# **RestAssured Mock**

# **What is SOA**

Web applications traditionally designed in Client Server fashion, where a web browser as a client will send request to the server and the server will respond to the client. Data is transferred throughout these request.

There are two types of webservices SOAP and RESTful. A large application like Google or Facebook can have multiple webservices communicating with each other for processing data. So the systems/applications built using these webservices can be called as Service Oriented Architecture.

# **What are Rest API's and explain caching:**

REST is the underlying architectural principle of the web. The important thing about the web is the fact that clients (browsers) and servers can interact in complex ways without the client knowing anything beforehand about the server and the resources it hosts.

Rest API's are stateless and cacheable. They don't store information. REST is implemented in XML as well as JSON. Conversion of JSON objects into Java objects is very easy. REST architecture follows the CRUD ( Create, Read, Update and Delete ) Style where Create means POST request, Read means GET request, Update means PUT request and Delete request.We can communicate with REST API with above four requests.

**Caching** is the process in which server response is stored so that a cached copy can be used when required and there is no need of generating the same response again. This process not only reduces the server load but in turn increase the scalability and performance of the server. Only the client is able to cache the response and that too for a limited period of time. It is the ability to store copies of frequently accessed data in several places along the request-response path.

Optimizing the network using caching improves the overall quality-of-service in following ways:

* Reduce bandwidth
* Reduce latency
* Reduce load on servers
* Hide network failures

Usually browsers treat all GET requests cacheable. POST requests are not cacheable by default but can be made cacheable if either an Expires header, or a Cache-Control header with a directive that explicitly allows caching, is added to the response. Responses to PUT and DELETE requests are not cacheable at all.

# **What is BDD and its advantages.**

Behavior Driven testing is an extension of TDD. Like in TDD in BDD also we write tests first and the add application code. The major difference that we get to see here are

* Tests are written in plain descriptive English type grammar
* Tests are explained as behavior of application and are more user focused
* Using examples to clarify requirements

This difference brings in the need to have a language which can define, in an understandable format.

**Features of BDD**

1. Shifting from thinking in “tests” to thinking in “behavior”
2. Collaboration between Business stakeholders, Business Analysts, QA Team and developers
3. Ubiquitous language, it is easy to describe
4. Driven by Business Value
5. Extends Test Driven Development (TDD) by utilizing natural language that non technical stakeholders can understand

# **Explain Rest Assured**

Rest Assured follows the BDD ( Behaviour Driven Development ) approach for writing tests i.e Given- When - Then structure.

In Given section we declare things like content type or request body.

In When section we provide HTTP method and endpoint.

In Then section we declare response verification.

The method hitting our first API method looks as follows:

@Test

**public** **void** get1() {

*given*().

when().

get(endpoint+"?authentication=false").

then().

statusCode(200);

}

# **Explain Authentication and Authorization in Rest Webservices**

**Authentication**

Authentication is a process to prove that you are the person who you intend to be.

Taking the example of email login, we know that in order to Authenticate our self we have to provide a Username and a Password. In a very basic Authentication flow using Username and Password, we will do the same thing in REST API call as well.

A REST request can have a special header called Authorization Header, this header can contain the credentials (username and password) in some form. Once a request with Authorization Header is received, server can validate the credentials and can let you access the private resources.

**Authorization**

Authorization is the process of giving access to someone. If you are Authorized then you have access to that resource. Now to Authorize you need to present credentials and as we discussed earlier that process is called Authentication. Hence Authorization and Authentication are closely related terms and often used interchangeably.

given().auth().basic(username, password).when().get("/uri/").then().statusCode(200);

# **how to validate JSON schema in RestAssured.**

Schema is the structure behind data organization. It is a visual representation of how different fields, values are organized in a JSON/XML file.

validation of schema means, checking whether our JSON response is in its standard structure or not, all the required fields are there or not, with the required data types.

We need to

import io.restassured.module.jsv.JsonSchemaValidator;

@Test

    public void SchemaValidationTest{

        given()

        .accept(ContentType.JSON)

        .when()

        .get("https://reqres.in/api/users/")

        .then()

        .assertThat()

        .body(JsonSchemaValidator.matchesJsonSchemaInClasspath("schemaFile.json"))

        .statusCode(is(HttpStatus.SC\_OK));

}

JSON Schema validator is not part of REST Assured core so in order to use it you need to add the json-schema-validator module to classpath, for example by adding the following Maven dependency:

<dependency>

      <groupId>com.jayway.restassured</groupId>

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

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

      <scope>test</scope>

</dependency>

# **Explain GPath in REST Assured**

REST Assured uses GPath, a path expression language integrated into the Groovy language. It is similar to XPath for XML, but GPath can handle both XML and JSON.

It is used in RestAssured for

**Extracting a single element value**

@Test

public void extractAndCheckSingleValue() {

    given().

    when().

        get("<http://ergast.com/api/f1/2016/drivers.json>").

    then().

        assertThat().

        body("MRData.DriverTable.Drivers.driverId[-1]",equalTo("wehrlein"));

}

**Extracting a set of element values**

@Test

public void extractAndCheckMultipleValues() {

    given().

    when().

        get("<http://ergast.com/api/f1/2016/drivers.json>").

    then().

        assertThat().

        body("MRData.DriverTable.Drivers.driverId",hasItems("alonso","button"));

**Filtering values**  
Using GPath, you can also filter values to return an even more specific subset of values. For example, if we only want to return the collection of *permanentNumber* values in use this year that are between 20 and 30 inclusive, we can do this:

[?](https://www.ontestautomation.com/selecting-response-elements-with-gpath-in-rest-assured/)

|  |
| --- |
| @Test  public void extractAndCheckRange() {        given().      when().          get("<http://ergast.com/api/f1/2016/drivers.json>").      then().          assertThat().          body("findAll{Drivers->Drivers.permanentNumber >= \"20\" && Drivers.permanentNumber <= \"30\"}.permanentNumber",hasItem("22")).          and().          body("findAll{Drivers->Drivers.permanentNumber >= \"20\" && Drivers.permanentNumber <= \"30\"}.permanentNumber",not(hasItem("33")));  } |

}

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

<https://www.ontestautomation.com/selecting-response-elements-with-gpath-in-rest-assured/>

<http://groovy-lang.org/processing-xml.html>

<http://www.projectdebug.com/send-get-request-using-rest-assured/>

Why GPath is preferred over Jayway in RestAssured?

* REST Assured was created before Jayway's JsonPath
* REST Assured is built in Groovy and thus it was natural to take advantage of its data structures and its query language.
* GPath also works for other data formats such as XML and HTML which means that once you learn GPath you can apply to other types of data in an easy manner

# **How to make a GET call using**[**Rest-Assured**](https://github.com/jayway/rest-assured)**in java to a endpoint which requires certificate**

org.apache.http.conn.ssl.SSLSocketFactory clientAuthFactory = new org.apache.http.conn.ssl.SSLSocketFactory(keyStore, password);

// set the config in rest assured

config = new SSLConfig().with().sslSocketFactory(clientAuthFactory).and().allowAllHostnames();

RestAssured.config = RestAssured.config().sslConfig(config);

RestAssured.given().when().get("/path").then();