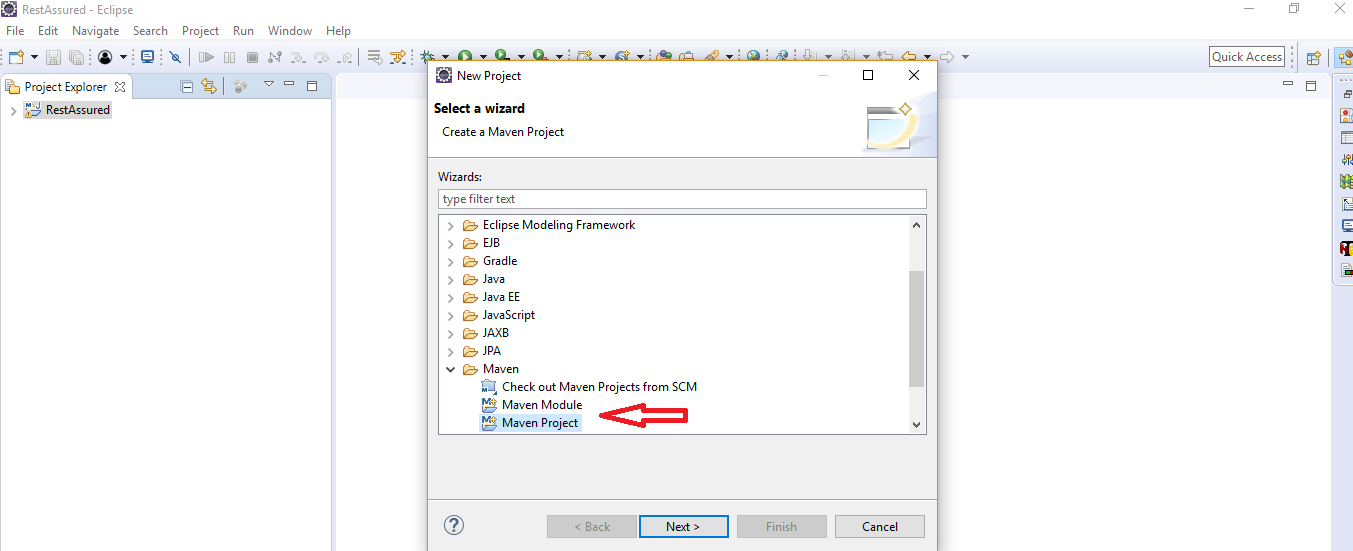
**Setup or Configure REST Assured:**

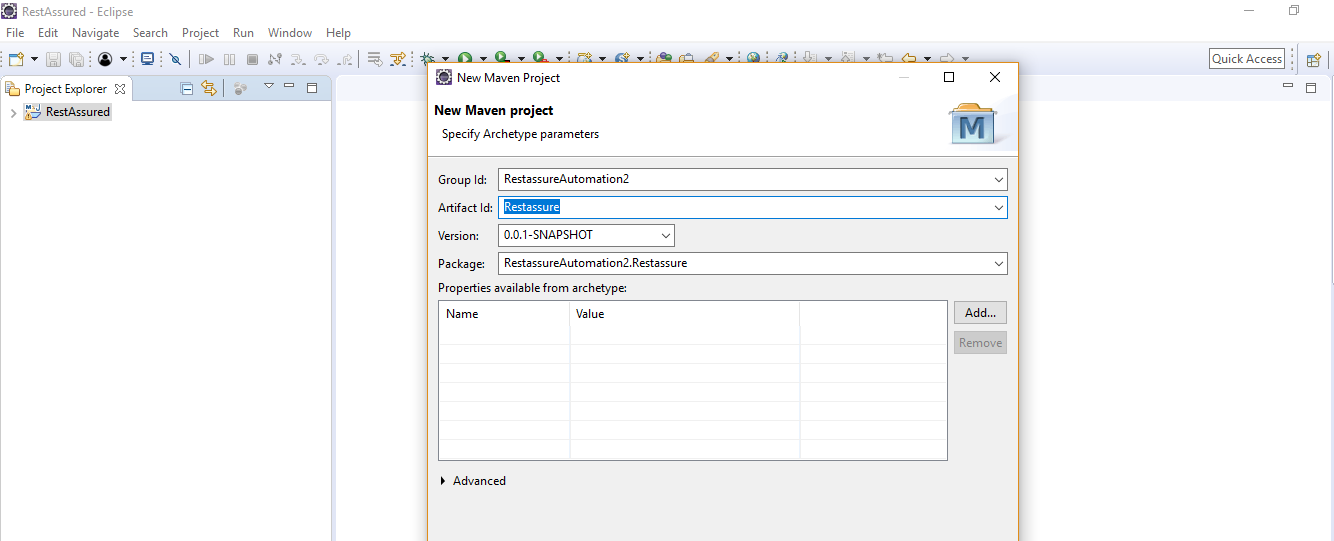
 **\***Creating a Maven project in Eclipse

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

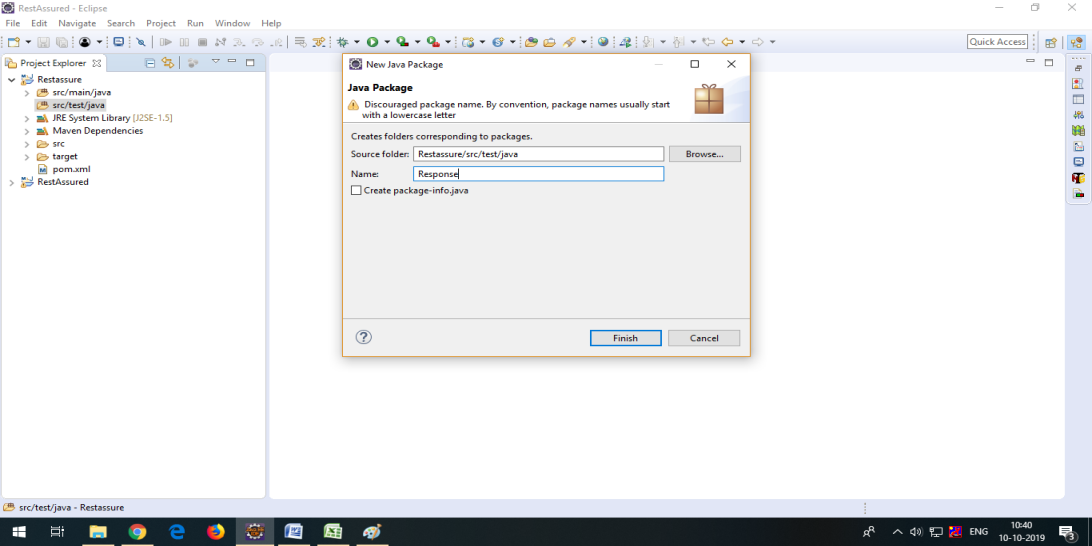
* Open Eclipse.
* Click on File---> click on New--->Project.
* Select the Maven project and click on Next.



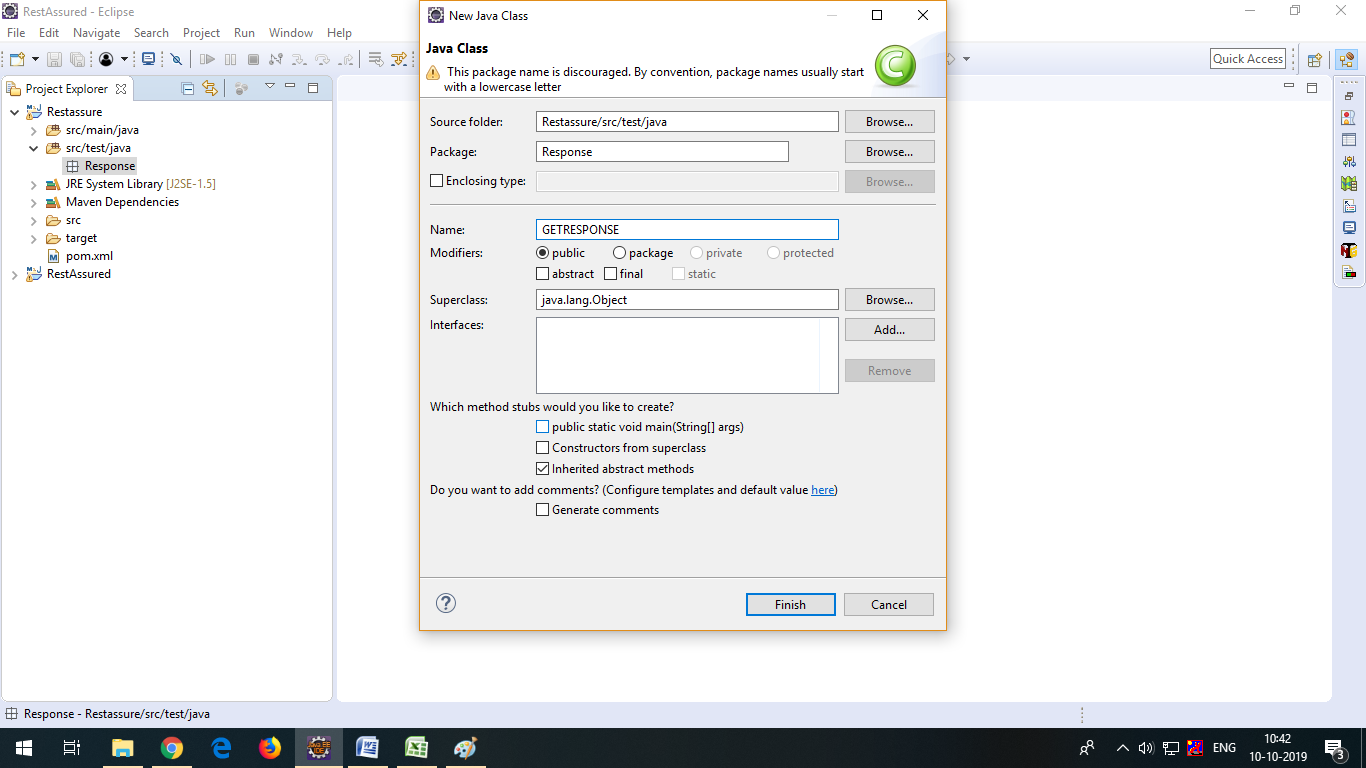
* Enter the Group id, Artifact id, and click on Finish.



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

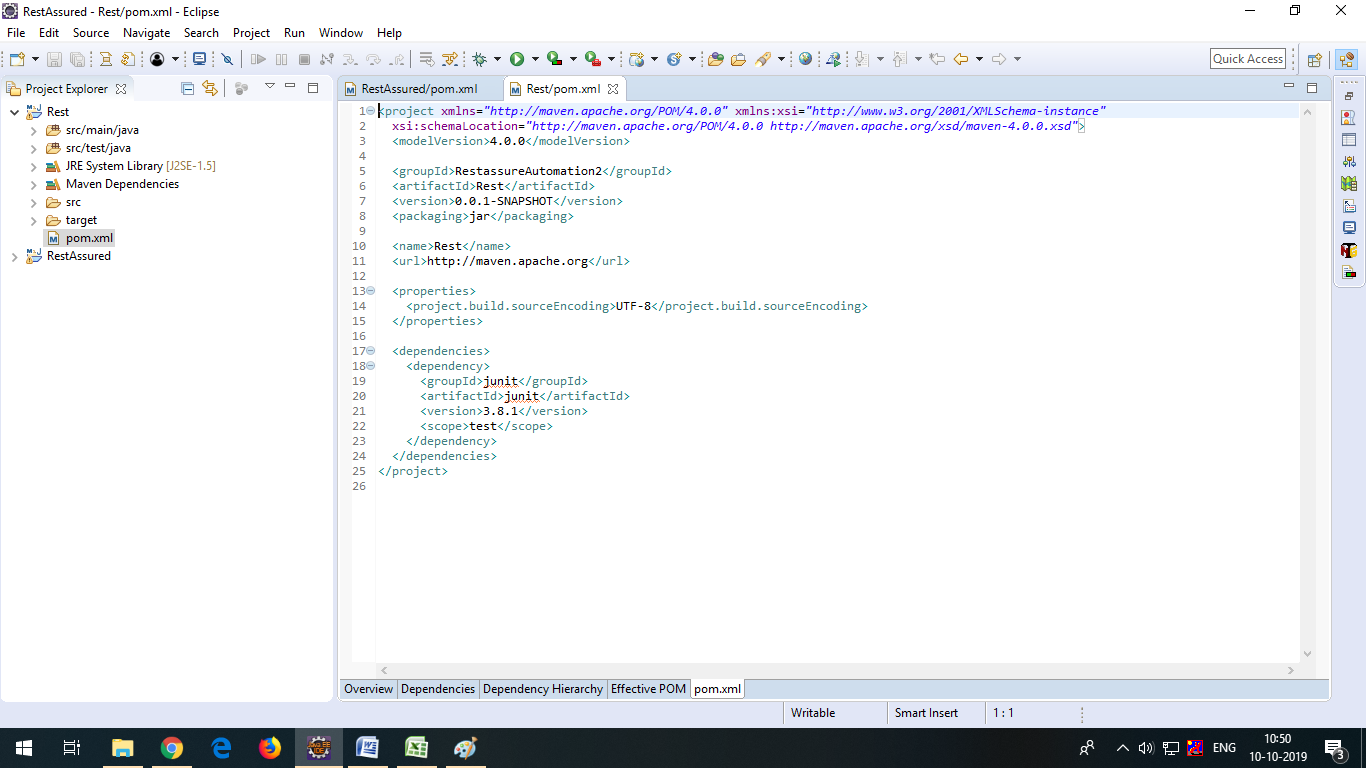


* Right click on Package---> New---> Class.
* Enter the class name and click on Finish.

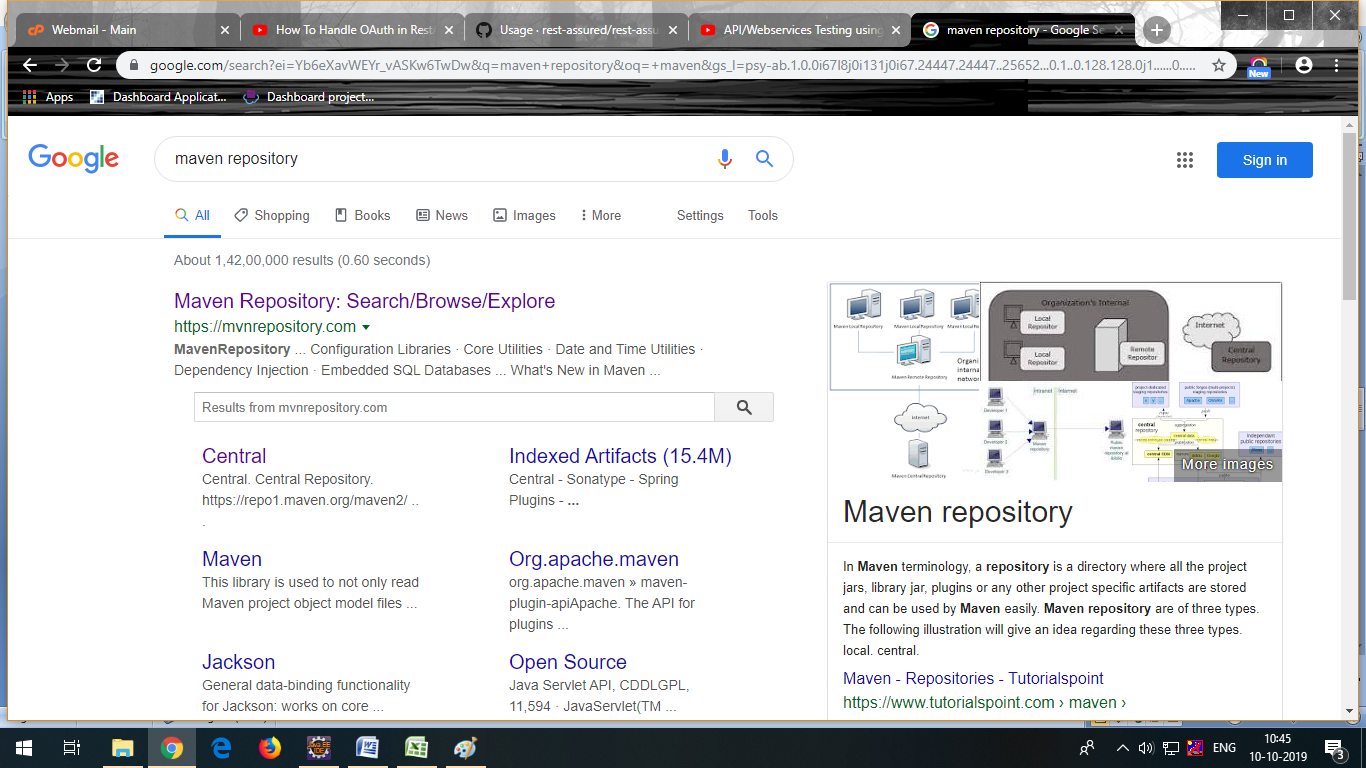


**Step 3.1.2:** Updating the pom.xml file with the latest stable REST Assured dependencies

* 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.

**To demonstrate automation of GET request and response:**

package in.amazon.RestAssuredDemo;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

public class GETDemo {

@Test

public void getServerResponse() {

RestAssured

.when()

.get("https://restcountries.com/v3.1/all")

.then()

.assertThat()

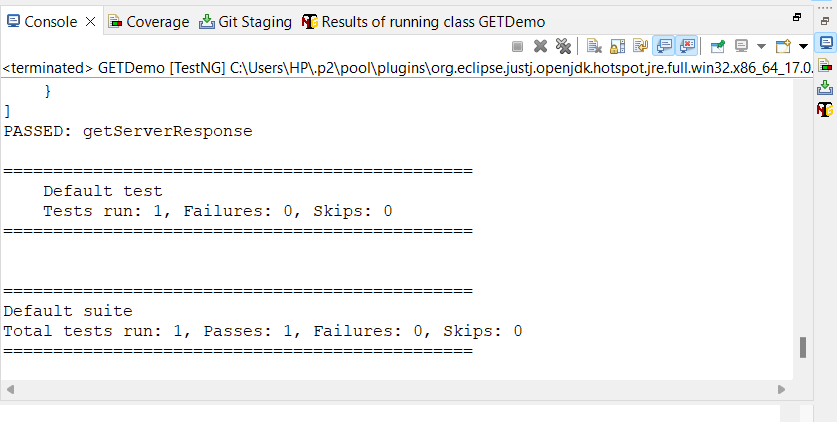
.statusCode(200)

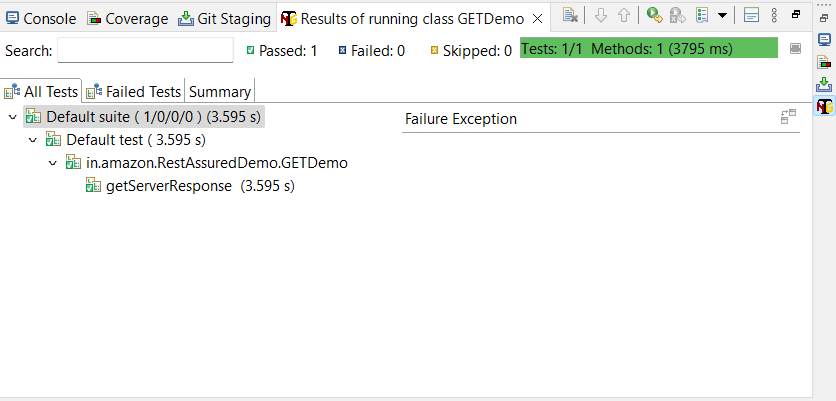
.log().all();

}

}

**Output:**

****

****

**To demonstrate automation of POST request and response:**

package in.amazon.RestAssuredDemo;

import java.util.HashMap;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

public class POSTDemo {

HashMap<String, String> map = new HashMap<String, String>();

@BeforeMethod

public void createPayLoad() {

map.put("name", "morpheus");

map.put("job", "painter");

RestAssured.baseURI = "https://reqres.in/";

RestAssured.basePath = "/api/users";

}

@Test

public void createResource() {

RestAssured

.given()

.contentType("application/json")

.body(map)

.when()

.post()

.then()

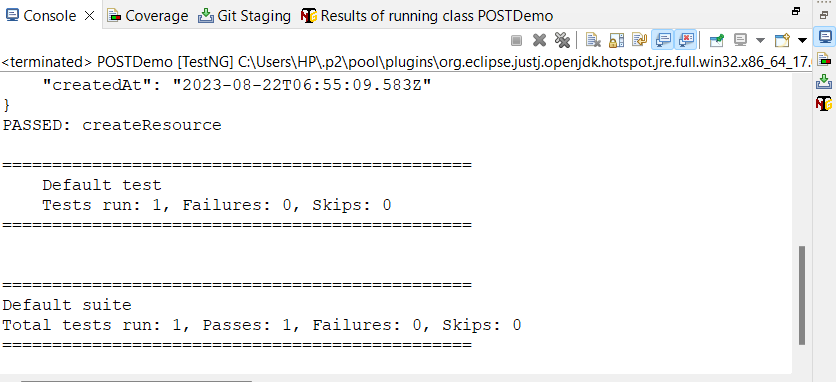
.statusCode(201)

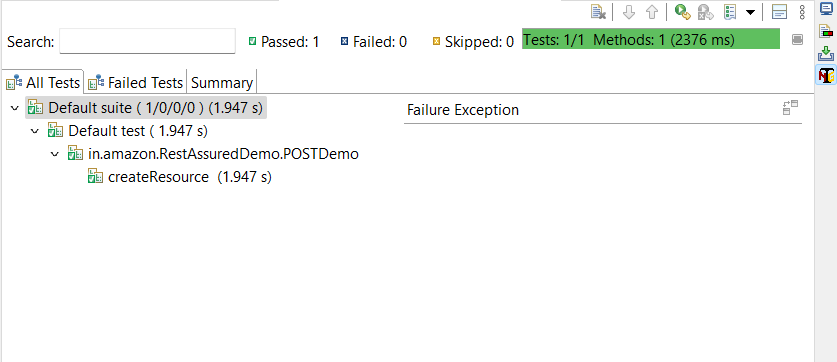
.log().all();

}

}

**Output:**





**To demonstrate automation of PUT request and response:**

package in.amazon.RestAssuredDemo;

import java.util.HashMap;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

public class PUTDemo {

HashMap<String, String> map = new HashMap<String, String>();

@BeforeMethod

public void createPayLoad() {

map.put("name", "morpheus");

map.put("job", "programmer");

RestAssured.baseURI = "https://reqres.in/";

RestAssured.basePath = "/api/users/687";

}

@Test

public void updateResource() {

RestAssured

.given()

.contentType("application/json")

.body(map)

.when()

.put()

.then()

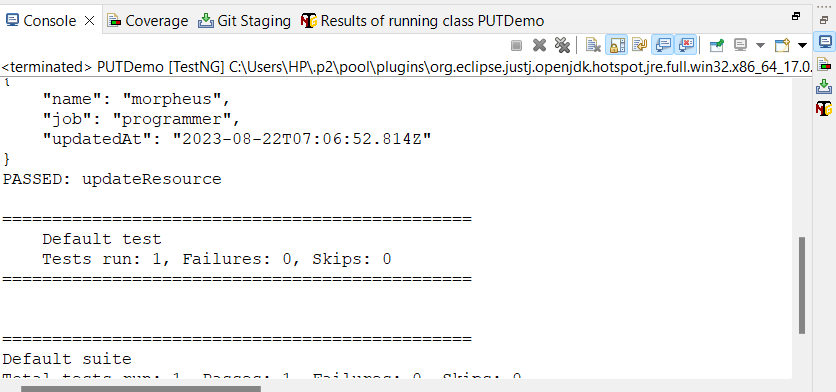
.statusCode(200)

.log().all();

}

}

**Output:**



**To demonstrate automation of DELETE request and response:**

package in.amazon.RestAssuredDemo;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

public class DELETEDemo {

@Test

public void deleteResource() {

RestAssured.baseURI = "https://reqres.in/";

RestAssured.basePath = "/api/users/687";

RestAssured

.when()

.delete()

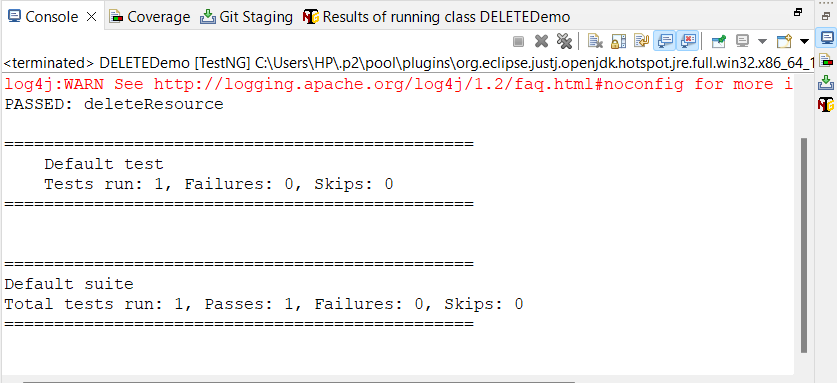
.then()

.statusCode(204);

}

}

**Output:**

****

**Program to 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:**

package in.amazon.RestAssuredDemo;

import java.util.HashMap;

import java.util.UUID;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

public class BearerTokenDemo {

HashMap<String, String> map = new HashMap<String, String>();

UUID uuid = UUID.randomUUID();

@BeforeMethod

public void createPayLoad() {

map.put("name", "Harsha");

map.put("email", uuid+"@gmail.com");

map.put("gender", "female");

map.put("status", "active");

RestAssured.baseURI = "https://gorest.co.in/";

RestAssured.basePath = "public/v2/users";

}

@Test

public void createResource() {

RestAssured

.given()

.contentType("application/json")

.body(map)

.header("Authorization", "Bearer 2091c363f152559dd329d6825054e9065f12f48265013b74e33a9490a727dcfd")

.when()

.post()

.then()

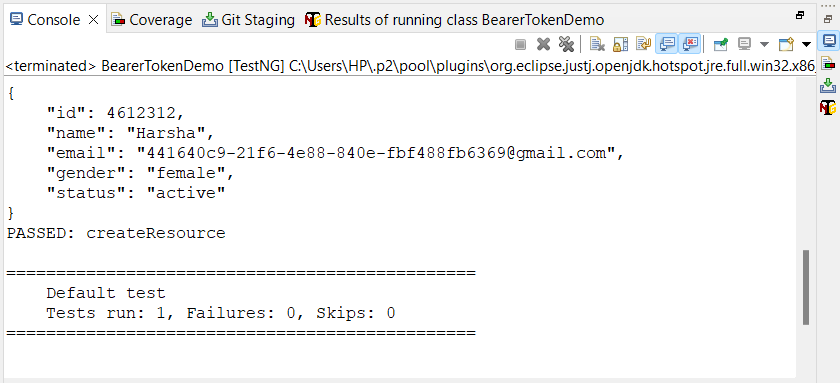
.statusCode(201)

.log().all();

}

}

**Output:**

****

**To demonstrate GET, POST, XML, and JSON in REST Assured:**

package in.amazon.RestAssuredDemo;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

public class GETDemoBearerToken {

@Test

public void verifyResource() {

RestAssured

.given()

.contentType("application/json")

.header("Authorization", "Bearer 2091c363f152559dd329d6825054e9065f12f48265013b74e33a9490a727dcfd")

.when()

.get("https://gorest.co.in/public/v2/users/4495061")

.then()

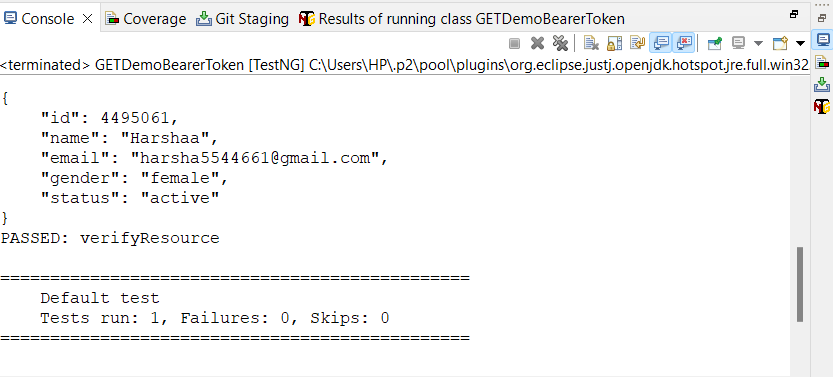
.statusCode(200)

.log().all();

}

}

**Output:**

****

**For PUT:**

package in.amazon.RestAssuredDemo;

import java.util.HashMap;

import org.junit.Assert;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

import io.restassured.path.json.JsonPath;

import io.restassured.response.Response;

public class PUTDemoBearerToken {

HashMap<String, String> map = new HashMap<>();

@BeforeMethod

public void createPayLoad() {

map.put("name", "Harshaa");

map.put("email", "harsha5544661@gmail.com");

map.put("gender", "female");

map.put("status", "active");

RestAssured.baseURI = "https://gorest.co.in/";

RestAssured.basePath = "public/v2/users/4495061";

}

@Test

public void updateResource() {

Response response = RestAssured

.given()

.contentType("application/json")

.header("Authorization", "Bearer 2091c363f152559dd329d6825054e9065f12f48265013b74e33a9490a727dcfd")

.body(map)

.when()

.put()

.then()

.extract().response();

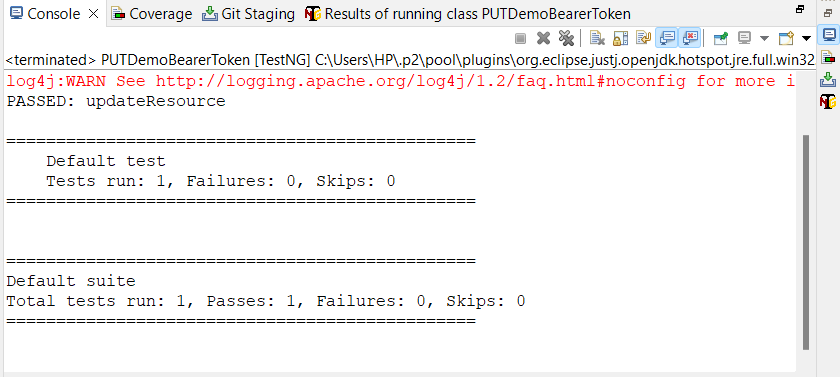
JsonPath jsonpath = response.jsonPath();

Assert.assertTrue(jsonpath.get("name").toString().equals("Harshaa"));

}

}

**Output:**

****

**To demonstrate OAuth authorization in REST API using REST assured:**

package in.amazon.RestAssuredDemo;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

public class APIKeyDemo {

@Test

public void getWeatherInfo() {

RestAssured

.given()

.param("q", "chennai")

.param("appid", "40b0035f4392615c2a098e924e56d548")

.when()

.get("https://api.openweathermap.org/data/2.5/weather")

.then()

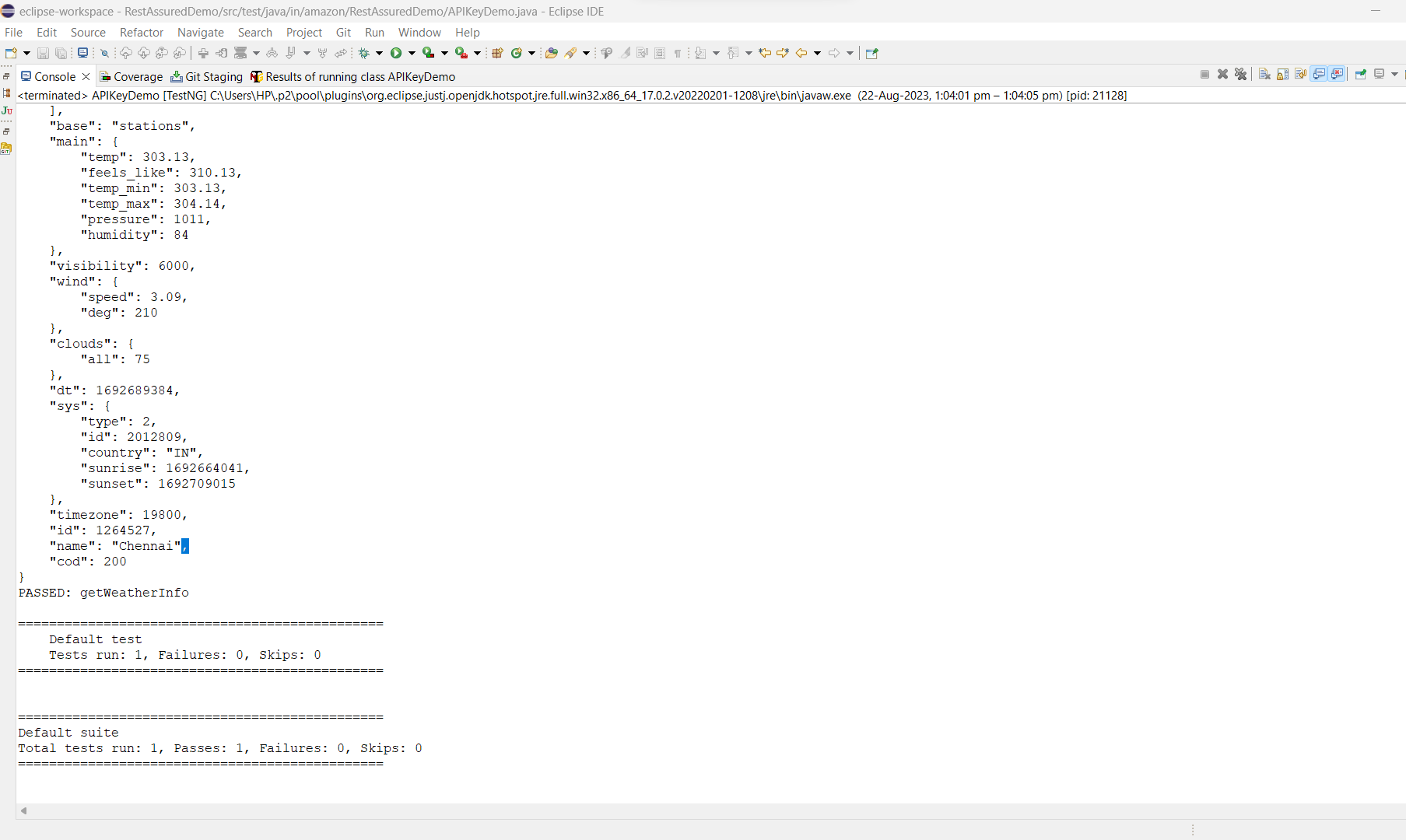
.statusCode(200)

.log().all();

}

}

**Output:**



package in.amazon.RestAssuredDemo;

import org.junit.Assert;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

import io.restassured.path.json.JsonPath;

import io.restassured.response.Response;

public class VerifyCurrencyName {

@Test

public void verifyCurrencyName() {

Response response = RestAssured

.when()

.get("https://restcountries.com/v3.1/name/japan")

.then()

.extract().response();

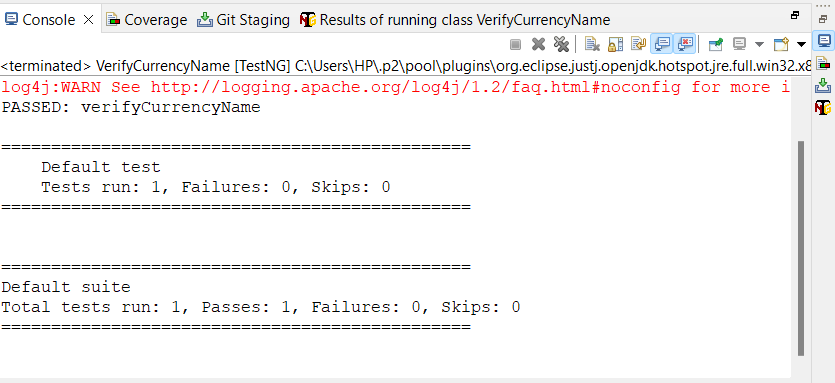
JsonPath jsonPath = response.jsonPath();

Assert.assertTrue(jsonPath.get("[0].currencies.JPY.name").toString().equals("Japanese yen"));

}

}

**Output:**

****

**To configure Log4j, generate logs on Eclipse Console, write log file using Log4j in Eclipse Maven project:**

**Create\_update\_delete\_Resource.java**

package in.amazon.RestAssuredDemo;

import java.util.HashMap;

import java.util.UUID;

import org.junit.Assert;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

import static org.hamcrest.Matchers.is;

import io.restassured.RestAssured;

import io.restassured.path.json.JsonPath;

import io.restassured.response.Response;

public class Create\_Update\_Delete\_Resource extends TestBase {

HashMap<String, String> map = new HashMap<String, String>();

UUID uuid = UUID.randomUUID();

int id;

Response response;

JsonPath jsonPath;

@Test(priority=0)

public void createPayload() {

map.put("name", "Harsha");

map.put("email", uuid+"@gmail.com");

map.put("gender", "female");

map.put("status", "active");

RestAssured.baseURI = "https://gorest.co.in/";

RestAssured.basePath = "public/v2/users";

logger.info("Payload created");

}

@Test(priority=1)

public void createResource() {

response = RestAssured

.given()

.contentType("application/json")

.body(map)

.header("Authorization", "Bearer 2091c363f152559dd329d6825054e9065f12f48265013b74e33a9490a727dcfd")

.when()

.post()

.then()

.log().all()

.extract().response();

logger.info("Resource created successfully");

jsonPath = response.jsonPath();

id = jsonPath.getInt("id");

System.out.println(id);

}

@Test(priority=2)

public void verifyResource() {

System.out.println("Id = " + id);

RestAssured

.given()

.contentType("application/json")

.header("Authorization", "Bearer 2091c363f152559dd329d6825054e9065f12f48265013b74e33a9490a727dcfd")

.when()

.get("https://gorest.co.in/public/v2/users/4495061")

.then()

.statusCode(200)

.log().all();

logger.info("Resource verified");

Assert.assertTrue(jsonPath.get("name").equals("Harsha"));

}

@Test(priority=3)

public void updateResource() {

map.put("name", "Harshaa");

map.put("email", uuid+"@gmail.com");

map.put("gender", "female");

map.put("status", "active");

RestAssured.baseURI = "https://gorest.co.in/";

RestAssured.basePath = "public/v2/users/" + id;

RestAssured

.given()

.contentType("application/json")

.header("Authorization", "Bearer 2091c363f152559dd329d6825054e9065f12f48265013b74e33a9490a727dcfd")

.body(map)

.when()

.put()

.then()

.statusCode(200)

.log().all()

.assertThat()

.body("name", is("Harshaa"));

logger.info("Resource Updated");

}

@Test(priority=4)

public void deleteResource() {

RestAssured.baseURI = "https://gorest.co.in/";

RestAssured.basePath = "public/v2/users/" + id;

RestAssured

.given()

.contentType("application/json")

.header("Authorization", "Bearer 2091c363f152559dd329d6825054e9065f12f48265013b74e33a9490a727dcfd")

.when()

.delete()

.then()

.statusCode(204);

logger.info("Resource Deleted");

}

}

**TestBase.java:**

package in.amazon.RestAssuredDemo;

import org.apache.log4j.Logger;

import org.apache.log4j.Level;

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 Logger logger;

@BeforeClass

public void setUp() {

logger = Logger.getLogger("LogDemo");

PropertyConfigurator.configure("log4j.properties");

logger.setLevel(Level.DEBUG);

}

}

**log4j.properties:**

log4j.rootLogger=ERROR,stdout

log4j.logger.com.endeca=INFO

log4j.logger.LogDemo=DEBUG, dest1

# Logger for crawl metrics

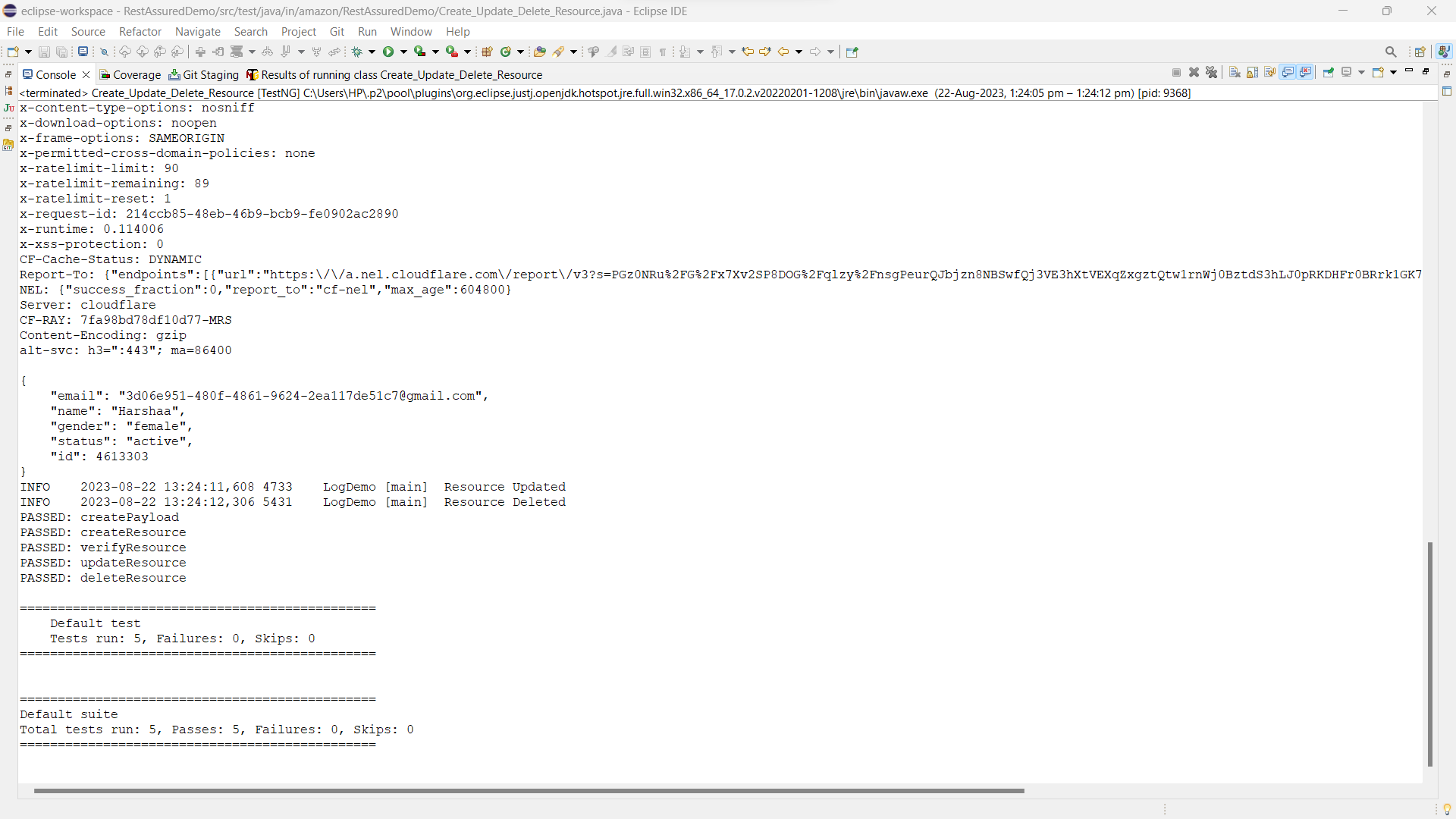
log4j.logger.com.endeca.itl.web.metrics=INFO

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

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%p\t%d{ISO8601}\t%r\t%c\t[%t]\t%m%n

**Output:**

****