Rest Assured Tutorial – Learn API Testing Step by Step

**Rest Assured** is one of the most popular libraries which is highly used in **API Test Automation** in most of the companies. In this Rest Assured tutorial, I will try to explain **Rest API**, **API Testing**, **API Automation**, **REST**, and **SOAP** protocols.

**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 of 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 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 is 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**. Logic tier comprises of all of the **business logic** and it has **more complexity than the other tiers** and the **test executed on this tier is called as API Testing**.

**API testing tests 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 theory so much in this post. If you want to learn more theory on API testing, you can visit below websites.

<https://www.soapui.org/testing-dojo/world-of-api-testing/what-makes-api-testing-special-.html>  
<http://www.guru99.com/api-testing.html> (*Very introductive article*.)  
<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 **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 in 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.

exampleRestTest

Java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | *@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.

exampleJsonPathTest

Java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | *@Test*  public void exampleJsonPathTest() {    Response res = 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(“”,””)

Java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | *@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().authentication().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**

Java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | *@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);    } |