**OAuth, SSL, and Log4j**

**1. Handle OAuth Authorization in REST API Using REST Assured**

**Source Code**

package RestAssuredScripts;

import org.testng.Assert;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

import io.restassured.response.Response;

public class Script09\_oauth2 {

String accessToken;

@Test(priority='1')

public void GetAccessToekn()

{

String clientID = "AYezS7RdFg0F-WXQRYDcu7tKOXn3t00y9yQwfgfHFmg4JPpILcGo8DONcBUtedp2tfPyECCK9nlJ-Vlb";

String clinetsecret = "EBvWXywcnIPNNwZhp46YERSLWglI0PG4DmBNzhgcgqCzcvLHWNFBrLD1uanC4zdnomE2lxwbrxViJCYN";

Response res = RestAssured.given()

.baseUri("https://api-m.sandbox.paypal.com")

.basePath("/v1/oauth2/token")

.auth().preemptive().basic(clientID, clinetsecret)

.param("grant\_type", "client\_credentials")

.when().post();

Assert.assertEquals(res.statusCode(), 200);

accessToken = res.getBody().path("access\_token");

System.out.println(accessToken);

}

@Test(priority='2')

public void paypalTestAPI()

{

Response res = RestAssured.given()

.baseUri("https://api-m.sandbox.paypal.com")

.basePath("/v1/invoicing/invoices")

.queryParam("page", "3")

.queryParam("page\_size", "4")

.queryParam("total\_count\_required", true)

.auth().oauth2(accessToken)

.when().get();

res.prettyPrint();

}

}

**Output**

****

**2. SSL Authentication**

**Source Code**

**package** RestAssuredScripts;

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

**import** io.restassured.RestAssured;

**public** **class** Script10SSL {

@Test

**public** **void** SSL\_authentication()

{

RestAssured.*given*().

relaxedHTTPSValidation()

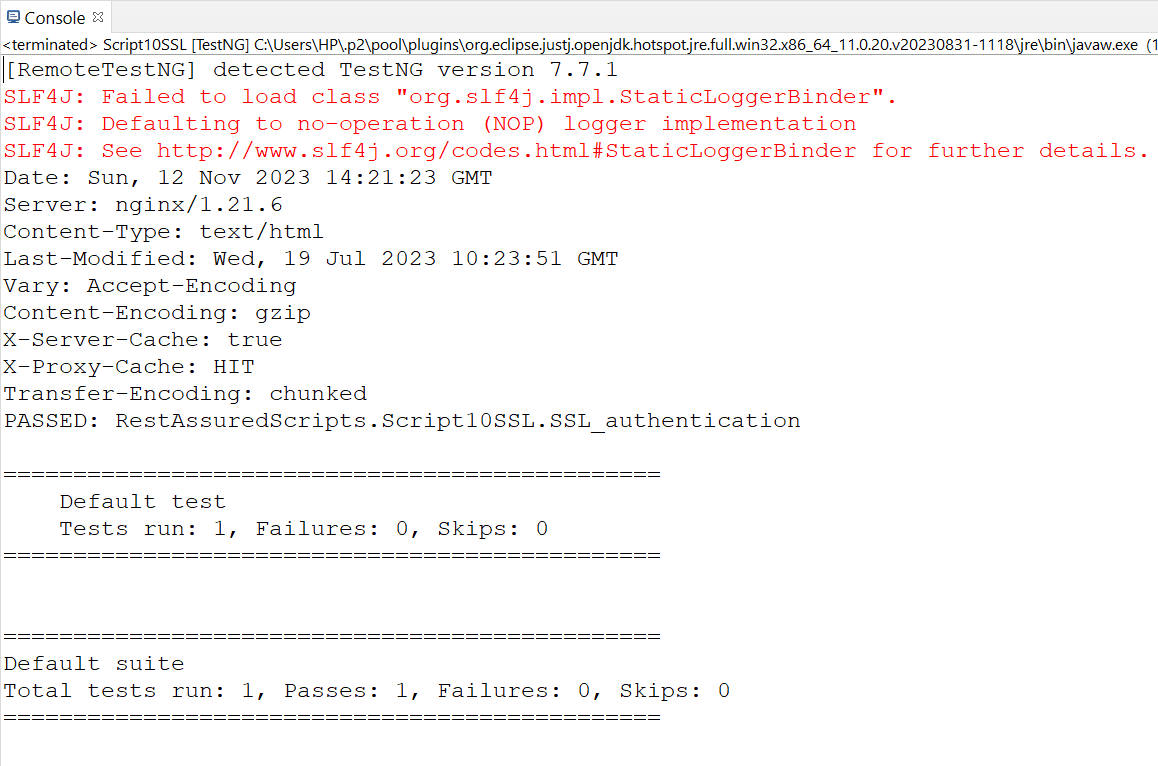
.when().get("https://testautomasi.com/Home/")

.then().log().headers();

}

}

**Output**

****

**3. Write Log File Using Log4j**

**1. Adding dependencies to pom.xml**

<dependency>

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

<artifactId>log4j-api</artifactId>

<version>2.14.1</version>

</dependency>

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

<dependency>

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

<artifactId>log4j-core</artifactId>

<version>2.14.1</version>

</dependency>

**2. Adding log4j.properties file into src/main/resource folder**

name=PropertiesConfig

property.filename=logs

appenders = console,file

appender.console.type = Console

appender.console.name = STDOUT

appender.console.layout.type = PatternLayout

appender.console.layout.pattern=[%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %c**{1}** - %msg%n

appender.file.type = File

appender.file.name = LOGFILE

appender.file.fileName=${filename}/Mylogs.log

appender.file.layout.type=PatternLayout

appender.file.layout.pattern=[%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %c**{1}** - %msg%n

loggers=file

logger.file.name=logspractice

logger.file.level = debug

logger.file.appenderRefs = file

logger.file.appenderRef.file.ref = LOGFILE

rootLogger.level = debug

rootLogger.appenderRefs = stdout

rootLogger.appenderRef.stdout.ref = STDOUT

**3. Source Code**

**package** RestAssuredScripts;

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

**import** org.apache.logging.log4j.LogManager;

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

**public** **class** Log4jDemo {

Logger logger = LogManager.*getLogger*(LessonEndProject.**class**);

@Test

**public** **void** method1()

{

System.***out***.println("this is method1");

logger.trace(" This is trace method");

logger.info(" this is information message");

logger.error(" this is an error message");

logger.fatal(" this is a fatal messgae");

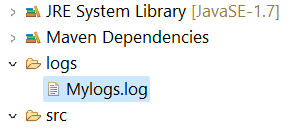
logger.warn(" this is a warning message");

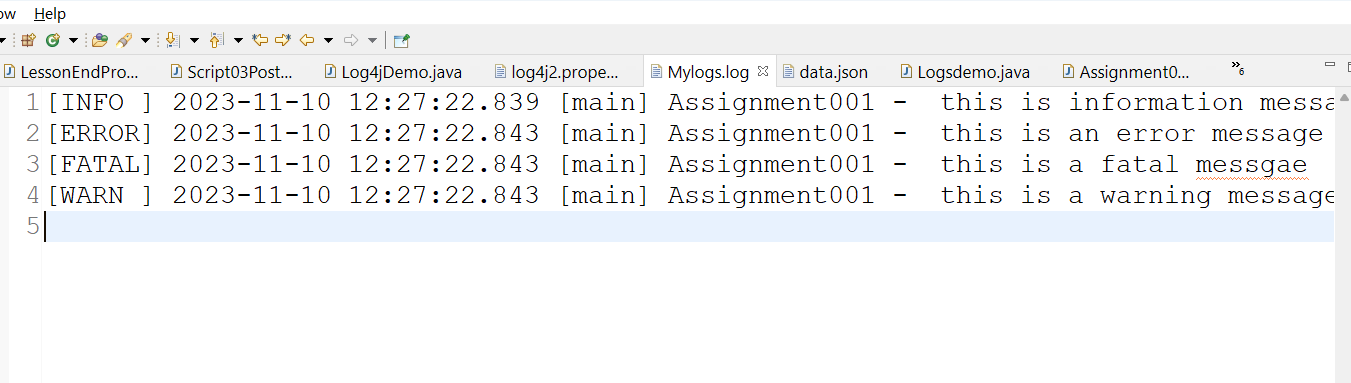
System.***out***.println("this is end of method1");

}

}

**Output:**

****

****