**إيه هو Rest Assured؟**

* **Rest Assured** هو **Java library** (مكتبة جافا) بتُستخدم لاختبار **REST APIs**.
* **سهل الاستخدام** – syntax بتاعه readable وقريب من اللغة الطبيعية./ **BDD Style** – بيدعم كتابة التستات بطريقة Given-When-Then زي Cucumber.
* **Integration** – بيتكامل مع **JUnit / TestNG / Maven / Gradle**.
* **Built-in JSON & XML support** – مش محتاج libraries خارجية عشان تتعامل مع responses.
* بيسهّل كتابة **automated API tests** من غير ما تحتاج تكتب كود طويل ومعقد. / بديل للـ Postman في الـ Automation
* معمول مخصوص عشان يعمل requests زي: (GET, POST, PUT, DELETE...) ويتحقق من **response** اللي راجع (status code, headers, body).
* تختبر الـ **functional behavior** بتاع الـ API (الـ response صح ولا لأ).
* تتحقق من **data validation** (القيمة اللي راجعة زي المتوقع). / سهولة التحقق (Assertions)
* تستخدمها في **integration testing** بين الـ frontend والـ backend.



## 1) GET Request بيستخدم لاسترجاع بيانات

@Test

public void testGET() {

given().baseUri("https://reqres.in/api")

.when().get("/users/2")

.then().statusCode(200).body("data.id", equalTo(2));}

## 🔹 2) POST Request بيستخدم لإضافة **Resource جديد** (User جديد مثلًا).

@Test

public void testPOST() {

String requestBody = "{ \"name\": \"Dina\", \"job\": \"QA Engineer\" }";

given().baseUri("https://reqres.in/api").header("Content-Type", "application/json".body(requestBody)

.when().post("/users")

.then().statusCode(201).body("name", equalTo("Dina"));}

## 🔹 3) PUT Request بيستخدم لـ **تحديث كامل** للـ Resource.

@Test

public void testPUT() {

String requestBody = "{ \"name\": \"Dina\", \"job\": \"Senior QA Engineer\" }";

given().baseUri("https://reqres.in/api").header("Content-Type", "application/json").body(requestBody)

.when().put("/users/2")

.then().statusCode(200).body("job", equalTo("Senior QA Engineer"));}

## 🔹 4) PATCH Request بيستخدم لـ **تحديث جزئي** للـ Resource (مثلاً تحديث الوظيفة بس).

@Test

public void testPATCH() {

String requestBody = "{ \"job\": \"Lead QA Engineer\" }";

given().baseUri("https://reqres.in/api").header("Content-Type", "application/json").body(requestBody)

.when().patch("/users/2")

.then().statusCode(200).body("job", equalTo("Lead QA Engineer"));}

## 🔹 5) DELETE Request 📌 بيستخدم لـ **مسح Resource**.

@Test

public void testDELETE() {

given().baseUri("https://reqres.in/api")

.when().delete("/users/2")

.then().statusCode(204); // No Content}

|  |
| --- |
| Given().baseuri(url).header(pp ).body(……………………….)  When(). “method(path/record except post)”  Then().statuscode(200/300400/500).body(all except delete) |

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

**How to make a demo project in details:**

Eclipse-file-new-other-maven project (create a simple project)

If I want to see only the project im working on :

Left window 3pts-select working set-new-java-name &add then select

<https://mvnrepository.com/> >>copy&paste Dependences in pom.xml for :

(restassured/testing/json simple/json schema validator from restassured/apache comments)

<https://rest-assured.io/> >>docs-getting started

<https://testng.org/> >>  
navigate to eclipse plugin

scroll down to latest release and copy url then paste it in eclipse  
open eclipse-help-install new software-paste url-add-name it TestNG and paste url again

select TestNG and click next  
then check by clicking (already installed) and search TestNG  
then eclipse help-about eclipse ide and click (installation details) and u will see TestNG



some abbreviation:

syso ctrl space=System.***out***.println();

Ctrl shift o to organize imports

Ctrl+ to increase the font

Ctrl shift f to format code

<https://reqres.in/> >>The site we will work on add an API key so we create the basetest class:

@BeforeSuite

public void setup() {

RequestSpecification reqSpec = new RequestSpecBuilder()

.setBaseUri("https://reqres.in/api") .addHeader("x-api-key", "reqres-free-v1") .build();

RestAssured.requestSpecification = reqSpec; } }

Project-src/test/java-add demo package & tests package

Under tests package – add classes

<https://github.com/rest-assured/rest-assured/wiki/Usage> >> rest assured usage examples

<https://jsonpathfinder.com/>>> if u have complex json , use a tool to find the json path

(click beautify / data-id1 / copy the path data[1].id )

(ex: .body(“data[1].id”,equalTo(8));)

<https://reqres.in/api/users?page=2> >> db to try get method

public void testGet() {

given().

when().get("/users?page=2")

.then().statusCode(200).body("data[4].first\_name", equalTo("George")) .body("data.first\_name", hasItems("George", "Rachel"));

<https://reqres.in/api/users> >> db to try post method

public void testPost() {

Map<String, Object> map = new HashMap<>(); لحفظ البيانات هنا body كـ map (key/value) بنجهز

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

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

JSONObject request = new JSONObject(map); تحويل الـ Map لـ JSON object

given().contentType(ContentType.JSON).accept(ContentType.JSON).body(request.toJSONString())

.when() .post("/users")

.then().statusCode(201).log().all();

}

}

<https://reqres.in/api/users?page=2> >> db to try put&patch method

public void testPut() {

JSONObject request = new JSONObject();

request.put("name", "Raghav");

request.put("job", "Teacher");

given().contentType(ContentType.JSON).accept(ContentType.JSON).body(request.toJSONString())

.when().put("/users/2")

.then().statusCode(200).log().all();

}

public void testPatch() {

JSONObject request = new JSONObject();

request.put("name", "Raghav");

request.put("job", "Teacher");

given().contentType(ContentType.JSON).accept(ContentType.JSON).body(request.toJSONString())

.when().patch("/users/2")

.then().statusCode(200).log().all();

}

public void testDelete() {

when().delete("/users/2")

.then().statusCode(204).log().all();

}

**How to create fake rest API:**

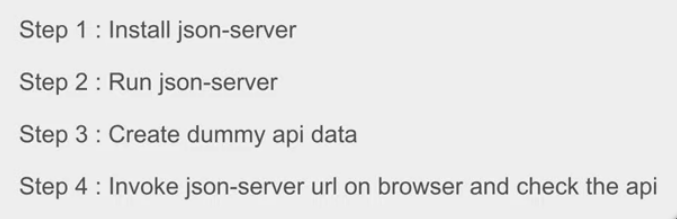
Open cmd & make sure u r in your project folder path

<https://nodejs.org/en/download> >> to download node

<https://github.com/typicode/json-server> >> to install json server

then create a file “db.json” and put ur data in it and save it in project folder

<http://localhost:3000/> >> ur local server



### خطوات تشغيل JSON Server بالملف db.json

1. **تأكدي إن عندك Node.js و npm** npm –version / node.js -version
2. **ثبتي json-server (جلوبي أو لوكال)**استخدمي npm install -g json-server أو شغلتيه مباشرة بـ: npx json-server
3. **جهزتي ملف البيانات db.json** G:\SenSe\Imp\TESTING\SS\_TESTING\API\_Restassured\db.json
4. **فتحتِ Git Bash / CMD وانتقلتِ للفولدر الصحيح**  cd G:/SenSe/Imp/TESTING/SS\_TESTING/API\_Restassured
5. **شغلتي السيرفر على الملف db.json** استخدمتي: npx json-server --watch db.json --port 3000
6. **النتيجة:** السيرفر اشتغل على:👉 <http://localhost:3000> وأي تعديل في db.json دلوقتي هيتعكس مباشرة على الـ API.

لو ال cmd رافض اكتب اي حاجة**/ لو السيرفر شغال عادي وعايزاه يقفل**

دوسي CTRL + C في نفس نافذة الـ cmd → هيوقف السيرفر وتقدري تكتبي أوامر جديدة.

لو ال cmd مش راضي يقرا الملف من ال path   
فحددي الـ path كامل للملف من أي مكان افتحي cmd واكتبي:

json-server --watch G:\SenSe\Imp\TESTING\SS\_TESTING\API\_Restassured\db.json

ده بيشغّل السيرفر مباشرة على الملف اللي في الـ G:.

**How to test on local rest API:**

public class TestsOnLocalAPI {

@Test

public void get() {

given().baseUri("http://localhost:3000")

.when().get("/users")

.then().statusCode(200).log().all();

}

@Test

public void delete() {

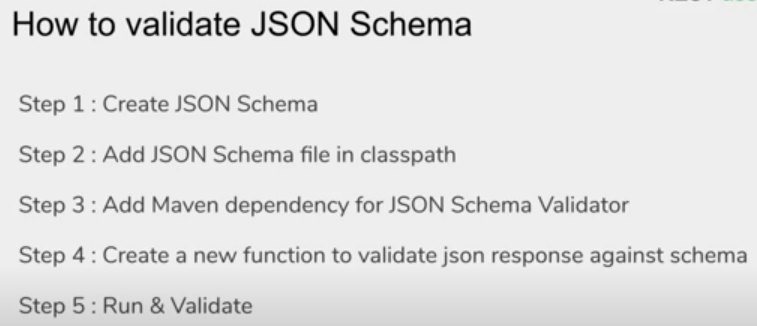
when().delete("http://localhost:3000/users/4") // full URL is fine for local

.then().statusCode(anyOf(is(200), is(404))); // 404 because it is already deleted

}

}

**How to validate json schema:**



<https://www.liquid-technologies.com/online-json-to-schema-converter> >>   
copy the response from(<https://reqres.in/api/users?page=2>) and paste it, it will convert it to json schema   
then from the package explorer left side menu on eclipse:  
target-properties-browse(target-classes-create file schema.json and paste in it the schema json)

then create a class and I will go to (<https://github.com/rest-assured/rest-assured/wiki/Usage>) to make an example  
public class JSONSchemaValidator extends BaseTest {

@Test

public void testGet() {

given().

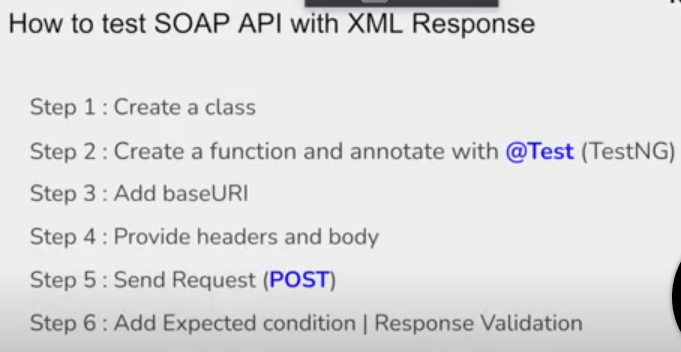
when().get("/users?page=2")

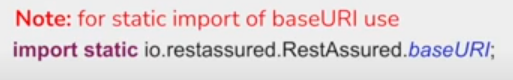
.then().assertThat().body(matchesJsonSchemaInClasspath("db.json")) .statusCode(200);

}

}

**How to test soap API with xml response:**





Create a class then create a function

<http://www.dneonline.com/calculator.asmx> >> soap request & response

And I will create a file “add.xml”

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"

targetNamespace="http://tempuri.org/"

elementFormDefault="qualified"

attributeFormDefault="unqualified">

<xs:element name="AddResponse">

<xs:complexType>

<xs:sequence>

<xs:element type="xs:int" name="AddResult" />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

Project-create a new folder “ soap request” and put the xml file in it

public class SoapXMLRequest {

@Test

public void validateSoapXML() throws IOException {

File file = new File("./SoapRequest/Add.xml"); عشان يقرا من الفايل اللي عملته في البروجيكت

if (file.exists())

System.out.println(" >> File Exists");

FileInputStream fileInputStream = new FileInputStream(file);

String requestBody = IOUtils.toString(fileInputStream, "UTF-8");

baseURI = "http://www.dneonline.com";

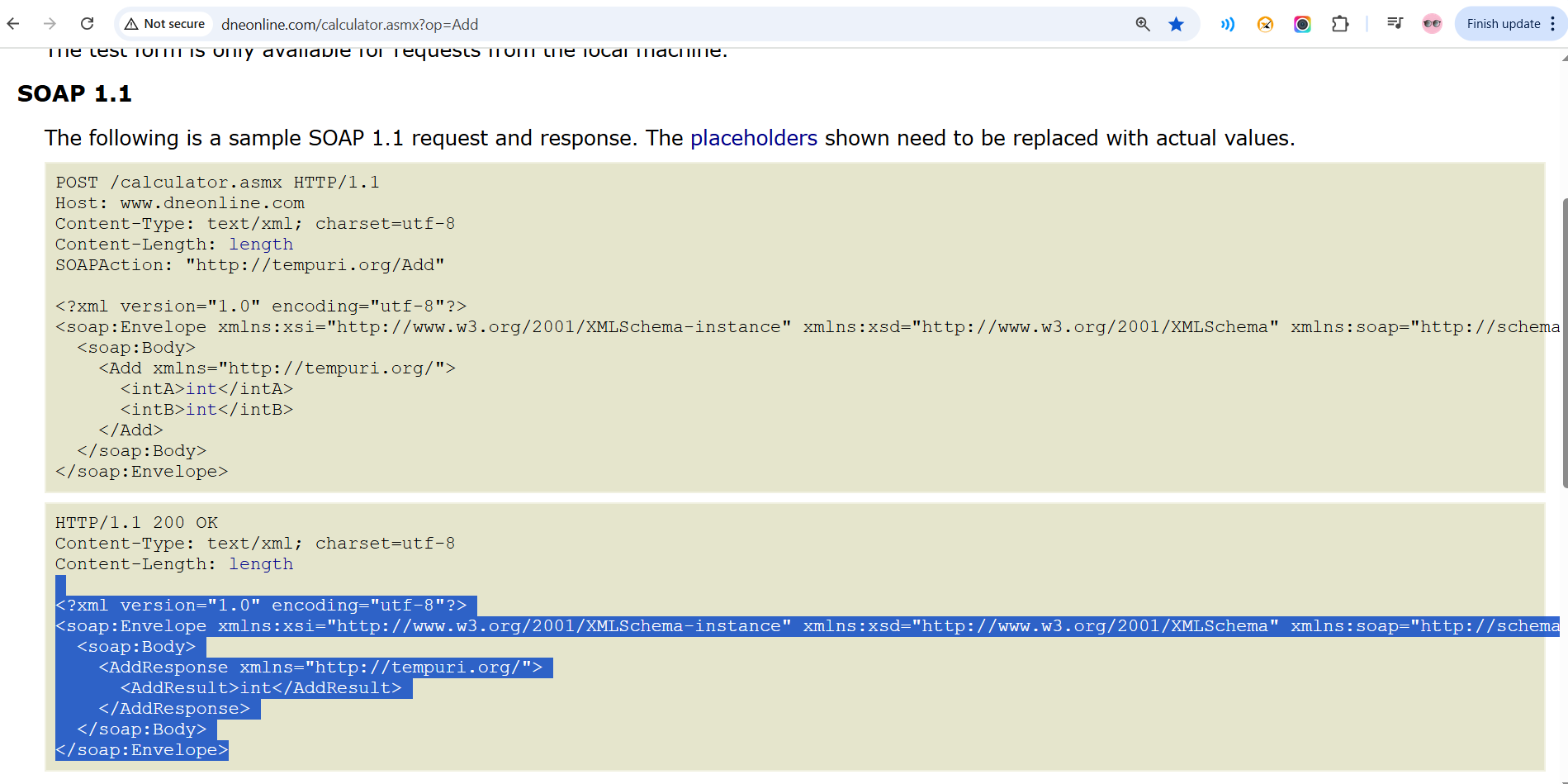
given().contentType("text/xml").accept(ContentType.XML).body(requestBody)

.when().post("/calculator.asmx")

.then() .statusCode(anyOf(is(200), s(201))).log().all().body("//\*:AddResult.text()", equalTo("5"));

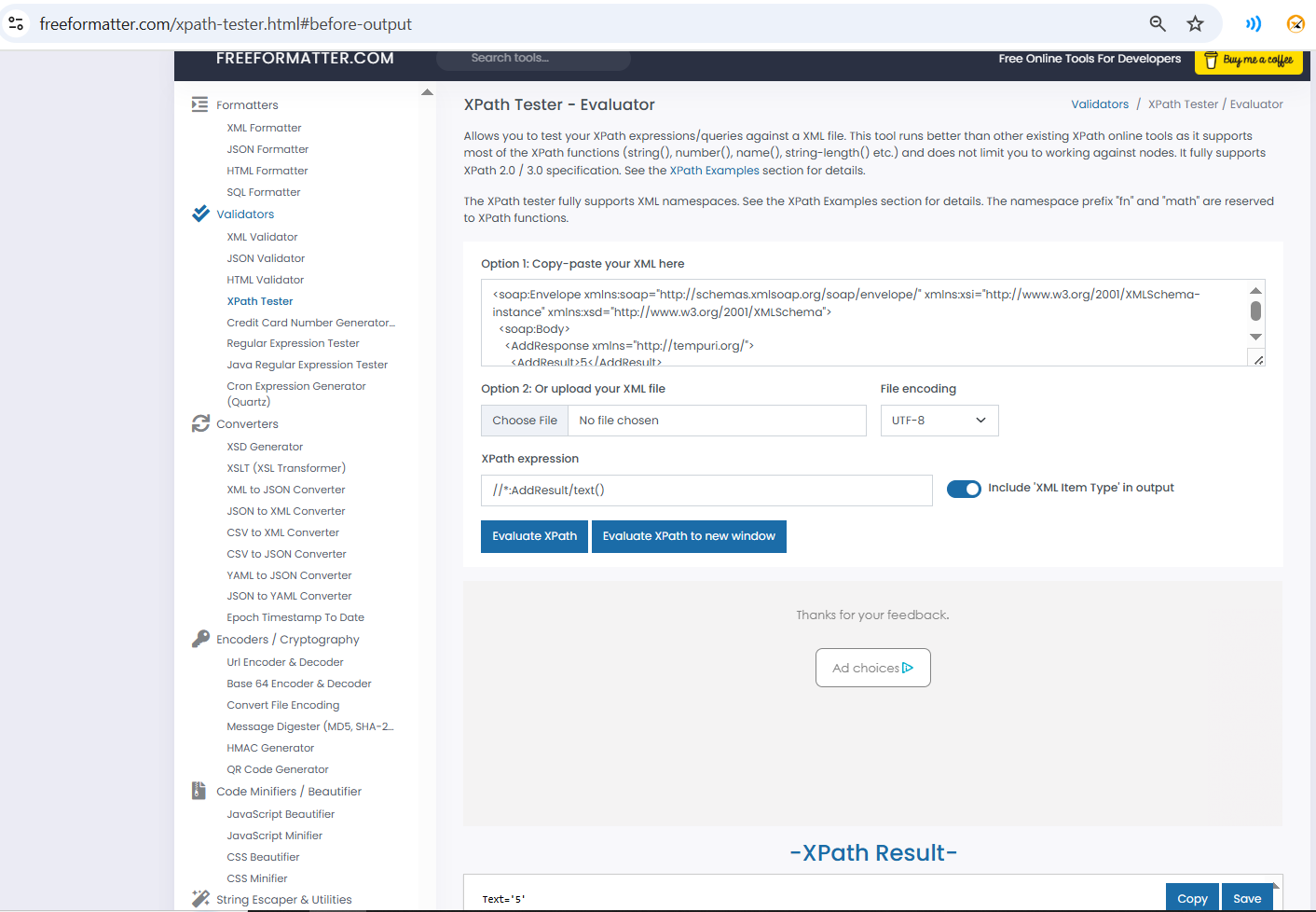
}

}

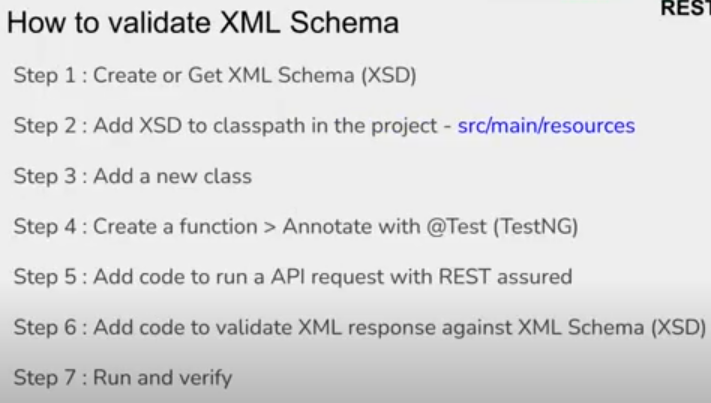


Then I need validation

<https://www.freeformatter.com/> >> to get Xpath tester



**How to validate xml schema: I want to verify my response against an xml schema**



I will create a file “calculator.xsd”

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"

targetNamespace="http://tempuri.org/"

elementFormDefault="qualified"

attributeFormDefault="unqualified">

<xs:element name="AddResponse">

<xs:complexType>

<xs:sequence>

<xs:element type="xs:int" name="AddResult" />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

And I will add it in src/test/resources

Create a class  
public class XMLSchemaValidation {

@Test

public void schemaValidation() throws IOException {

File file = new File("./SoapRequest/Add.xml");

if (file.exists())

System.out.println(" >> File Exists");

FileInputStream fileInputStream = new FileInputStream(file);

String requestBody = IOUtils.toString(fileInputStream, "UTF-8");

baseURI = "http://www.dneonline.com";

given().contentType("text/xml").accept(ContentType.XML).body(requestBody)

.when().post("/calculator.asmx")

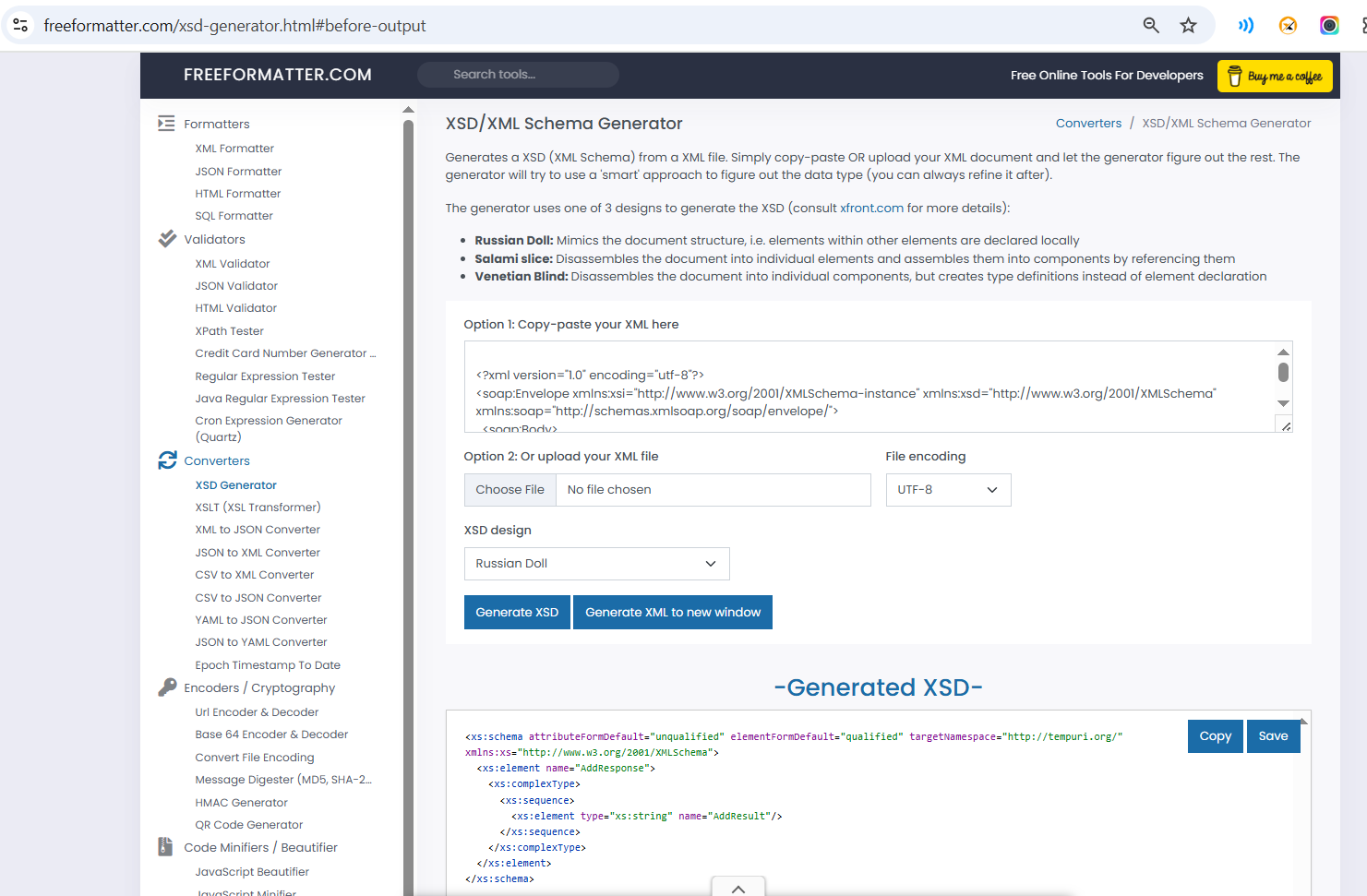
.then() .statusCode(anyOf(is(200), is(201))).log().all().and().body("//\*:AddResult.text()", equalTo("5")).and()

.assertThat().body(matchesXsdInClasspath("CalculatorEnvelope.xsd"));

}

}

<https://www.freeformatter.com/> >> to get Xsd generator (just paste xml response and it will convert it to xsd schema)

`

There was an error (Cannot Find The Declaration Of Element 'soap:Envelope') in calculator.xsd file

So I added there 2 lines in it

xsi:schemaLocation="http://tempuri.org/ tempuri.xsd

http://schemas.xmlsoap.org/soap/envelope/

http://schemas.xmlsoap.org/soap/envelope/

There was a problem here to navigate to URLs in the calculator.xsd file and it keeps give me error (Premature End of File Error?),, So I added a file calculator envelope.xsd

<!-- Schema for the SOAP/1.1 envelope

Portions © 2001 DevelopMentor.

© 2001 W3C (Massachusetts Institute of Technology, Institut National de

Recherche en Informatique et en Automatique, Keio University). All Rights

Reserved.

This document is governed by the W3C Software License [1] as described in the

FAQ [2].

[1] http://www.w3.org/Consortium/Legal/copyright-software-19980720

[2] http://www.w3.org/Consortium/Legal/IPR-FAQ-20000620.html#DTD

By obtaining, using and/or copying this work, you (the licensee) agree that you

have read, understood, and will comply with the following terms and conditions:

Permission to use, copy, modify, and distribute this software and its

documentation, with or without modification, for any purpose and without fee or

royalty is hereby granted, provided that you include the following on ALL copies

of the software and documentation or portions thereof, including modifications,

that you make:

1. The full text of this NOTICE in a location viewable to users of the

redistributed or derivative work.

2. Any pre-existing intellectual property disclaimers, notices, or terms and

conditions. If none exist, a short notice of the following form (hypertext is

preferred, text is permitted) should be used within the body of any

redistributed or derivative code: "Copyright © 2001 World Wide Web Consortium,

(Massachusetts Institute of Technology, Institut National de Recherche en

Informatique et en Automatique, Keio University). All Rights Reserved.

http://www.w3.org/Consortium/Legal/"

3. Notice of any changes or modifications to the W3C files, including the date

changes were made. (We recommend you provide URIs to the location from which the

code is derived.)

Original W3C files; http://www.w3.org/2001/06/soap-envelope

Changes made:

- reverted namespace to http://schemas.xmlsoap.org/soap/envelope/

- reverted mustUnderstand to only allow 0 and 1 as lexical values

- made encodingStyle a global attribute 20020825

- removed default value from mustUnderstand attribute declaration

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE

NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED

TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT

THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY

PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR

CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR DOCUMENTATION.

The name and trademarks of copyright holders may NOT be used in advertising or

publicity pertaining to the software without specific, written prior permission.

Title to copyright in this software and any associated documentation will at all

times remain with copyright holders.

-->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"

xmlns:tns="http://schemas.xmlsoap.org/soap/envelope/"

targetNamespace="http://schemas.xmlsoap.org/soap/envelope/">

<!-- Envelope, header and body -->

<xs:element name="Envelope" type="tns:Envelope" />

<xs:complexType name="Envelope">

<xs:sequence>

<xs:element ref="tns:Header" minOccurs="0" />

<xs:element ref="tns:Body" minOccurs="1" />

<xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"

processContents="lax" />

</xs:sequence>

<xs:anyAttribute namespace="##other" processContents="lax" />

</xs:complexType>

<xs:element name="Header" type="tns:Header" />

<xs:complexType name="Header">

<xs:sequence>

<xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"

processContents="lax" />

</xs:sequence>

<xs:anyAttribute namespace="##other" processContents="lax" />

</xs:complexType>

<xs:element name="Body" type="tns:Body" />

<xs:complexType name="Body">

<xs:sequence>

<xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"

processContents="lax" />

</xs:sequence>

<xs:anyAttribute namespace="##any" processContents="lax">

<xs:annotation>

<xs:documentation> Prose in the spec does not specify that

attributes are allowed on the Body element </xs:documentation>

</xs:annotation>

</xs:anyAttribute>

</xs:complexType>

<!-- Global Attributes. The following attributes are intended to be usable

via qualified attribute names on any complex type referencing them. -->

<xs:attribute name="mustUnderstand">

<xs:simpleType>

<xs:restriction base="xs:boolean">

<xs:pattern value="0|1" />

</xs:restriction>

</xs:simpleType>

</xs:attribute>

<xs:attribute name="actor" type="xs:anyURI" />

<xs:simpleType name="encodingStyle">

<xs:annotation>

<xs:documentation> 'encodingStyle' indicates any canonicalization

conventions followed in the contents of the containing element.

For example, the value

'http://schemas.xmlsoap.org/soap/encoding/' indicates the

pattern described in SOAP specification </xs:documentation>

</xs:annotation>

<xs:list itemType="xs:anyURI" />

</xs:simpleType>

<xs:attribute name="encodingStyle" type="tns:encodingStyle" />

<xs:attributeGroup name="encodingStyle">

<xs:attribute ref="tns:encodingStyle" />

</xs:attributeGroup>

<xs:element name="Fault" type="tns:Fault" />

<xs:complexType name="Fault" final="extension">

<xs:annotation>

<xs:documentation> Fault reporting structure </xs:documentation>

</xs:annotation>

<xs:sequence>

<xs:element name="faultcode" type="xs:QName" />

<xs:element name="faultstring" type="xs:string" />

<xs:element name="faultactor" type="xs:anyURI" minOccurs="0" />

<xs:element name="detail" type="tns:detail" minOccurs="0" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="detail">

<xs:sequence>

<xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"

processContents="lax" />

</xs:sequence>

<xs:anyAttribute namespace="##any" processContents="lax" />

</xs:complexType>

</xs:schema>

There was a problem that 2 classes use a different url that mentioned in the basetest class that include api key for the common url between classes, so I did 2 things

1- I make an update to the 2 classes which use a different url   
given().baseUri("http://www.dneonline.com") // Local base URI

.header("x-api-key", "") // Remove global API key if inherited

.contentType("text/xml").accept(ContentType.XML).body(requestBody).

2-i created a testing.xml under the project in eclipse  
 <!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >

<suite name="ApiTestsSuite" verbose="1" parallel="methods" thread-count="5">

<test name="AllApiTests">

<packages>

<package name="tests.\*" /> <!-- Includes all classes in tests package -->

</packages>

</test>

</suite>

دلوقتي انا عملت مشروع automation rest assured كامل هكتبهولك وعايزاك تقولي كل سطر معمول ليه وتشرحهولي عشان اعرف ارجعله بعد كدا واعمله من اول وجديد

|  |  |
| --- | --- |
| package tests;  import org.testng.annotations.BeforeSuite;  import io.restassured.RestAssured;  import io.restassured.builder.RequestSpecBuilder;  import io.restassured.specification.RequestSpecification;  public class BaseTest {  @BeforeSuite  public void setup() {  RequestSpecification reqSpec = new RequestSpecBuilder()  .setBaseUri("https://reqres.in/api")  // global base for reqres tests  .addHeader("x-api-key", "reqres-free-v1")  // global API key  .build();  RestAssured.requestSpecification = reqSpec;  System.out.println("API Key verified and default baseUri set for reqres tests");  }  } | * BeforeSuite: TestNG annotation عشان يتنفذ الميثود مرة واحدة قبل أي Suite * RestAssured: الكلاس الرئيسي للتعامل مع الـ REST API * RequestSpecBuilder: بيستخدم عشان تبني Request Specification يعني إعدادات مشتركة * RequestSpecification: الواجهة اللي بتمثل إعدادات الـ request.   ➡️ الميثود دي هتتعمل مرة واحدة قبل أي tests هدفها إعداد الإعدادات العامة.  بنبني request specification:   1. setBaseUri: بنحدد الـ base URL الأساسي لكل التستات. 2. addHeader: بنضيف هيدر ثابت (API key). 3. build(): بيبني الـ specification.   ➡️ بيخلي الـ request specification اللي اتبنى هو الـ default لكل التستات  **الهدف**: إعداد إعدادات عامة للمشروع كله عشان متضطرش تكررها في كل test |
| package tests;  import static io.restassured.RestAssured.given;  import static org.hamcrest.Matchers.equalTo;  import static org.hamcrest.Matchers.hasItems;  import java.util.HashMap;  import java.util.Map;  import org.json.simple.JSONObject;  import org.testng.annotations.Test;  import io.restassured.http.ContentType;  public class GetAndPostExamples extends BaseTest {  @Test  public void testGet() {  given().  when().get("/users?page=2") // relative path, global baseUri used  .then().statusCode(200).body("data[4].first\_name", equalTo("George")) .body("data.first\_name", hasItems("George", "Rachel"));  }  @Test  public void testPost() {  Map<String, Object> map = new HashMap<>();  map.put("name", "Raghav");  map.put("job", "Teacher");  JSONObject request = new JSONObject(map);  given().contentType(ContentType.JSON).accept(ContentType.JSON).body(request.toJSONString())  .when() .post("/users")  .then().statusCode(201).log().all();  }  } | * ➡️ استدعاء static للـ RestAssured و Hamcrest matchers (عشان نكتب الكود بشكل مختصر). * ➡️ الكلاس ده بيورث من BaseTest فبيستخدم الـ baseUri والهيدر. خطوات الـ test:  1. given() بداية request. 2. get("/users?page=2")   بيعمل GET باستخدام relative path (لإن الـ baseUri جاهز).   1. statusCode(200) بيتأكد إن الاستجابة ناجحة. 2. body("data[4].first\_name", equalTo("George")): 3. بيتأكد إن اسم الـ user الخامس George. 4. body("data.first\_name", hasItems("George", "Rachel")): 5. بيتأكد إن الاتنين موجودين في الـ response.    Map<String, Object> map: إنشاء Map لحفظ البيانات   * هنا بنجهز body كـ map (مفتاح-قيمة).     JSONObject request: تحويل الـ Map لـ JSON object   * نحولها لـ JSON Object. * POST request:  1. Content-Type JSON. 2. Body جاهز. 3. POST للـ endpoint /users. 4. يتأكد إن الاستجابة 201 Created. 5. log().all(): يطبع كل التفاصيل. |
| package tests;  import static io.restassured.RestAssured.given;  import static io.restassured.module.jsv.JsonSchemaValidator.matchesJsonSchemaInClasspath;  import org.testng.annotations.Test;  public class JSONSchemaValidator extends BaseTest {  @Test  public void testGet() {  given().  when().get("/users?page=2")  .then().assertThat().body(matchesJsonSchemaInClasspath("db.json")) .statusCode(200);  }  } |  matchesJsonSchemaInClasspath():  التحقق إن الـ JSON response يتطابق مع schema محدد لملف JSON Schema (db.json) موجود في resources. |
| package tests;  import static io.restassured.RestAssured.given;  import static io.restassured.RestAssured.when;  import org.json.simple.JSONObject;  import org.testng.annotations.Test;  import io.restassured.http.ContentType;  public class PutPatchDeleteExamples extends BaseTest {  @Test  public void testPut() {  JSONObject request = new JSONObject();  request.put("name", "Raghav");  request.put("job", "Teacher");  given().contentType(ContentType.JSON).accept(ContentType.JSON).body(request.toJSONString())  .when().put("/users/2")  .then().statusCode(200).log().all();  }  @Test  public void testPatch() {  JSONObject request = new JSONObject();  request.put("name", "Raghav");  request.put("job", "Teacher");  given().contentType(ContentType.JSON).accept(ContentType.JSON).body(request.toJSONString())  .when().patch("/users/2")  .then().statusCode(200).log().all();  }  @Test  public void testDelete() {  when().delete("/users/2")  .then().statusCode(204).log().all();  }  } | PUT: لتحديث resource كامل  /users/2: تحديث الـ user اللي ID بتاعه 2  Status code 200: تأكيد نجاح العملية  PATCH: لتحديث جزء من الـ resource مش كله  نفس الطريقة بس HTTP method مختلف  DELETE: لحذف resource  Status code 204: يعني العملية نجحت بس مافيش محتوى يرجع  كلهم بنفس الفكرة: إعداد body → إرسال request → التأكد من الـ status code. |
| package tests;  import static io.restassured.RestAssured.baseURI;  import static io.restassured.RestAssured.given;  import static org.hamcrest.Matchers.anyOf;  import static org.hamcrest.Matchers.equalTo;  import static org.hamcrest.Matchers.is;  import java.io.File;  import java.io.FileInputStream;  import java.io.IOException;  import org.apache.commons.io.IOUtils;  import org.testng.annotations.Test;  import io.restassured.http.ContentType;  public class SoapXMLRequest {    @Test  public void validateSoapXML() throws IOException {  File file = new File("./SoapRequest/Add.xml");  if (file.exists())  System.out.println(" >> File Exists");  FileInputStream fileInputStream = new FileInputStream(file);  String requestBody = IOUtils.toString(fileInputStream, "UTF-8");  baseURI = "http://www.dneonline.com";  given().contentType("text/xml").accept(ContentType.XML).body(requestBody)  .when().post("/calculator.asmx")  .then() .statusCode(anyOf(is(200), s(201))).log().all().body("//\*:AddResult.text()", equalTo("5"));  }  } | * هنا بيتعامل مع SOAP API مش REST: * يقرأ ملف XML (Add.xml) من ./SoapRequest.   File file = new File(): قراءة ملف الـ XML request   * يحوله String باستخدام IOUtils.   IOUtils.toString(): تحويل محتوى الملف لـ String  يرسل POST للـ  baseURI = "http://www.dneonline.com": تغيير الـ base URI للـ SOAP service  "//\*:AddResult.text()": XPath للوصول لقيمة النتيجة في الـ XML response  SOAP endpoint http://www.dneonline.com/calculator.asmx.  يتأكد إن AddResult = 5.  .contentType("text/xml"): تحديد نوع المحتوى كـ XML  anyOf(is(200), is(201)): قبول إما 200 أو 201 كـ status code صحيح |
| package tests;  import static io.restassured.RestAssured.given;  import static org.hamcrest.Matchers.equalTo;  import org.testng.Assert;  import org.testng.annotations.Test;  import io.restassured.response.Response;  public class TestsExamples extends BaseTest {  @Test  public void test\_1() {  Response response = given().  when().get("/users?page=2");  int statusCode = response.getStatusCode();  Assert.assertEquals(statusCode, 200);  }  @Test  public void test\_2() {  given().  when().get("/users?page=2")  .then().statusCode(200).body("data[1].id", equalTo(8)).log().all();  }  } | طريقة أخرى للتحقق باستخدام method chaining بدل حفظ الـ response   * test\_1: بيجيب response ويحفظه → يتأكد من الـ status code باستخدام Assert.   Response response: حفظ الـ response في متغير للتعامل معه  .getStatusCode(): استخراج الـ status code من الـ response  Assert.assertEquals(): استخدام TestNG assertions للتحقق   * test\_2: مباشرة بيتأكد من القيمة اللي في response. |
| package tests;  import static io.restassured.RestAssured.given;  import static io.restassured.RestAssured.when;  import static org.hamcrest.Matchers.anyOf;  import static org.hamcrest.Matchers.is;  import org.testng.annotations.Test;  public class TestsOnLocalAPI {  @Test  public void get() {  given().baseUri("http://localhost:3000")  .when().get("/users")  .then().statusCode(200).log().all();  }  @Test  public void delete() {  when().delete("http://localhost:3000/users/4") // full URL is fine for local  .then().statusCode(anyOf(is(200), is(404))); // 404 because it is already deleted  }  } | بيجرب على API محلي (شغال بـ JSON Server على <http://localhost:3000>   * Get GET كل الـ users. * delete يحذف user معين. |
| package tests;  import static io.restassured.RestAssured.baseURI;  import static io.restassured.RestAssured.given;  import static io.restassured.matcher.RestAssuredMatchers.matchesXsdInClasspath;  import static org.hamcrest.Matchers.anyOf;  import static org.hamcrest.Matchers.equalTo;  import static org.hamcrest.Matchers.is;  import java.io.File;  import java.io.FileInputStream;  import java.io.IOException;  import org.apache.commons.io.IOUtils;  import org.testng.annotations.Test;  import io.restassured.http.ContentType;  public class XMLSchemaValidation {  @Test  public void schemaValidation() throws IOException {  File file = new File("./SoapRequest/Add.xml");  if (file.exists())  System.out.println(" >> File Exists");  FileInputStream fileInputStream = new FileInputStream(file);  String requestBody = IOUtils.toString(fileInputStream, "UTF-8");  baseURI = "http://www.dneonline.com";  given().contentType("text/xml").accept(ContentType.XML).body(requestBody)  .when().post("/calculator.asmx")  .then() .statusCode(anyOf(is(200), is(201))).log().all().and().body("//\*:AddResult.text()", equalTo("5")).and()  .assertThat().body(matchesXsdInClasspath("CalculatorEnvelope.xsd"));  }  } | * نفس فكرة SOAP test بس كمان بيتأكد إن الاستجابة متوافقة مع ملف XSD Schema (CalculatorEnvelope.xsd).   .baseUri("http://localhost:3000"): تغيير الـ base URI للـ local server  الهدف: اختبار APIs محلية بدل الخدمات الخارجية |
|  |  |
| <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  targetNamespace="http://tempuri.org/"  elementFormDefault="qualified"  attributeFormDefault="unqualified">  <xs:element name="AddResponse">  <xs:complexType>  <xs:sequence>  <xs:element type="xs:int" name="AddResult" />  </xs:sequence>  </xs:complexType>  </xs:element>  </xs:schema> | * Xsd file |
| -->  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  xmlns:tns="http://schemas.xmlsoap.org/soap/envelope/"  targetNamespace="http://schemas.xmlsoap.org/soap/envelope/">  <!-- Envelope, header and body -->  <xs:element name="Envelope" type="tns:Envelope" />  <xs:complexType name="Envelope">  <xs:sequence>  <xs:element ref="tns:Header" minOccurs="0" />  <xs:element ref="tns:Body" minOccurs="1" />  <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"  processContents="lax" />  </xs:sequence>  <xs:anyAttribute namespace="##other" processContents="lax" />  </xs:complexType>  <xs:element name="Header" type="tns:Header" />  <xs:complexType name="Header">  <xs:sequence>  <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"  processContents="lax" />  </xs:sequence>  <xs:anyAttribute namespace="##other" processContents="lax" />  </xs:complexType>  <xs:element name="Body" type="tns:Body" />  <xs:complexType name="Body">  <xs:sequence>  <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"  processContents="lax" />  </xs:sequence>  <xs:anyAttribute namespace="##any" processContents="lax">  <xs:annotation>  <xs:documentation> Prose in the spec does not specify that  attributes are allowed on the Body element </xs:documentation>  </xs:annotation>  </xs:anyAttribute>  </xs:complexType>  <!-- Global Attributes. The following attributes are intended to be usable  via qualified attribute names on any complex type referencing them. -->  <xs:attribute name="mustUnderstand">  <xs:simpleType>  <xs:restriction base="xs:boolean">  <xs:pattern value="0|1" />  </xs:restriction>  </xs:simpleType>  </xs:attribute>  <xs:attribute name="actor" type="xs:anyURI" />  <xs:simpleType name="encodingStyle">  <xs:annotation>  <xs:documentation> 'encodingStyle' indicates any canonicalization  conventions followed in the contents of the containing element.  For example, the value  'http://schemas.xmlsoap.org/soap/encoding/' indicates the  pattern described in SOAP specification </xs:documentation>  </xs:annotation>  <xs:list itemType="xs:anyURI" />  </xs:simpleType>  <xs:attribute name="encodingStyle" type="tns:encodingStyle" />  <xs:attributeGroup name="encodingStyle">  <xs:attribute ref="tns:encodingStyle" />  </xs:attributeGroup>  <xs:element name="Fault" type="tns:Fault" />  <xs:complexType name="Fault" final="extension">  <xs:annotation>  <xs:documentation> Fault reporting structure </xs:documentation>  </xs:annotation>  <xs:sequence>  <xs:element name="faultcode" type="xs:QName" />  <xs:element name="faultstring" type="xs:string" />  <xs:element name="faultactor" type="xs:anyURI" minOccurs="0" />  <xs:element name="detail" type="tns:detail" minOccurs="0" />  </xs:sequence>  </xs:complexType>  <xs:complexType name="detail">  <xs:sequence>  <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"  processContents="lax" />  </xs:sequence>  <xs:anyAttribute namespace="##any" processContents="lax" />  </xs:complexType>  </xs:schema> | * CalculatorEnvelope.xsd * schema للتحقق من SOAP response. |
| {  "users": [  {  "firstName": "Yehia",  "lastName": "Mostafa",  "subjectId": 1,  "id": 1  },  {  "firstName": "Adam",  "lastName": "Mostafa",  "subjectId": 2,  "id": 2  },  {  "firstName": "Lara",  "lastName": "Mohamed",  "subjectId": 1,  "id": 3  },  {  "firstName": "Laila",  "lastName": "Mohamed",  "subjectId": 1,  "id": 5  }  ],  "subjects": [  {  "name": "Automation",  "id": 1  },  {  "name": "Devops",  "id": 2  }  ]  } | * db.json * data/ schema عشان الـ local API |
| <Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">  <Body>  <Add xmlns="http://tempuri.org/">  <intA>2</intA>  <intB>3</intB>  </Add>  </Body>  </Envelope> | Add.xml |

**steps to install new java**

## 1) Pick and download a JDK

1. Go to the Eclipse Temurin (Adoptium) downloads page and choose **JDK 17 (LTS)** or **JDK 11 (LTS)** → Windows x64 installer (msi/exe). [Adoptium+1](https://adoptium.net/temurin/releases?utm_source=chatgpt.com)
2. Download the Windows installer (EXE / MSI).

## 2) Run the installer

1. Run the downloaded .exe/.msi as administrator.
2. Accept the license, pick install folder (default: C:\Program Files\Java\jdk-17.x.x is fine).
3. Some installers offer to set JAVA\_HOME for you — if available you can allow it, but I still show how to set manually below. [Adoptium](https://adoptium.net/installation?utm_source=chatgpt.com)

## 3) Set JAVA\_HOME and update PATH (GUI method — safest)

1. Press **Windows** key → type **Environment variables** → click **Edit the system environment variables**.
2. In System Properties → **Advanced** → **Environment Variables…**.
3. Under **System variables**:
   * Click **New…**
     + Variable name: JAVA\_HOME
     + Variable value: C:\Program Files\Java\jdk-17.0.x (the folder where you installed the JDK). [Oracle Documentation+1](https://docs.oracle.com/cd/E19182-01/821-0917/inst_set_jdk_windows_t/index.html?utm_source=chatgpt.com)
   * Find the Path system variable → **Edit** → **New** → add: %JAVA\_HOME%\bin and **move it to the top** (so the new java is found before any old java installs).
4. Click OK on all dialogs.
5. **Important:** close and reopen any Command Prompt, PowerShell or IDE windows so they pick up the new variables.

Alternative (command-line) to set JAVA\_HOME for all users (run Command Prompt **as admin**):

setx /M JAVA\_HOME "C:\Program Files\Java\jdk-17.0.x"

To add %JAVA\_HOME%\bin to PATH with setx you must include the existing PATH value — GUI is simpler. (Example setx usage documented). [Atlassian Documentation+1](https://confluence.atlassian.com/spaces/DOC/pages/8895/Setting%2Bthe%2BJAVA_HOME%2BVariable%2Bin%2BWindows?utm_source=chatgpt.com)

## 4) Verify the new JDK is used

Open **new** Command Prompt or PowerShell and run:

java -version

javac -version

where java

echo %JAVA\_HOME%

Expected: java -version shows openjdk version "17.x" (or 11.x if you chose 11). where java may show multiple results — ensure the path that comes first is under C:\Program Files\Java\.... If it still shows 1.8.0\_391, you need to remove/adjust the old Java path or ensure %JAVA\_HOME%\bin comes before the old path. [Stack Overflow+1](https://stackoverflow.com/questions/51243045/echo-java-home-returns-java-home?utm_source=chatgpt.com)

## 5) Update your IDE / build tools

* **IntelliJ IDEA:** File → Project Structure → SDKs → **+** → JDK → point to C:\Program Files\Java\jdk-17.x; then set Project SDK to that.
* **Maven/Gradle builds on CI**: ensure CI machines and JAVA\_HOME on those systems are updated too. (Tools may use the JAVA\_HOME or an IDE-configured JDK.)
* **Eclipse:** Window → Preferences → Java → Installed JREs → Add → Standard VM → point to the new JDK folder and check it as default for the workspace.

Perfect 👌 you already installed JDK 17 at:  
C:\Program Files\Eclipse Adoptium\jdk-17.0.16.8-hotspot

Now in the Eclipse window you opened ("Add JRE"):

1. **JRE home:**  
   👉 Click **Directory…** and browse to:  
   C:\Program Files\Eclipse Adoptium\jdk-17.0.16.8-hotspot  
   (select the JDK root folder, not bin).
2. **JRE name:**  
   You can type something descriptive, e.g.:  
   jdk-17
3. **Default VM arguments:**  
   Leave it **empty** (unless you need special flags, not common).
4. After that, the **JRE system libraries** box will auto-populate with JARs from lib.
5. Click **Finish**.
6. Back in the "Installed JREs" list, checkmark the new jdk-17 so it becomes the default for new projects.
7. Then click **Apply and Close**.

If Eclipse is **still running your project with Java 8** even though you installed and added Java 17. Let’s fix that step by step.

### 1. Check Eclipse itself

Eclipse can run with a different JVM than your project.

* In Eclipse, go to:  
  **Help → About Eclipse IDE → Installation Details → Configuration**  
  Look for the line:
* java.version=...

If it says 1.8.0\_391, Eclipse itself is still running on Java 8.  
→ In that case, you need to edit your Eclipse shortcut or eclipse.ini file to point to JDK 17.

Add these two lines at the **top** (before -vmargs) in eclipse.ini:

-vm

C:/Program Files/Eclipse Adoptium/jdk-17.0.16.8-hotspot/bin/javaw.exe

### 2. Make sure your project is compiled with JDK 17

Right-click your project → **Properties**:

* **Java Build Path** → Libraries tab → make sure the JRE System Library is jdk-17.
* If it still shows JavaSE-1.8, remove it and click **Add Library → JRE System Library → Alternate JRE → jdk-17**.

Then:

* Go to **Java Compiler** → check "Enable project specific settings" → set **Compiler compliance level: 17**.

Click **Apply and Close**, then **Project → Clean** to rebuild.

### 3. Check Run Configuration

Sometimes, even if the project is correct, the **Run/Debug configuration** is still using Java 8.

* Go to **Run → Run Configurations…**
* Select your TestNG run config.
* Go to the **JRE tab**.
* Choose **Alternate JRE → jdk-17**.

Great 👍 thanks for showing me this screenshot.  
Now we know **Eclipse itself is running with Java 17**:

java.version=17.0.9

So the issue is **not Eclipse’s JVM**, but your **project or run configuration** still using Java 8 settings.

### ✅ What to check next:

#### 1. Project JRE in Build Path

* Right-click your project → **Properties**.
* Go to **Java Build Path → Libraries**.
* Look at **JRE System Library**:
  + If it says something like JavaSE-1.8, remove it.
  + Click **Add Library → JRE System Library → Alternate JRE → jdk-17** (the one you added earlier).
* Apply and Close.

#### 2. Compiler Compliance Level

* Still in **Project Properties → Java Compiler**.
* Check “Enable project specific settings”.
* Set **Compiler compliance level = 17**.
* Apply and Close.
* Then do **Project → Clean…**.

#### 3. Run Configuration

* Go to **Run → Run Configurations…**.
* Select your TestNG (or Java Application) configuration.
* Go to the **JRE tab**.
* Change to **Alternate JRE → jdk-17**.
* Apply, then Run.

#### 4. Verify inside your project

In your main class (or test), add this quick check:

System.out.println("Running Java version: " + System.getProperty("java.version"));

Run again — it should print something like 17.0.9.  
If it still prints 1.8.0\_391, then your Run Configuration is still tied to Java 8.