Rest-assured: Rest-assured is a Java library that provides a domain-specific language (DSL) for writing powerful, maintainable tests for RESTful APIs. It simplifies the testing of HTTP-based services and supports various HTTP methods such as GET, POST, PUT, DELETE, etc.

json-path: json-path is a JSON processing library that provides a simple DSL for querying JSON documents. It allows you to extract values from JSON responses received from RESTful APIs using JSONPath expressions.

json: This is the JSON library provided by the org.json project. It offers functionality to parse, generate, transform, and query JSON data in Java. It's commonly used for handling JSON data in Java applications.

gson: Gson is a Java library for serializing and deserializing Java objects to and from JSON. It provides simple to use APIs to convert Java objects into their JSON representation and vice versa.

testng: TestNG is a testing framework for the Java programming language inspired by JUnit and NUnit. It provides additional features such as annotations, grouping, parameterization, and dependency testing over JUnit.

scribejava-apis: ScribeJava is a simple OAuth library for Java. It provides support for OAuth 1.0a and OAuth 2.0 for interacting with various OAuth-enabled services, including popular social media platforms like Twitter, Facebook, etc.

json-schema-validator: json-schema-validator is a Java implementation for validating JSON against a JSON Schema. It allows you to define a schema for your JSON data and validate whether your JSON documents conform to that schema.

xml-schema-validator: xmlschema-core is part of the Apache XMLSchema project. It's a library for parsing, validating, and manipulating XML Schema documents in Java. It provides APIs to work with XML Schema documents, validate XML documents against schemas, etc.

Each of these dependencies serves a specific purpose in Java development, ranging from REST API testing, JSON manipulation, testing framework support, OAuth authentication, to XML schema validation. Depending on your project requirements, you include these dependencies to leverage their functionality and simplify development tasks.

**Rest assured Structure** ==================================================================================

given ()

content type, set cookies, add auth, add param, set headers info etc....

.when ()

get, post, put, delete

then (), and()

Note :

Ex🡪 given ().when ().then () || when ().then () 🡪(if given no req. we can directly remove that.)

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

**Static imports**

In order to use REST assured effectively it's recommended to statically import methods from the following classes: make sure all import should be static

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

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

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

Basic Example

**package Day1;**

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

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

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

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

**import java.util.HashMap;**

**public class HTTPRequest {**

**int id;**

**@Test**

**void getUsers() {**

**given()**

**.when()**

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

**.then()**

**.statusCode(200)**

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

**.log().all();**

**}**

**@Test(priority=2)**

**void createUser() {**

**HashMap hm = new HashMap();**

**hm.put("name","kuresh");**

**hm.put("job","leader");**

**id=given()**

**.contentType("application/json")**

**.body(hm)**

**.when()**

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

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

**// .then()**

**// .statusCode(201)**

**// .log().all();**

**}**

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

**void updateUser() {**

**HashMap hm = new HashMap();**

**hm.put("name","kuresh");**

**hm.put("job","leader");**

**given()**

**.contentType("application/json")**

**.body(hm)**

**.when()**

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

**.then()**

**.statusCode(200)**

**.log().all();**

**}**

**@Test(priority=4)**

**void deleteUser()**

**{**

**given()**

**.when()**

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

**.then()**

**.statusCode(204)**

**.log().all();**

**}**

**}**