What is API?

**API** (*Application Programming Interface*) helps in communication and data exchange between two software systems. API act as an interface between two applications and allows the two software systems communicate with one another. API is a collection of functions which can be executed by another software program.

API works as; it takes a request from the source, takes that request to the database, fetches the request data from the database and returns a response to the source. API takes the requests from the user and gives the response without exposing the internal details. API acts as Abstraction.

**Example:** Amazon API, Google Map API

ADVERTISEMENT

2) What is API testing?

**API testing** is a type of software testing that involves testing APIs directly. API is a part of integration testing to check whether the API meets expectations in terms of functionality, reliability, performance, and security of applications. Multiple API system can performed API testing. In API testing, our primary focus is on Business Logic Layer of the software architecture.

3) What are the types of API testing?

API testing involves the following types of testing:

ADVERTISEMENT

* Unit Testing
* Functional Testing
* Load Testing
* Runtime/Error Detection
* Security Testing
* UI Testing
* Interoperability and WS compliance Testing
* Penetration Testing
* Fuzz Testing

4) What are the protocols used in API Testing?

Protocols used in API testing are:

* HTTP
* REST
* SOAP
* JMS
* UDDI

5) What are the tools used for API Testing?

Tools used for API testing are:

* Parasoft SOAtest
* PostMan
* AlertSite API monitoring

6) What is API test environment?

For API the test environment is a quite complex method where the configuration of server and database is done as per the requirement of the software application. API testing does not involve graphical user interface (GUI).

API is checked for its proper functioning after installation.

ADVERTISEMENT

ADVERTISEMENT

7) What is API framework?

API framework is described by the config. File which consist of the list of all APIs that are required to be activated and are activated for any particular program run. This is essential as every test run does not require all APIs.

8) What are the limits of API usage?

Many APIs have certain limit set up by the provider. Hence, try to estimate our usage and understand how that will impact the overall cost of the offering.

9) What are the advantages of API testing?

Advantages of API testing are:

* **Test for core functionality:** API testing provides access to the application without the user interface. The core functionality of the application will be tested before the GUI tests. This will help to detect the minor issue which can become bigger during the GUI testing.
* **Time effective:** API testing is less time consuming than GUI testing. Particularly, API test requires less code so it can provide better and faster test coverage compare to GUI test automation. This will reduce the cost for the testing project.
* **Language Independent:** In API testing data is exchange using XML or JSON. These transfer mode are completely language-independent, which allows users to select any code language when adopting automation test service for the project.
* **Easy Integration with GUI:** API tests provide highly integrable tests which is useful to perform functional GUI tests after GUI tests. Simple integration would allow new user accounts to be created within the application before GUI started.

10) What are the principles of an API test design?

Here, are the seven principles of API test design.

1. **Exhaustive Testing:** Exhaustive testing is not possible. Instead we need optimal amount of testing which is based on the risk assessment of the application.
2. **Defect Clustering:** Defect Clustering states that a small number of modules contain the most of the defect detected. Approximately 80% of the defect found in 20% of the modules. By experience we can identify such risky modules. But this approach has its own problems. If the same tests are repeated over and over again, eventually the same test case will no longer find new bugs.
3. **Pesticide Paradox:** Testers cannot depend on existing technique. They must have to look continually to improve the existing method to make testing more effective. But even all these hard work in testing we can never claim our product is bug free. To overcome this, test cases need to be regularly reviewed and revised add new and different test cases to help find more defects.
4. **Testing shows presence of defects:** Testing principle states that- testing talks about the presence of defects not about the absence of defect. Software testing reduces the probability of undiscovered defects remaining in the software but even if no defects found, it is not a proof of correctness.

But if we work hard, taking all precautions and make our software products 99% bug free. The software does not meet the needs and requirements of the client.

1. **Absence of error -fallacy:** This can be possible the software which is 99% bug free is still unusable. The case can be if the system is tested for the wrong requirement. Software testing is not finding the defects but also to check that software addresses the business needs. The absence of error is fallacy i.e. finding and fixing defects does not help if the system build is unusable and doesn't fulfill the user's needs and requirements.
2. **Early Testing:** Testing should start as soon as possible in the software development lifecycle. So that defects in the requirement or design phase captured in the early stages. It is cheaper to fix defect in the early stages of testing. We should start finding the bug at the moment the requirements are defined.
3. **Testing is context dependent:** Testing is context dependent that we test an e-commerce site will be different from the way we test the commercial. All the developed software's are not identical. We will use different methodology; techniques and type of testing depend on the application type.

11) What is API framework?

A framework or software framework is a platform for developing software applications. API framework is a foundation on which software developer can build applications for a specific platform.

**Example:** A framework can include predefined classes and functions that can be used to process input, manage hardware devices and interact with system software.

Framework is similar to an Application Programming Interface, technically framework includes API. Framework serves foundation for programming while API provides access to the elements supported by the framework. Framework also includes code libraries, compiler and other programs used in the software development process.

ADVERTISEMENT

ADVERTISEMENT

API framework is defined by configuration file which consists the list of all APIs that is required to be activated and activated for a particular program run.

12) What are the common tests that performed on API?

Here, are the common tests that performed on API are as:

1. Response of the API should be verified based on the request. We will verify that the return value is based on request.
2. When API is updating any data structure we should verify the system is authenticating the outcome.
3. We will verify whether the API is trigger other event or request another API.
4. We will verify the behavior of the API when no value is return.

13) What exactly needs to verify in API testing?

In API testing, we send a request to API with the known data and then analysis the response.

ADVERTISEMENT

ADVERTISEMENT

1. We will verify the accuracy of the data.
2. Will see the HTTP status code.
3. We will see the response time.
4. Error codes in case API returns any errors.
5. Authorization would be check.
6. Non-Functional testing such as performance testing, security testing.

14) What are the differences between API and Web Services?

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **API** | **Web Services** |
| **1.** | API may or may not need network for its operations. | Web Services always need network for its operation. |
| **2.** | API can be communicated through SOAP, REST, XML-RPC and CURL calls as well. API can also be exposed in number of ways like JAR, DLL, XML over HTTP, JSON over HTTP etc. | Web service can be communicated through SOAP, REST, AND RPC. |
| **3.** | API can perform all the operations which web service can't perform. | Web service can't perform all the operations like API. |
| **4.** | All APIs are not web service. | All web services are API |

15) What is API documentation?

A good documentation is must for any foundation. API documentation serves as quick reference for accessing library or working within a program.

When we use any such documents, it must consists of proper plan, content source, proper layout, information related to each function etc.

ADVERTISEMENT

There are various documentation tools like Doxygen and JavaDoc. Here, are the functions which are documented which revolve around the parameters like:

* Function description
* Type and syntax of error message that may occure
* Syntax, elements and sequence needed for each parameter
* Links regarding functions

16) What is the most used template for API documentation?

Here, are the various documentation template that make the whole process simple and easy. They are:

* Swagger
* Miredot
* Slate
* FlatDoc
* API blueprint
* RestDoc
* Web service API specification

17) What are the types of bug that can be found during API testing?

API testing helps us to find many types of bugs which are:

* Stress
* Security
* Duplicate or missing functionality
* Reliability
* Unused flags
* Performance
* Incompatible error handling
* Multi-threaded issue
* Improper errors

18) What are the difference between API testing and UI testing?

UI (User Interface) testing means the testing of the graphical user interface. The focus of UI testing is on the look and feel of the application. In user interface testing the main focus is on how users can interact with app elements such as images, fonts, layout etc. are checked.

API testing allows the communication between two software systems. API testing works on backend also known as backend testing.

19) What is SOAP?

SOAP (Simple Object Access Control) . It is an XML based protocol that helps in exchanging information among computers.

20) What is REST API?

ADVERTISEMENT

ADVERTISEMENT

**REST API** is a set of function helps the developers performing requests when the response is receiving. Through HTTP protocol interaction is made in REST API.

REST is defined as Representational state transfer. It is an effective standard for API creation.

21) What are the differences between SOAP and REST API?

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **SOAP API** | **REST API** |
| **1.** | **SOAP** stands as Simple Object Access Protocol. | **REST** stands as Representational State Transfer. |
| **2.** | **SOAP** is a protocol. | **REST** is an architectural pattern. |
| **3.** | **SOAP** can work with XML format. In SOAP all the data passed in XML format. | **REST** permit different data format such as Plain text, HTML, XML, JSON etc. But the most preferred format for transferring data is in JSON. |

22) What are the major challenges faced during API testing?

The major challenges faced during the API testing are:

* Parameter Selection
* Parameter Combination
* Call sequencing
* Output verification and validation
* A major challenge is providing input values which are very difficult because GUI is not available.

23) What are the difference between API Testing and Unit Testing?

Difference between API testing and Unit testing are:

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **API Testing** | **UNIT Testing** |
| **1.** | API testing is a form of black box testing. | Unit testing is a form of white box testing. |
| **2.** | API testing is performed after the project completion during the test. | Unit testing is performed when the project is created. |
| **3.** | In API testing there is a wide scope of testing. | In Unit testing there is a limited scope of testing we can test only the basic functionality. |
| **4.** | API testing is done by the testers. The whole purpose of API testing is end to end testing of the functionality. | Unit testing is done by the developer. In unit testing every functionality is separately tested. |

ADVERTISEMENT

ADVERTISEMENT

24) What is a RESTFUL web services?

There are two kinds of web services

1. SOAP Web Services
2. RESTFUL Web Services

**1. SOAP (Simple Object Access Protocol) -** SOAP is a XML based method which is used in Web Services.

**2. RESTFUL Web Services -** To implement the concept of REST architecture HTTP method is used. RESTFUL Web Services defines URI (Uniform Resource Identifier), and also provides resource representation like JSON and a set of HTTP method.

ADVERTISEMENT

ADVERTISEMENT

25) What is Resource in REST?

REST architecture treats any content as resource, which can be text files, HTML pages, images, videos or dynamic business information. REST server gives the functionality to access the resources and modifies them. We can identify the each resources by URIs/ global IDs.

26) What is the way to represent the resource in REST?

REST uses different representation to define the resources like text, JSON and XML. The most popular representation of resources is JSON and XML.

27) What protocol is used by the RESTFUL Web Services?

RESTFUL Web Services uses the HTTP protocol. They use the HTTP protocol as a medium of communication between the client and the server.

28) What are the characteristics of REST?

Here, are the two characteristics of REST.

1. REST is stateless. With the use of the REST API the server has no status, we can restart the server between two calls, inspite of all the data is transferred to the server.
2. Web Services uses POST method to perform operations, while REST uses GET method to access the resources.

29) What is messaging in RESTFUL Web Services?

RESTFUL Web Services use the HTTP protocol as a communication tool between the client and the server. This is the technique when the client sends a message in the form of HTTP request the server send back the HTTP reply which is called Messaging. This message consists message data and Meta data i.e. information on the message itself.

30) What are the components of an HTTP request?

An HTTP request have five components. These are:

ADVERTISEMENT

ADVERTISEMENT

1. **Action showing HTTP method** like GET, PUT, POST, DELETE.
2. **Uniform Resource Identifier (URI):** URI is the identifier for the resource on the server.
3. **HTTP version:** Indicate the HTTP version like- HTTP V1.1.
4. **Request Header:** Request Header carries metadata for the HTTP request message. Metadata could be a client type, format supported by the client, format of a message body, cache setting etc.
5. **Request Body:** Resource body indicates message content or resource representation.

31) What is the HTTP protocol supported by REST?

**GET:** GET is used to request data from the specified resource.

GET request can be cached and bookmark. It remains in the browser history and has length restriction. When dealing with sensitive data GET requests should not be used.

**POST:** POST is used to send data to server for creation or updating the resources.

POST requests are never cached or bookmark.

**PUT:** PUT replaces the current representation of the target resource with the request payload.

**DELETE:** DELETE removes the specified resource.

**OPTIONS:** OPTION is used to describe the communication option for the target resources.

**HEAD:** HEAD asks for response which is identical to GET requests, but without the response body.

32) Can we use GET request instead of PUT to create a resource?

PUT or POST method is used create a resource. GET is only used to request the resources.

33) What is URI? What is the purpose of web-based service and what is it's format?

URI stands for Uniform Resource Identifier. It is a string of characters designed for unambiguous identification of resources and extensibility by the URI scheme. The purpose of URI is to locate the resource on the server hosting of the web service.

A URIs format is ***<protocol>://<service-name>/<Resource Type>/<ResourceID>***

34) What are SOAP Web Services?

**SOAP** (*Simple Object Access Protocol*) is defined as the XML based protocol. SOAP is also known for developing and designing web services and also enable the communication between the applications developed on different platform by using different programming languages on the internet. SOAP is platform and language independent.

35) When we can use SOAP API?

We can use SOAP API to perform the operation on records like create, retrieve, update or delete. We can use API to manage password, perform searches etc.

## **Rest Assured Tutorial Outline**

In this post, I will explain what is **API** and **API testing**, what is the difference between **SOAP** and **REST** services, and how to test REST APIs with **Rest Assured Library**.

## **What is API?**

**API** stands for **A**pplication **P**rogramming **I**nterface. It comprises a set of functions that can be accessed and executed by another software system.  Thus, it serves as an **interface** between different software systems and establishes their interaction and data exchange.

## **What is API Testing?**

In the modern development world, many web applications are designed based **on a three-tier architecture model**. These are **1)** **Presentation Tier** – User Interface (**UI**) **2) Logic Tier** – Business logic is written in this tier. It is also called Business Tier. (**API**) **3) Data Tier** – Here information and data are stored and retrieved from a Database. (**DB**) Ideally, these three layers (tiers) should not know anything about the platform, technology, and structure of each other.

We can test **UI with GUI testing tools** and we can test **logic tier (API) with API testing tools**. The logic tier comprises all of the **business logic** and it has **more complexity than the other tiers** and the **test executed on this tier is called API Testing**. **API testing tests the logic tier directly and checks expected functionality, reliability, performance, and security.**

In the agile development world, requirements are changing during short release cycles frequently and GUI tests are more difficult to maintain according to those changes. Thus, API testing becomes critical to test application logic. In GUI testing we send inputs via keyboard texts, button clicks, drop-down boxes, etc., on the other hand in **API testing we send requests (method calls) to the API and get output (responses).** These APIs are generally REST**APIs** or **SOAP web services** with **JSON** or **XML** message payloads being sent over HTTP, HTTPS, JMS, and MQ.

[Ref: wiki] I don’t want to go into the theory so much in this post. If you want to learn more theory of API testing, you can visit the below websites.

<https://www.soapui.org/testing-dojo/world-of-api-testing/what-makes-api-testing-special-.html>

<https://en.wikipedia.org/wiki/API_testing>

## **REST vs SOAP**

### ****REST (Representational State Transfer)****

**REST** is an architectural style that uses simple HTTP calls for inter-machine communication. REST does not contain an additional messaging layer and focuses on design rules for creating **stateless** **services**. A client can access the resource using the **unique URI** and **a representation of the resource is returned**. With each new resource representation, the client is said to transfer state. While accessing RESTful resources with **HTTP protocol**, the URL of the resource serves as the resource identifier, and **GET, PUT, DELETE, POST and HEAD** are the standard HTTP operations to be performed on that resource. [1][2][6]

### ****SOAP (Simple Object Access Protocol)****

**SOAP** relies heavily on **XML**, and together with schemas, defines a very **strongly typed messaging framework**. Every operation the service provides is explicitly defined, along with the **XML structure** of the **request** and **response** for that operation. Each input parameter is similarly defined and bound to a type: for example, an integer, a string, or some other complex object. All of this is codified in the **WSDL – Web Service Description** (or Definition, in later versions) Language.

The **WSDL** is often explained as a **contract between the provider and the consumer of the service**. SOAP uses different transport protocols, such as **HTTP** and **SMTP**. The standard protocol **HTTP makes it easier for SOAP** model to tunnel across firewalls and proxies without any modifications to the SOAP protocol. [3][4][6]

**References**:

<http://searchsoa.techtarget.com/definition/REST> [1]

<http://blog.pluralsight.com/representational-state-transfer-tips> [2]

[https://www.soapui.org/testing-dojo/world-of-api-testing/soap-vs–rest-challenges.html](https://www.soapui.org/testing-dojo/world-of-api-testing/soap-vs--rest-challenges.html) [3]

<http://blog.smartbear.com/apis/understanding-soap-and-rest-basics/> [4]

<http://spf13.com/post/soap-vs-rest> [5]

<http://searchsoa.techtarget.com/tip/REST-vs-SOAP-How-to-choose-the-best-Web-service> [6]

## **REST API Testing with Rest Assured**

### ****What is Rest Assured?****

In order to test REST APIs, I found the **REST Assured library** so useful. It is developed by JayWay Company and it is a really powerful catalyzer for automated testing of **REST services**. REST-assured provides a lot of nice features, such as **DSL-like syntax**, **XPath-Validation**, **Specification Reuse**, **easy file uploads,** and with those features, we will handle automated API testing much easier. Rest Assured has **a gherkin type syntax** which is shown below code. If you are a fan of **BDD**(**B**ehavior **D**riven **D**evelopment), I believe that you will love this kind of syntax.

@Test

**public** void exampleRestTest**()** **{**

given**()**

.contentType**(**ContentType.JSON**)**

.pathParam**(**"id", "AskJsd8Sd"**)**

.when**()**

.get**(**"/examplepath/{id}"**)**

.then**()**

.statusCode**(**200**)**

.body**(**"firstName", equalTo**(**"Onur"**))**

.body**(**"Surname", equalTo**(**"Baskirt"**))**;

**}**

Also, you can **get JSON response** as a string and **send it to the JsonPath class** and use its methods to write more structured tests. I generally prefer JsonPath for more structured tests.

@Test **public** void exampleJsonPathTest**()** **{**

Response res = given**()**.get**(**"/service/example"**)**;

assertEquals**(**200, res.getStatusCode**())**;

String json = res.asString**()**;

JsonPath jp = new JsonPath**(**json**)**;

assertEquals**(**"onur@swtestacademy", jp.get**(**"email"**))**;

assertEquals**(**"Onur", jp.get**(**"firstName"**))**;

assertEquals**(**"Baskirt", jp.get**(**"lastName"**))**;

**}**

## **How to Make a POST Request with RestAssured?**

The following code uses **requestSpecBuilder** to make a post request. Parameter descriptions are listed below.

* **restAPIURL** – URL of the Rest API
* **APIBody** – Body of the Rest API. **Example**: {“key1″:”value1″,”key2″:”value2”}
* **setContentType()** – Pass the “application/json”, “application/xml” or “text/html” etc. headers to setContenType() method.
* **Authentication credentials** – Pass the username and password to the basic() method or if there is no authentication leave them blank basic(“”,””)

@Test

**public** void httpPostMethod**()** **throws** JSONException, InterruptedException **{**

//Rest API's URL

String restAPIUrl = "http://{URL of API}";

//API Body

String apiBody = "{\"key1\":\"value1\",\"key2\":\"value2\",\"key3\":\"value3\"}";

// Building request by using requestSpecBuilder

RequestSpecBuilder builder = new RequestSpecBuilder**()**;

//Set API's Body

builder.setBody**(**apiBody**)**;

//Setting content type as application/json

builder.setContentType**(**"application/json; charset=UTF-8"**)**;

RequestSpecification requestSpec = builder.build**()**;

//Making post request with authentication or leave blank if you don't have credentials like: basic("","")

Response response = given**()**

.auth**()**

.preemptive**()**

.basic**({** username **}**, **{** password **})**

.spec**(**requestSpec**)**

.when**()**

.post**(**restAPIUrl**)**;

JSONObject JSONResponseBody = new JSONObject**(**response.body**()**.asString**())**;

//Get the desired value of a parameter

String result = JSONResponseBody.getString**({** key **})**;

//Check the Result

Assert.assertEquals**(**result, "{expectedValue}"**)**;

**}**

### ****Gherkin Style Rest Assured POST Request****

@Test

**public** void postExamplebyGherkin**(){**

RestAssured.baseURI = "Your API URL";

Response res = given**()**

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

.body**(**"{\"name\":\"Onur Baskirt\"}"**)**

.when**()**

.post**(**""**)**;

String body = res.getBody**()**.asString**()**;

System.out.println**(**body**)**;

**}**

For more theory, please visit references. I want to go with **example questions and their solutions**.

***References:***

<https://rest-assured.io/>[1] (**Rest Assured GitHub page**)

<http://www.hascode.com/2011/10/testing-restful-web-services-made-easy-using-the-rest-assured-framework/> [2]

[http://static.javadoc.io/com.jayway.restassured/rest-assured/1.6.1/com/jayway/restassured/path/json/JsonPath.html](http://static.javadoc.io/com.jayway.restassured/rest-assured/1.6.1/com/jayway/restassured/path/json/JsonPath.html%20%5b4)[3]

<http://www.javahotchocolate.com/notes/rest-assured.html> [4]

<http://artoftesting.com/automationTesting/restAPIAutomationPostRequest.html>[5]

<https://automationqahub.com/how-to-test-graphql-using-rest-assured/> [6] For GraphQL Testing with RestAssured.

## **API Testing Automation Examples**

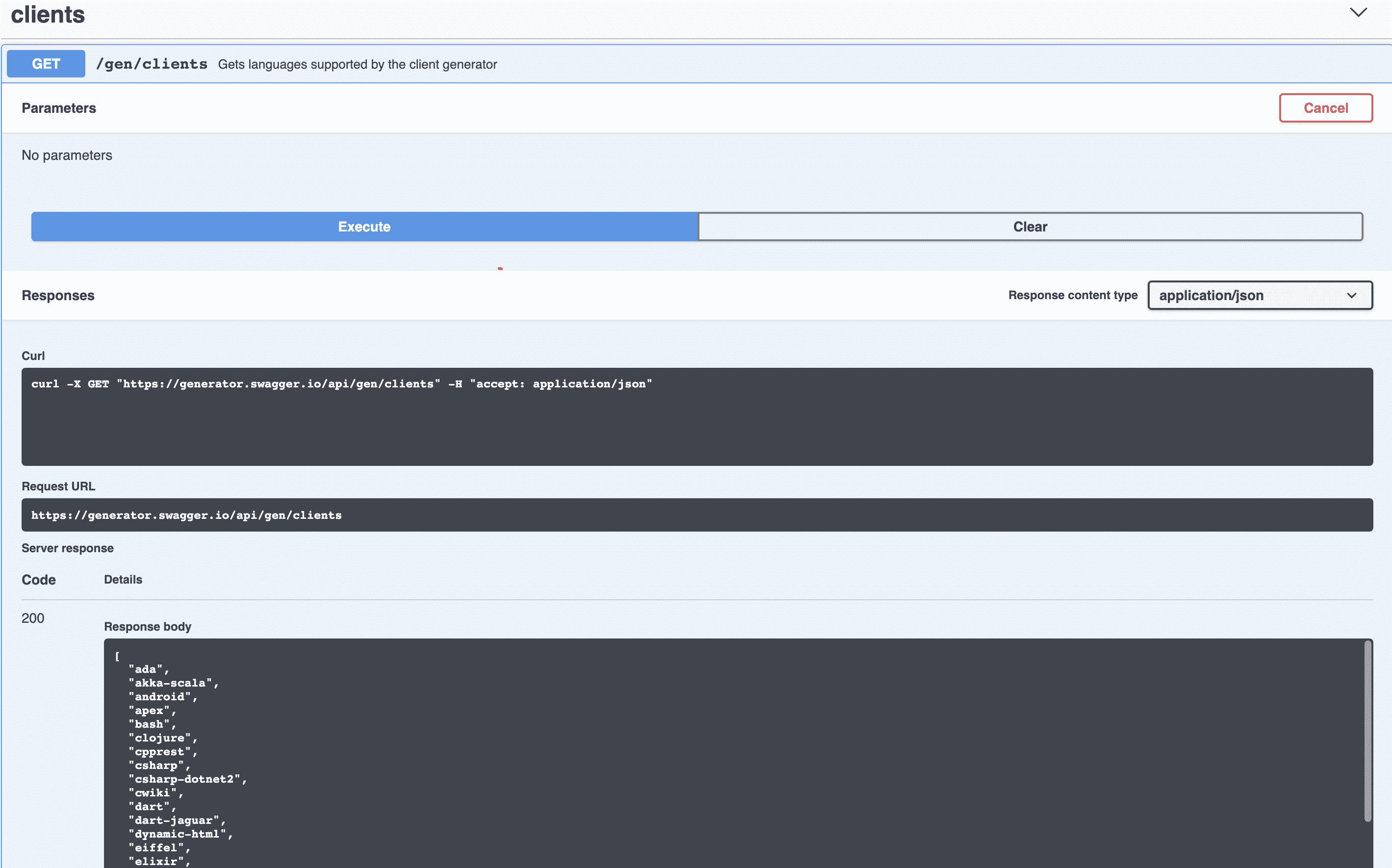
### ****Example-1:****

**Test Description:**Get the clients from [**https://generator.swagger.io/**](https://generator.swagger.io/). We will do a GET request to get and print all clients.

**Base URL:** https://generator.swagger.io/

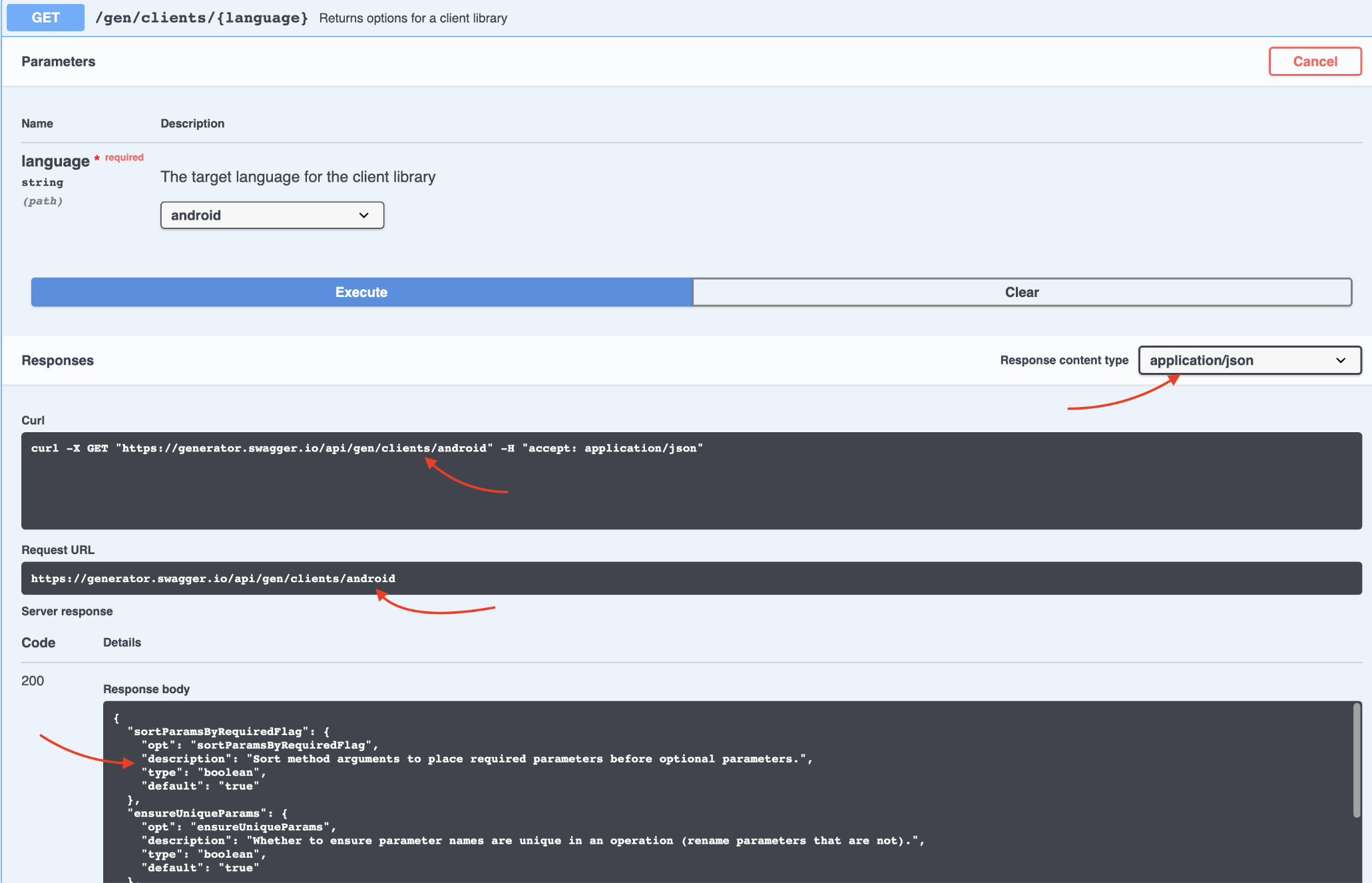
**Resource Path:**/gen/clients

We will verify the status and print the clients which we get from the API.

[](https://www.swtestacademy.com/wp-content/uploads/2019/10/Pasted-into-Rest-Assured-Tutorial-Learn-API-Testing-Step-by-Step.png)

### ****Example-2:****

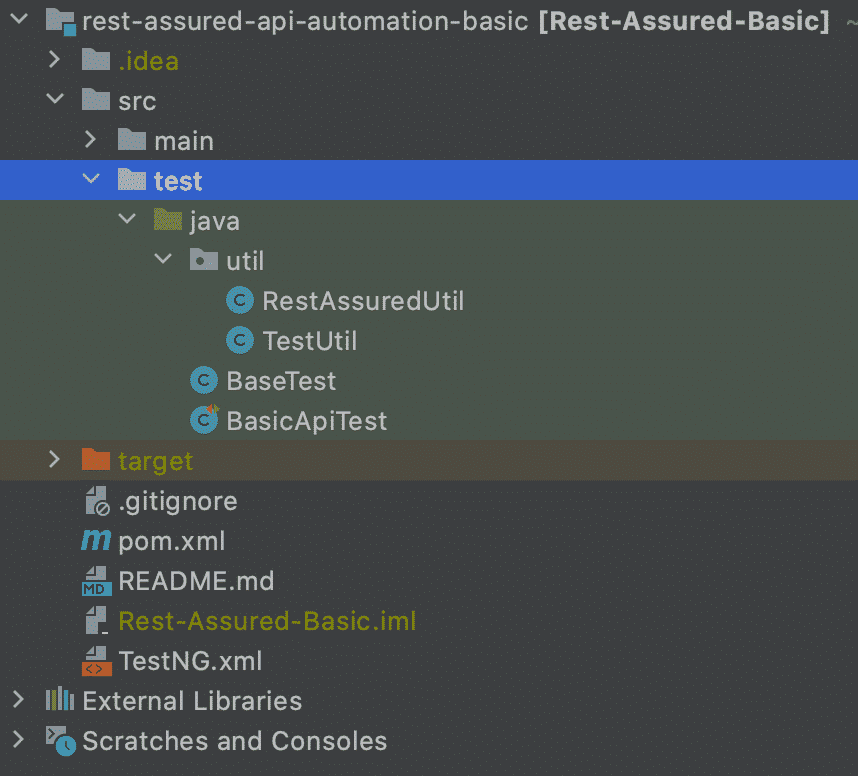
**Test Description:**Get android clients. Then, check the status and print **modelPackage.opt**, **modelPackage.description**, and **modelPackage.type** values.

[](https://www.swtestacademy.com/wp-content/uploads/2019/10/Pasted-into-Rest-Assured-Tutorial-Learn-API-Testing-Step-by-Step-2.png)

## **API Testing Automation Solutions**

**Strategy**: First, it is very reasonable to use a framework/library which provides us to test an API easily in a short period of time and we chose the **Rest-assured library**. It is better to write the code with the below rules.

**Project Structure:**

[](https://www.swtestacademy.com/wp-content/uploads/2019/10/Pasted-into-Rest-Assured-Tutorial-Learn-API-Testing-Step-by-Step-4.png)

We have utility functions and test classes.

BasicAPiTest contains our test implementations.

I used TestNG as a test runner framework and all dependencies are in pom.xml file.

### ****pom.xml****

<?xml version="1.0" encoding="UTF-8"?>

**<project** xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>**

**<modelVersion>**4.0.0**</modelVersion>**

**<groupId>**api-automation-basic**</groupId>**

**<artifactId>**api-automation-basic**</artifactId>**

**<version>**1.0-SNAPSHOT**</version>**

**<properties>**

**<rest-assured-version>**4.4.0**</rest-assured-version>**

**</properties>**

**<dependencies>**

**<dependency>**

**<groupId>**org.testng**</groupId>**

**<artifactId>**testng**</artifactId>**

**<version>**7.4.0**</version>**

**<scope>**test**</scope>**

**</dependency>**

**<dependency>**

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

**<artifactId>**rest-assured**</artifactId>**

**<version>**${rest-assured-version}**</version>**

**<scope>**test**</scope>**

**</dependency>**

**<dependency>**

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

**<artifactId>**rest-assured**</artifactId>**

**<version>**${rest-assured-version}**</version>**

**<scope>**compile**</scope>**

**</dependency>**

**<dependency>**

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

**<artifactId>**json-path**</artifactId>**

**<version>**${rest-assured-version}**</version>**

**</dependency>**

**<dependency>**

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

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

**<version>**${rest-assured-version}**</version>**

**</dependency>**

**</dependencies>**

**<build>**

**<plugins>**

**<plugin>**

**<groupId>**org.apache.maven.plugins**</groupId>**

**<artifactId>**maven-compiler-plugin**</artifactId>**

**<configuration>**

**<source>**11**</source>**

**<target>**11**</target>**

**</configuration>**

**</plugin>**

**</plugins>**

**</build>**

**</project>**

### ****TestNG.xml****

<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >

**<suite** name="API Testing Suite" verbose="1"**>**

**<test** name="Nopackage"**>**

**<classes>**

**<class** name="BasicApiTest"**/>**

**</classes>**

**</test>**

**</suite>**

I tried to explain the other codes line by line. Now I want to go on with the project’s JAVA files.

### ****RestAssuredUtil.java****

**RestAssuredUtil.java**It is a **utility class for Rest Assured Library**. It contains many methods that help us to write our codes more effectively.

**package** utils;

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

**import** io.restassured.RestAssured;

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

**import** io.restassured.path.json.JsonPath;

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

**public** **class** RestAssuredUtil **{**

//Sets Base URI

**public** **static** void setBaseURI**()** **{**

RestAssured.baseURI = "http://generator.swagger.io/";

**}**

//Sets base path

**public** **static** void setBasePath**(**String basePathTerm**)** **{**

RestAssured.basePath = basePathTerm;

**}**

//Reset Base URI (after test)

**public** **static** void resetBaseURI**()** **{**

RestAssured.baseURI = **null**;

**}**

//Reset base path

**public** **static** void resetBasePath**()** **{**

RestAssured.basePath = **null**;

**}**

//Sets ContentType

**public** **static** void setContentType**(**ContentType Type**)** **{**

given**()**.contentType**(**Type**)**;

**}**

//Returns response by given path

**public** **static** Response getResponse**(**String path**)** **{**

**return** given**()**.get**(**path**)**;

**}**

//Returns response

**public** **static** Response getResponse**()** **{**

**return** given**()**.get**()**;

**}**

//Returns JsonPath object

**public** **static** JsonPath getJsonPath**(**Response res**)** **{**

String json = res.asString**()**;

**return** new JsonPath**(**json**)**;

**}**

**}**

### ****TestUtil.java****

We will get the clients and check the status code in this class.

**package** util;

**import** io.restassured.path.json.JsonPath;

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

**import** java.util.ArrayList;

**import** org.testng.Assert;

**public** **class** TestUtil **{**

//Verify the http response status returned. Check Status Code is 200?

**public** void checkStatusIs200**(**Response res**)** **{**

Assert.assertEquals**(**res.getStatusCode**()**, 200, "Status Check Failed!"**)**;

**}**

//Get Clients

**public** **<**T**>** ArrayList**<**T**>** getClients**(**JsonPath jp**)** **{**

**return** jp.get**()**;

**}**

**}**

### ****BasicApiTest.java****

We have some basic API tests in our test class.

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

**public** **class** BasicApiTest **extends** BaseTest **{**

@Test

**public** void T01\_StatusCodeAndGetClientsTest**()** **{**

res = utils.RestAssuredUtil.getResponse**(**"/gen/clients"**)**;

testUtil.checkStatusIs200**(**res**)**;

jp = utils.RestAssuredUtil.getJsonPath**(**res**)**;

System.out.println**(**testUtil.getClients**(**jp**))**;

**}**

@Test

**public** void T02\_GetAndroidModelPackageOptions**()** **{**

res = utils.RestAssuredUtil.getResponse**(**"/gen/clients/android"**)**;

testUtil.checkStatusIs200**(**res**)**;

jp = utils.RestAssuredUtil.getJsonPath**(**res**)**;

System.out.println**(**"Opt: " + jp.get**(**"modelPackage.opt"**))**;

System.out.println**(**"Description: " + jp.get**(**"modelPackage.description"**))**;

System.out.println**(**"Type: " + jp.get**(**"modelPackage.type"**))**;

**}**

**}**

Let’s run our tests by right-clicking the TestNG.xml file and t

<https://www.tutorialspoint.com/how-to-use-assertion-in-response-in-rest-assured>