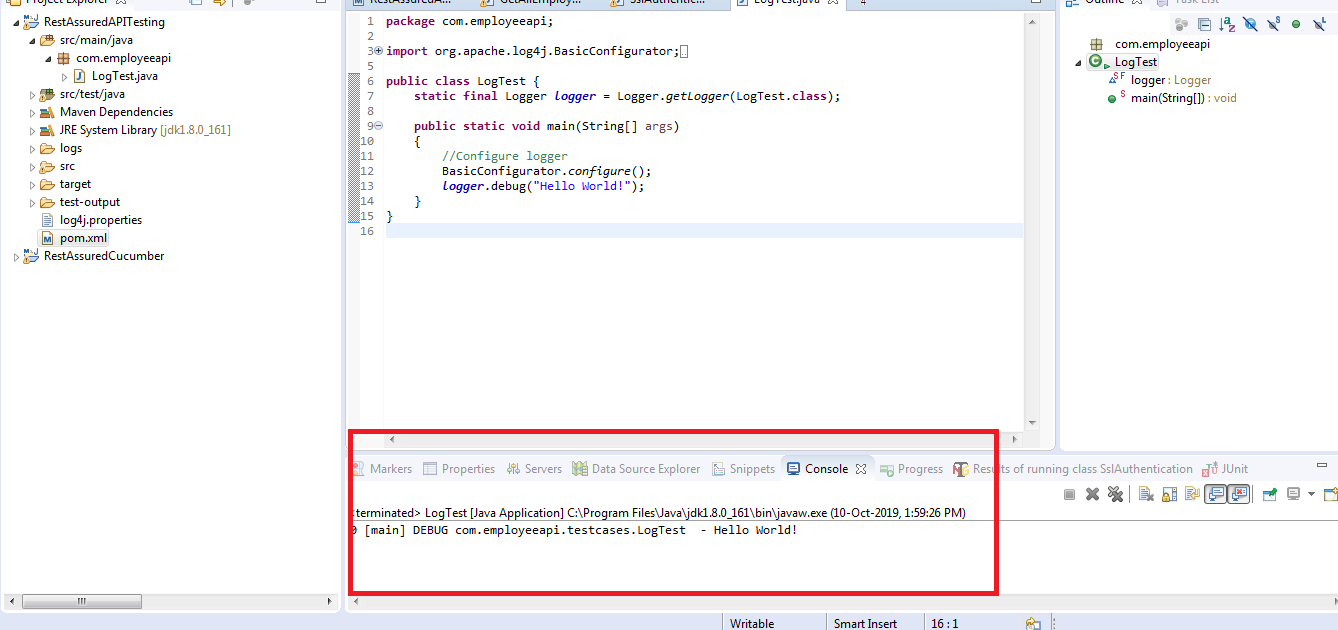
BasicConfigurator.configure();

logger.debug("Hello World!");

}

}

* Run the Java application and verify the output.



9 Generate Log on Eclipse Console Using Log4j

**Step 9.1:** Creating aMaven project

* Open Eclipse.
* Click on File---> New--->Project.
* Select the Maven project and click on Next.
* Enter the Group id and Artifact id and click on Finish.

**Step 9.2:** Updating the pom.xml file with the required dependencies

* Open the pom.xml file.
* Add the dependencies given below to pom.xml file:

<dependency>

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

<artifactId>rest-assured</artifactId>

<version>3.3.0</version>

<scope>test</scope

></dependency>

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>6.14.3</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version>

</dependency>

**Step 9.3:** Creating alog4j.properties file for ConsoleAppender

* Right click on Project --> New --> File.
* Name the file **log4j.properties** and click on Finish.
* Open log4j.properties.
* Write the code as given below:

# Root Logger option

log4j.rootLogger=INFO, file, stdout

# Direct log messages to stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.Target=System.out

log4j.appender.stdout.layout=

org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=

%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L %m%n

**Step 9.4:** Testing the REST API using REST Assured and Log4j

* Create a package **com.employeeapi.base** inside the src/test/java directory.
* Create a class **TestBase.java** inside the package **com.employeeapi.base.**
* Write the code given below:

1. EmployeesRestAPI is the name given to logger.
2. Log4j.properties is the name of the file created.

**package** com.employeeapi.base;

**import** org.apache.log4j.Level;

**import** org.apache.log4j.Logger;

**import** org.apache.log4j.PropertyConfigurator;

**import** org.testng.annotations.BeforeClass;

**import** io.restassured.response.Response;

**import** io.restassured.specification.RequestSpecification;

**public** **class** TestBase {

**public** **static** RequestSpecification httpRequest;

**public** **static** **Response** response;

**public** **String** empId="55123";

**public** **Logger** logger;

**@BeforeClass**

**public** void setup()

{

logger=**Logger**.getLogger("EmployeesRestAPI");

PropertyConfigurator.configure("Log4j.properties");

logger.setLevel(**Level**.DEBUG);

}

}

* Create a package **com.employeeapi.testcases** inside the src/test/java directory.
* Create a class **GetAllEmployees.java** inside the package **com.employeeapi.testcases.**
* Write the code given below:

**package** com.employeeapi.testcases;

**import** org.testng.Assert;

**import** org.testng.annotations.BeforeClass;

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

**import** com.employeeapi.base.TestBase;

**import** io.restassured.RestAssured;

**import** io.restassured.http.Method;

**import** io.restassured.response.ResponseBody;

**public** **class** GetAllEmployees **extends** TestBase{

**@BeforeClass**

void getAllEmployees() **throws** **InterruptedException** {

logger.info

("\*\*\*\*\*\*\*\*statrt of GetAllEmployees class\*\*\*\*\*\*\*\*\*\*");

RestAssured.baseURI=

"http://192.168.1.207:8080/api/employee/search";

httpRequest = RestAssured.given();

response =

httpRequest.request(**Method**.GET,"/8095393564");

**Thread**.sleep(5000);

}

**@Test**

void checkResponseBody() {

logger.info

("\*\*\*\*\*\*\*Inside checkResponseBody\*\*\*\*\*\*\*");

**String** responseBody=response.getBody().asString();

logger.info("Response Body ==> "+responseBody);

Assert.assertTrue(responseBody!=**null**);

}

**@Test**

void checkStatusCode() {

logger.info("\*\*\*\*Inside checkStatusCode\*\*\*\*\*\*\*");

int statusCode=response.getStatusCode();

logger.info("StatusCode ==>"+statusCode);

Assert.assertEquals(statusCode, 200);

}

**@Test**

void checkStatusLine() {

logger.info

("\*\*\*\*\*\*\*\*Inside checkStatusLine\*\*\*\*\*\*\*\*\*");

**String** statusLine=response.getStatusLine();

logger.info("StatusLine ==>"+statusLine);

Assert.assertEquals(statusLine, "HTTP/1.1 200 ");

}

**@Test**

void checkContentType() {

logger.info

("\*\*\*\*\*\*\*\*Inside checkContentType\*\*\*\*\*\*\*\*\*");

**String** contentType=response.header("Content-Type");

logger.info("Content type is ==>"+contentType);

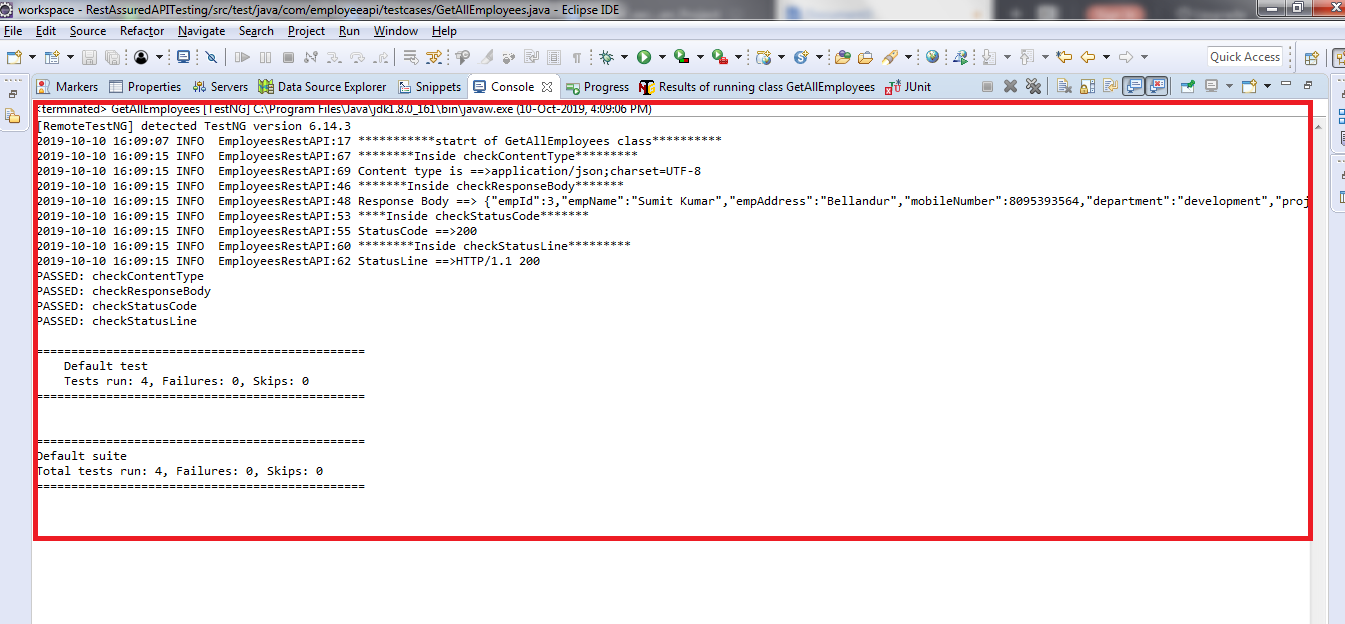
Assert.assertEquals

(contentType, "application/json; charset=UTF-8");

}

}

* Right click on GetAllEmployees class --> Run As --> TestNG Test.
* Verify the output in the console.



10 Write Log File Using Log4j

**Step 10.1:** Creating a Maven project

* Open Eclipse.
* Click on File---> New--->Project.
* Select the Maven project and click on Next.
* Enter the Group id and Artifact id and click on Finish.

**Step 10.2:** Updating the pom.xml file with the required dependencies

* Open the pom.xml file.
* Add the dependencies given below to the pom.xml file:

<dependency>

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

<artifactId>rest-assured</artifactId>

<version>3.3.0</version>

<scope>test</scope

></dependency>

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>6.14.3</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version>

</dependency>

**Step 10.3:** Creating alog4j.properties file for RollingFileAppender

* Right click on Project --> New --> File.
* Name the file as **log4j.properties** and click on Finish.
* Open log4j.properties.
* Write the code given below:

# Root Logger option

log4j.rootLogger=INFO, file, stdout

# Direct log messages to a log file

log4j.appender.**file**=org.apache.log4j.RollingFileAppender

log4j.appender.**file**.**File**=${user.dir}/logs/restAPI.log

log4j.appender.**file**.MaxFileSize=10MB

log4j.appender.**file**.MaxBackupIndex=10

log4j.appender.**file**.layout=org.apache.log4j.PatternLayout

log4j.appender.**file**.layout.ConversionPattern=

%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L %m%n

log4j.appender.**file**.Append=true

**Step 10.4:** Testing the REST API using REST Assured and log4j

* Create a package **com.employeeapi.base** inside the src/test/java directory.
* Create a class **TestBase.java** inside the package **com.employeeapi.base.**
* Write the code given below:

1. EmployeesRestAPI is the name given to logger.
2. Log4j.properties is the name of the file we created.

**package** com.employeeapi.base;

**import** org.apache.log4j.Level;

**import** org.apache.log4j.Logger;

**import** org.apache.log4j.PropertyConfigurator;

**import** org.testng.annotations.BeforeClass;

**import** io.restassured.response.Response;

**import** io.restassured.specification.RequestSpecification;

**public** **class** TestBase {

**public** **static** RequestSpecification httpRequest;

**public** **static** **Response** response;

**public** **String** empId="55123";

**public** **Logger** logger;

**@BeforeClass**

**public** void setup()

{

logger=**Logger**.getLogger("EmployeesRestAPI");

PropertyConfigurator.configure("Log4j.properties");

logger.setLevel(**Level**.DEBUG);

}

}

* Create a package **com.employeeapi.testcases** inside the src/test/java directory.
* Create a class **GetAllEmployees.java** inside the package **com.employeeapi.testcases.**
* Write the code given below:

**package** com.employeeapi.testcases;

**import** org.testng.Assert;

**import** org.testng.annotations.BeforeClass;

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

**import** com.employeeapi.base.TestBase;

**import** io.restassured.RestAssured;

**import** io.restassured.http.Method;

**import** io.restassured.response.ResponseBody;

**public** **class** GetAllEmployees **extends** TestBase{

**@BeforeClass**

void getAllEmployees() **throws** **InterruptedException** {

logger.info

("\*\*\*\*\*\*\*\*statrt of GetAllEmployees class\*\*\*\*\*\*\*\*\*\*");

RestAssured.baseURI=

"http://192.168.1.207:8080/api/employee/search";

httpRequest = RestAssured.given();

response =

httpRequest.request(**Method**.GET,"/8095393564");

**Thread**.sleep(5000);

}

**@Test**

void checkResponseBody() {

logger.info

("\*\*\*\*\*\*\*Inside checkResponseBody\*\*\*\*\*\*\*");

**String** responseBody=response.getBody().asString();

logger.info("Response Body ==> "+responseBody);

Assert.assertTrue(responseBody!=**null**);

}

**@Test**

void checkStatusCode() {

logger.info("\*\*\*\*Inside checkStatusCode\*\*\*\*\*\*\*");

int statusCode=response.getStatusCode();

logger.info("StatusCode ==>"+statusCode);

Assert.assertEquals(statusCode, 200);

}

**@Test**

void checkStatusLine() {

logger.info

("\*\*\*\*\*\*\*\*Inside checkStatusLine\*\*\*\*\*\*\*\*\*");

**String** statusLine=response.getStatusLine();

logger.info("StatusLine ==>"+statusLine);

Assert.assertEquals(statusLine, "HTTP/1.1 200 ");

}

**@Test**

void checkContentType() {

logger.info

("\*\*\*\*\*\*\*\*Inside checkContentType\*\*\*\*\*\*\*\*\*");

**String** contentType=response.header("Content-Type");

logger.info("Content type is ==>"+contentType);

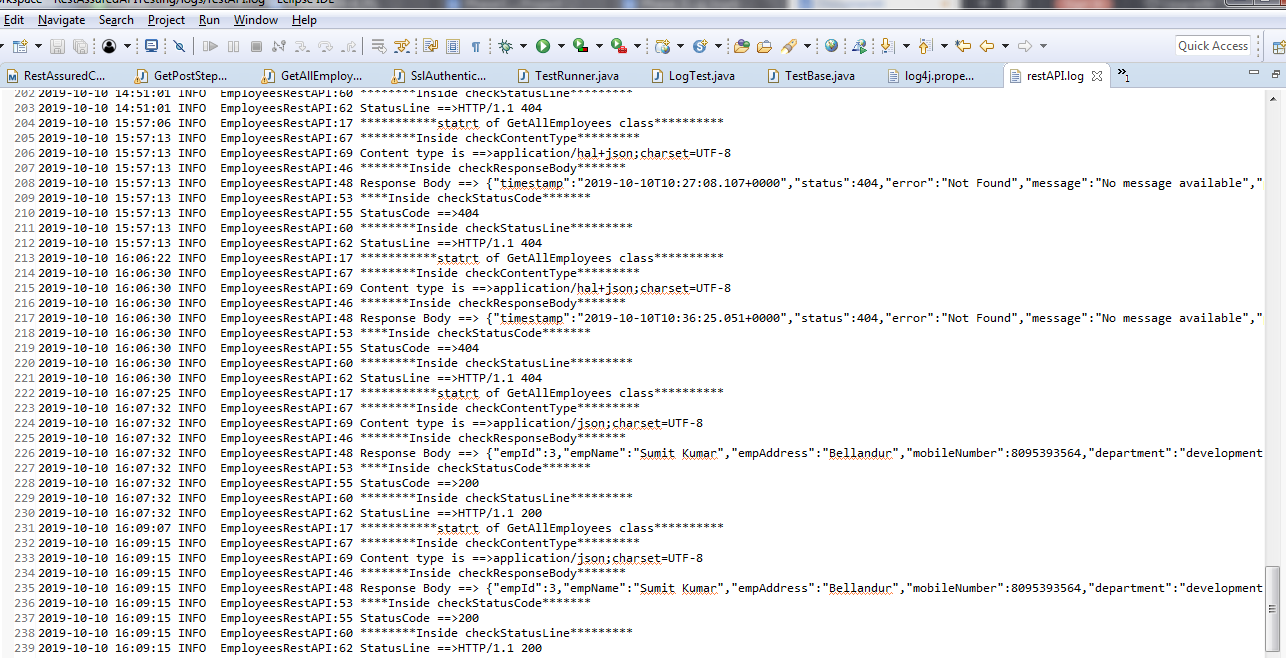
Assert.assertEquals

(contentType, "application/json; charset=UTF-8");

}

}

* Right click on GetAllEmployees class --> Run As --> TestNG Test.
* Open restAPI.log file from logs folder and verify the output:



11 OAuth, SSL, and Log4j

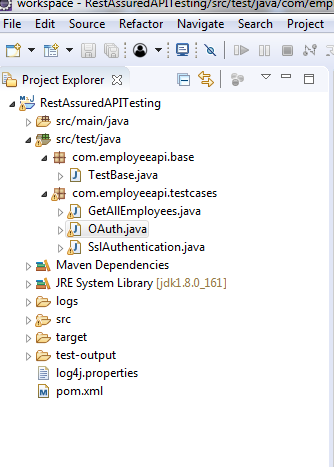
**Step 11.1:** Problem statement for OAuth, SSL, and Log4j

* **Objective**: Perform OAuth, SSL authentication for a given REST API, and track the execution using Log4j.
* **Steps involved**:

1. Create a Maven project.
2. Create class to write test cases.
3. Create a test method to handle SSLPeerUnverifiedException and to perform OAuth Authentication.
4. Include logger in the method.

**Step 11.2:** Solution for the problem statement

* The project structure looks like this:



* Open Eclipse.
* Click on file---> click on New--->Project.
* Select the Maven project and click on Next.
* Enter the Group id and Artifact id and click on Finish.
* Add the required dependencies to the pom.xml.
* Right click on Project --> New --> File.
* Name the file as “log4j.properties” and click on Finish.
* Open log4j.properties.
* Write the code shown below:

# Root Logger option

log4j.rootLogger=INFO, file, stdout

# Direct log messages to stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.Target=System.out

log4j.appender.stdout.layout=

org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=

%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L %m%n

* Right click on project---> src/test/java---> Package.
* Enter the package name(Ex: com.employeeapi.testcases) and click on Finish.
* Right click on Package---> New---> Class.
* Enter the class name(Ex: OAuth) and click on Finish.
* Write a below code inside OAuth Class:

1. EmployeesRestAPI is the name given to logger.
2. Log4j.properties is the name of the file we created.
3. relaxedHTTPSValidation() is used to handle SSLPeerUnverifiedException.
4. “a2c46473d65826bb118e5ae7e260d4cf604c8e982” is the key.

**package** com.employeeapi.testcases;

**import** org.apache.log4j.Level;

**import** org.apache.log4j.Logger;

**import** org.apache.log4j.PropertyConfigurator;

**import** org.testng.annotations.BeforeClass;

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

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

**public** **class** OAuth{

**public** **Logger** logger;

**@BeforeClass**

**public** void setup()

{

logger=**Logger**.getLogger("EmployeesRestAPI");

PropertyConfigurator.configure("Log4j.properties");

logger.setLevel(**Level**.DEBUG);

}

**@Test**

**public** void Oauth()

{

logger.info("\*\*\*\*\*\*\*\*\*\*\*statrt of OAuth

and SSLPeerUnverifiedException handling\*\*\*\*\*\*\*\*\*\*");

given().relaxedHTTPSValidation()

.auth()

.oauth2("a2c46473d65826bb118e5ae7e260d4cf604c8e982")

.post("http://192.168.1.207:8080/api/employee/search

/1597534560")

.then().statusCode(200);

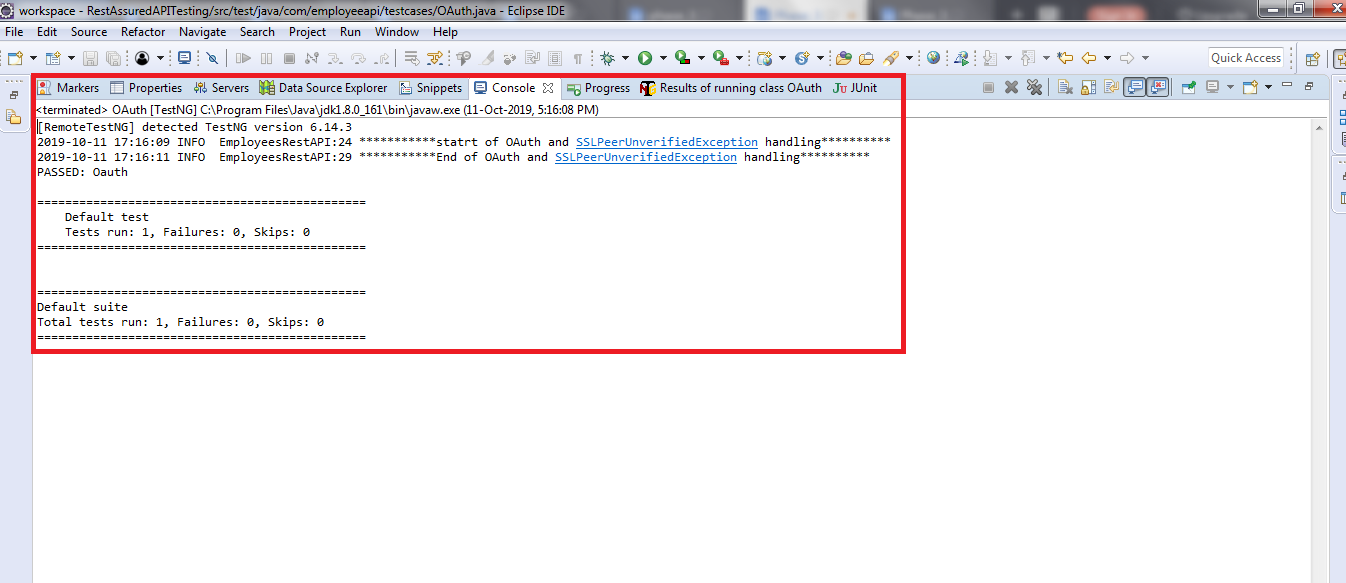
logger.info("\*\*\*\*\*\*\*\*\*\*\*End of OAuth

and SSLPeerUnverifiedException handling\*\*\*\*\*\*\*\*\*\*");

}

}

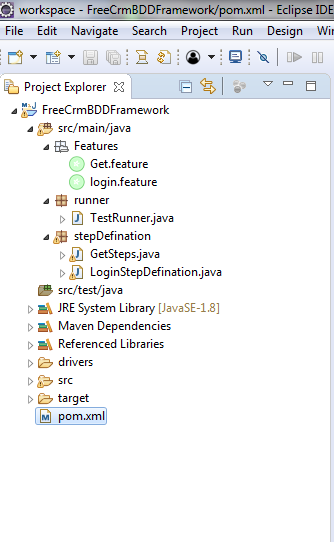
* Right click on OAuth class --> Run As --> TestNG Test and verify the output in the console:



12 BDD with REST Assured Using Cucumber

**Step 12.1:** Creating a Maven project

* Given below is the layout or example of a typical project structure:



* Open Eclipse.
* Open Eclipse Marketplace and install Natural.
* Restart Eclipse.
* Click on File---> New--->Project.
* Select the Maven project and click on Next.
* Enter the groupId and artifactId and click on Finish.

**Step 12.2:** Updating the pom.xml file with the required dependencies

* Open the pom.xml file.
* Add the dependencies given below to the pom.xml file:

<dependency>

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

<artifactId>rest-assured</artifactId>

<version>3.3.0</version>

<scope>test</scope></dependency>

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-java</artifactId>

<version>2.0.0</version></dependency>

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-junit</artifactId>

<version>3.0.0</version>

<scope>test</scope></dependency>

<dependency>

<groupId>info.cukes</groupId>

<artifactId>gherkin</artifactId>

<version>2.12.2</version>

<scope>provided</scope></dependency>

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-jvm-deps</artifactId>

<version>1.0.5</version>

<scope>provided</scope></dependency>

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-jvm</artifactId>

<version>2.0.0</version>

<type>pom</type></dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

<scope>test</scope></dependency>

**Step 12.3:** Testing an API BDD with REST Assured using Cucumber

* Create a package **Features** inside the src/main/java directory.
* Create a file **Get.feature** inside the package **Features.**
* Write the code given below:

Feature: Verify GET operation by BDD with Rest Assured **using**

Cucumber

Scenario: Verify Employee Details by Get Request

Given Perform get operation **for** employee

And Perform get **for** mobile number

Then Veriyfy the status code

* Create a package **stepDefination** inside the src/main/java directory.
* Create a class **GetSteps.java** inside the package **stepDefination.**
* Write the code given below:

**package** stepDefination;

**import** cucumber.api.java.en.And;

**import** cucumber.api.java.en.Given;

**import** cucumber.api.java.en.Then;

**import** io.restassured.http.ContentType;

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

**public** **class** GetSteps {

**@Given("^Perform get operation for employee$")**

**public** void getMethod() {

given().contentType(ContentType.JSON);

}

**@And("^Perform get for mobile number$")**

**public** void getdata() {

when()

.get("http://192.168.1.207:8080/api/employee/search/8095393564")

.then().statusCode(200);

}

**@Then("^Veriyfy the status code$")**

**public** void checkstatus() {

}

}

* Create a package **runner** inside the src/main/java directory.
* Create a class **TestRunner.java** inside the package **runner.**
* Write the code given below:

1. features= “created feature file location”
2. glue={“stepDefination”} package name of “GetSteps.java” class

**package** runner;

**import** org.junit.runner.RunWith;

**import** cucumber.api.junit.Cucumber;

**import** cucumber.api.CucumberOptions;

**@RunWith(Cucumber.class)@CucumberOptions**

**(features="C:\\Users\\Prakat-Intern\\Desktop\\Cucumber\\**

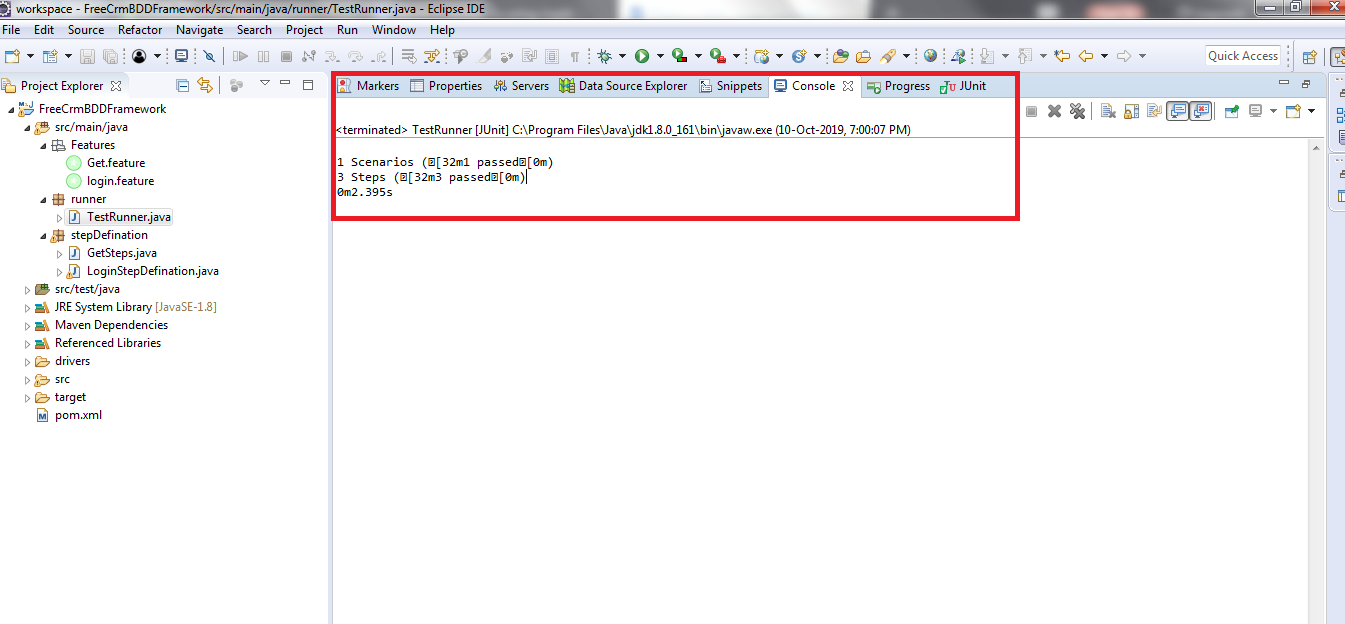
**workspace\\FreeCrmBDDFramework\\src\\main\\java\\Features**

**\\Get.feature",**

**glue= {"stepDefination"})public** **class** TestRunner {

}

* Right click on TestRunner class --> Run As --> JUnit Test.
* Verify the output in the console.



* Note: 1 Scenario and 3 steps are executed and all have passed.