**1.How to Read and Validate JSON Response Body using Rest Assured**

RestAssured.*baseURI*= "https://fakestoreapi.com/products/1";  
String responseBody = *given*().get().getBody().asString();  
  
JsonPath js = new JsonPath(responseBody);  
String category = js.getString("category");  
String rate = js.getString("rating.rate");

Assert.asserEquals(category,”men's clothing”);

*Assert.assertEquals(rate,"3.9");*

**OR**

*given*().when().get("https://fakestoreapi.com/products/1")  
 .then().log().body().body("category", *equalTo*("men's clothing"))  
 .body("rating.rate", *equalTo*(3.9F));

**2.Authentication v/s Authorization**

**Authentication:**

Authentication is the process of verifying the identity of a user,

System or entity trying to access a particular resource to perform an action.

Popular Authentication Technics:

1. Password-based authentication
2. Certificate-based authentication
3. Biometric authentication

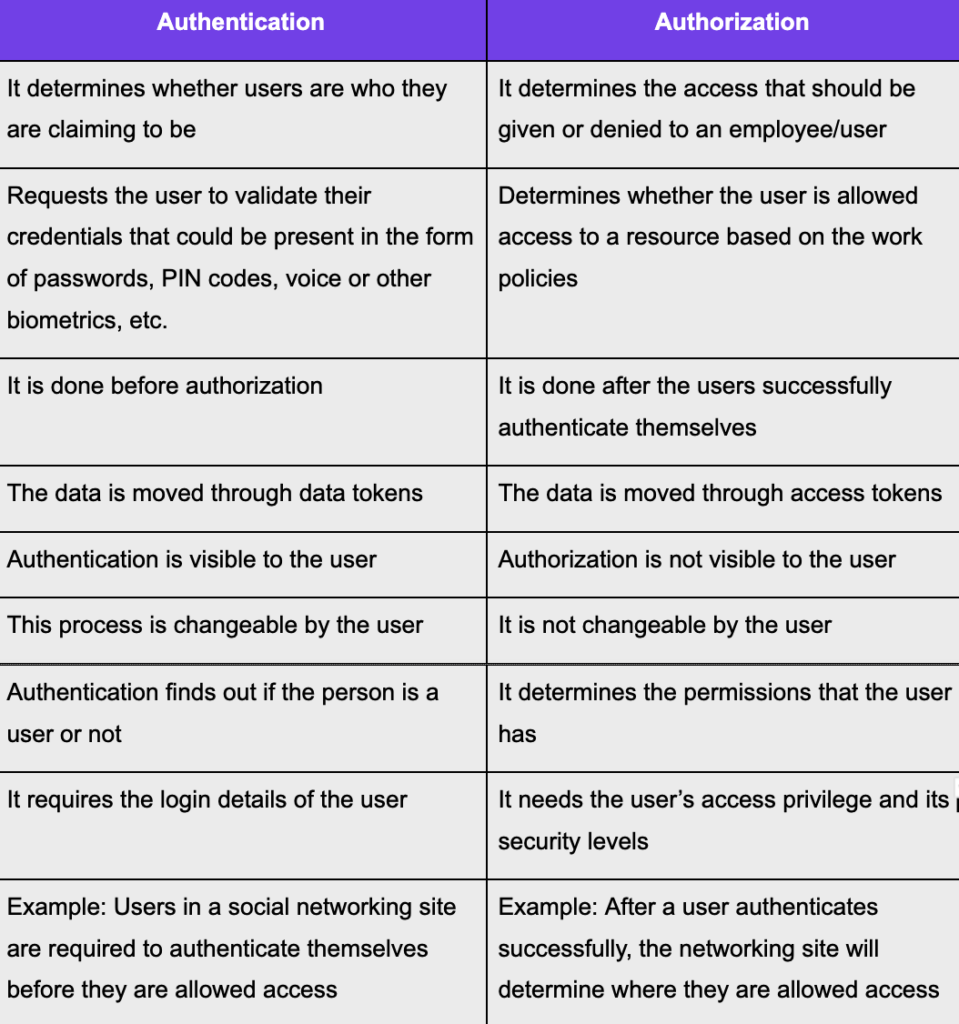
* **Fingerprint scanners:**
* **Facial recognition:**
* **Voice biometric recognition:**
* **Iris scanners:**

1. Token-based authentication
2. Single Sign-On (SSO)
3. Consumer Identity and Access Management (CIAM)

**Authorization:**

The process of giving permission to the user to access a specific function or resource is called authorization.

1. API Keys
2. Hash-based Message Authentication Code (HMAC)
3. OAuth
4. JWT



**What is REST Assured**

REST Assured is a Java library that enables developers to interact with RESTful web services and make HTTP requests without having to write complicated code. It provides a domain-specific language (DSL) that allows developers to write powerful, maintainable tests for their web services. With REST Assured, developers can easily perform assertions against response data, validate specific content types and responses codes, and even use XML Path Expressions and JSON Path Expressions to extract values from the response. REST Assured also provides a set of built-in assertions for validating the data returned by the web service. This library makes testing RESTful web services much easier by eliminating the need for developers to write complex code in order to make and validate HTTP requests. With its easy-to-use DSL, REST Assured can be used to efficiently test the functionality and performance of RESTful APIs.

**Why is it preferred to use REST-assured over Postman for automating RESTful services?**

REST Assured is a preferred tool for automating RESTful services due to its features, such as the ability to send requests quickly and validate the responses. With REST Assured, developers can easily make HTTP requests without having to write complex code and can also validate the responses against pre-defined criteria. Additionally, REST Assured provides a set of built-in assertions that can be used to check the data returned by the web service. Furthermore, REST Assured supports XML Path Expressions and JSON Path Expressions, which allows developers to extract values from the response and make assertions. Postman is a popular tool to test RESTful APIs, however, it lacks the ability to automate the tests and assertions. In contrast, REST Assured offers a powerful framework that enables automation, thus making it a preferred choice for automating RESTful services.

**What does the Request Specification object in Rest Assured do**

The Request Specification object in Rest Assured is used to configure and set properties of the request, such as the endpoint URL, query parameters, headers, authentication credentials, payloads, and many more. It is essential for setting up a request before making it. The Request Specification object is created using an instance of the RequestSpecBuilder class. The code snippet below shows how to use RequestSpecBuilder to create a Request Specification object.

RequestSpecBuilder builder = new RequestSpecBuilder();

builder.setBaseUri("https://example.com");

builder.addHeader("x-api-key", "abc123");

builder.setBody("{\"jsonData\": \"data\"}");

RequestSpecification reqSpec = builder.build();

**How can we send a POST Request in Rest Assured**

To send a POST request in Rest Assured, we need to setup and configure the request using the Request Specification object. This is done by setting the request URL, query parameters, headers, and payloads. After the request has been configured, the POST() method is used to make the actual API call. The code snippet below shows how to send a POST request using Rest Assured.

RequestSpecBuilder builder = new RequestSpecBuilder();

builder.setBaseUri("https://example.com");

builder.addHeader("x-api-key", "abc123");

builder.setBody("{\"jsonData\": \"data\"}");

RequestSpecification reqSpec = builder.build();

Response response = RestAssured.given(reqSpec).post();

**What is the best way to keep confidential data out of the log in rest assured**

The best way to keep confidential data out of the log in Rest Assured is to use the mask() method. This method hides sensitive information from the log and prevents it from appearing in the log files. The code snippet below shows how to use the mask() method to hide confidential data from the log.

RequestSpecBuilder builder = new RequestSpecBuilder();

builder.setBaseUri("https://example.com");

builder.addHeader("x-api-key", "abc123");

builder.setBody("{\"jsonData\": \"data\"}");

RequestSpecification reqSpec = builder.mask("body","sensitiveData","\*\*\*\*")

.build();

Response response = RestAssured.given(reqSpec).post();

**What is REST Assured?**

REST Assured is a Java library that provides a domain-specific language (DSL) for writing powerful, maintainable tests for RESTful APIs. This library allows you to make HTTP requests and validate the response against pre-defined criteria.  
  
**Provide a code example to test a REST API using Rest Assured**

Here is an example code snippet of a test that uses REST Assured to validate the response of a GET request to an API:

// Create a new RequestSpecification

RequestSpecification request = given().auth().oauth2("<access\_token>");

// Make a GET request with the specification

Response response = request.when().get("/api/endpoint");

// Validate the response

response.then().statusCode(200).assertThat().contentType(ContentType.JSON);

Here is an example written in Java that uses REST Assured to verify the status code, response times and content type of a given API endpoint:

public void testApi() {

// Set the base URL for the API

String baseUrl = 'http://myapi.com';

// Create a new RequestSpecification

RequestSpecification request = given().auth().oauth2("<access\_token>");

// Make a GET request with the specification

Response response = request.when().get(baseUrl + "/api/endpoint");

**What are the different methods to validate the response of a REST API in Rest Assured**

and code snippet REST Assured provides multiple methods for validating the response of a REST API. The most common methods are status code validation, content type validation, response time validation, and response body validation. For status code validation, you can use the statusCode() method to verify that the response code is the expected one. For example:

response.then().statusCode(200);

For content type validation, you can use the contentType() method to verify that the response has the expected content type. For example:

response.then().contentType(ContentType.JSON);

For response time validation, you can use the time() method to verify that the response time is within the expected range. For example:

response.then().time(lessThan(300L));

For response body validation, you can use the body() method to verify that the response body contains the expected information. For example:

response.then().body("id", equalTo("123"));

**What is the purpose of the Array slice operator in JsonPath in Rest Assured**

The Array slice operator in JsonPath allows you to retrieve elements from an array based on a start and end index. This is useful when you want to retrieve a specific subset of elements from an array without having to iterate over the entire array. For example, if you have an array with 10 elements and you only want to retrieve elements 3-7, you can use the Array slice operator to do this. The following code snippet shows how this could be done:

// Create a JsonPath object

JsonPath jsonPath = new JsonPath(jsonString);

// Retrieve the elements from index 3 to 7 (included)

List<Object> slice = jsonPath.read("$.array[3..7]");

The Array slice operator is very efficient and can be used to quickly retrieve elements from an array without the need for loops or other iteration techniques.

**How can we determine the size of a JSON array in Rest assured**

To determine the size of a JSON array in REST Assured, you can use the length() method. This method will return the number of elements in an array, allowing you to quickly and easily determine the size of the array. For example, if you have a JSON array with 10 elements, you can use the following code snippet to retrieve the size of the array:

// Create a JsonPath object

JsonPath jsonPath = new JsonPath(jsonString);

// Retrieve the size of the array

int size = jsonPath.read("$.array.length()");

This method is very efficient and can be used to quickly determine the size of a JSON array without the need for loops or other iteration techniques.

**How can we log errors when they occur in the response in Rest assured**

To log errors when they occur in the response, you can use the exceptionHandling() method in REST Assured. This method allows you to define a custom ExceptionHandler that will be called when an error occurs. For example, if you want to log the response code and the response body when an error occurs, you can use the following code snippet:

// Create a new ExceptionHandler

ExceptionHandler exceptionHandler = new ExceptionHandler() {

public void handle(RequestSpecification request, Response response) {

Logger.info(String.format("Error %s: %s", response.getStatusCode(), response.getBody().asString()));

}

};

// Set the exception handler to the RequestSpecification

RequestSpecification request = given().exceptionHandling(exceptionHandler);

Using the exceptionHandling() method, you can define a custom ExceptionHandler that will be called when an error occurs, allowing you to log the error so you can investigate it further.

**How to locate all employees ids from 15 to 300 using Rest Assured jsonPath**

You can use the Rest Assured jsonPath to locate all employees ids from 15 to 300 by using the code snippet below. The code creates a JsonPath object, finds all the elements in the array that has the id attribute greater than 15 and less than 300, then prints those elements out. Code Snippet:

JsonPath jsonPath = new JsonPath(json);

List<Integer> ids = jsonPath.get("employees[?(@.id > 15 && @.id < 300)].id");

System.out.println(ids);

**What is the purpose of method chaining in Rest Assured**

Method chaining in Rest Assured is a way to reduce the amount of code needed for making API requests and validating responses. It is used to chain together multiple calls in a single line, which makes the code more concise and easier to read. This can dramatically reduce the amount of code needed to send a request and validate its response. As an example, if you wanted to send a POST request to an endpoint and validate that it has returned a successful response, you could use method chaining as follows:

given().

body("{<json body>}").

when().

post("/endpoint").

then().

statusCode(200);

**What are jsonPath and how it is used in Rest Assured**

JSONPath is a lightweight query language used to query and extract data from JSON messages. It can be used with Rest Assured to locate specific elements within API responses. By using JSONPath, you can quickly extract values from API responses and validate the data that was returned. The following code snippet shows how JSONPath can be used with Rest Assured to extract a particular value from a response

body:

String responseBody = given().

body("{<json body>}").

when().post("/endpoint").

thenReturn()

.asString();

JsonPath jsonPath = new JsonPath(responseBody);

int id = jsonPath.get("employees.id");

System.out.println(id);

**How can we employ Request Specification in Rest Assured**

Request Specification in Rest Assured is an object used to set parameters and headers for a request prior to sending it. This makes it easier to build requests that have the same configuration across multiple tests. To create a request specification, you can use the given() method which allows you to configure headers, parameters, path parameters, cookies and authentication. The following code snippet shows how the given() method can be used to set the request type and parameters for a POST request:

RequestSpecification request = given().

contentType("application/json").

param("key1", "value1").

param("key2", "value2");

when().

post("/endpoint").

then().

statusCode(200);

**How can we log the request and response if validation failed in Rest Assured**

If validation fails in Rest Assured, you can log the request and response using the log() method. This method allows you to specify the type of log (request, response, or both) as well as the logging level. This is particularly useful when debugging failed tests by providing additional information about the request and response. The following code snippet shows how to use the log() method to log the request and response for a POST request:

given().

body("{<json body>}").

when().

post("/endpoint").

then().

log().all().

statusCode(200);

**How to use Path Variable with GET rest endpoint in Rest Assured**

Path variables can be used with GET requests in Rest Assured by passing in the variable name and value into the path() method. This allows you to make dynamic GET requests where the endpoint path is different for each request. The following code snippet shows how to use path() method with a variable containing an integer value:

int id = 5;

given().

pathParam("id", id).

when().

get("/endpoint/{id}").

then().

statusCode(200);

**What is the purpose of using static import in Rest Assured**

Static import in Rest Assured is used to reduce the amount of code needed to make API requests and validate the responses. By using static imports, you can access the classes and methods contained in the Rest Assured library without needing to use the fully qualified name of the class each time. This allows you to write code more quickly and readably, helping to improve productivity. As an example, if you wanted to send a POST request and validate the response, you could use static imports as follows:

import static io.restassured.RestAssured.\*;

given().

body("{<json body>}").

when().

post("/endpoint").

then().

statusCode(200);

*Top Rest Assured Interview Questions (2024)*

Rest Assured Interview Questions and Answers

1. What is REST Assured?
2. Why is REST-assured being used instead of Postman to automate RESTful services?
3. What is Request Specification in Rest Assured?
4. How to use Request Specification in Rest Assured?
5. Why do we use static import in Rest Assured?
6. What is rest assured method chaining?
7. What is client server architecture?
8. What is REST?
9. What is JSON?
10. How to configure Rest-Assured with Eclipse or any other IDE?
11. Write a code to test REST API using Rest Assured?
12. What are the ways and how to validate the response of REST API in Rest Assured?
13. What is the best way to keep sensitive data out of the log in rest assured?
14. What is jsonPath in Rest Assured?
15. How to log the request and response in case of validation failed in Rest Assured?
16. How to use Path Variable with GET rest endpoint in Rest Assured?
17. How to find all employees ids from 15 to 300 using Rest Assured jsonPath?
18. How to send a POST Request in Rest Assured?
19. What is Serialization and Deserialization in Java?
20. How to deserialize a response JSON as List of POJO in Rest assured?
21. What is the Array slice operator in JsonPath in Rest Assured?
22. How can we get size of JSON array in Rest assured?
23. How to log in case of error in response in Rest assured?

Q: What is REST Assured?  
Ans:

*REST Assured is a Java library RESTful APIs testing. It is extensively used to test web applications that are based on JSON and XML. Also, all methods are completely supported, including GET, PUT, POST, PATCH, and DELETE.*

Q: Why is REST-assured being used instead of Postman to automate RESTful services?  
Ans:

Below are few of the advantages of using REST Assured over Postman:

1. For the REST-assured, we can **customise the reports**. Postman, on the other hand, does not allow us to customise the reports.
2. **Code can be reused** in REST assured, as it is a Java client, whereas code reusability is not possible in Postman.
3. For each collection, we can only submit one data file to the Postman automation runner. However, there is no such restriction for REST-assu1red.

Q: What is Request Specification in Rest Assured?  
Ans:

*RequestSpecification in Rest Assured can be used to group together common request specs and turn them into a single object. This interface has methods for defining the****base URL, base path, headers, and other parameters****. To obtain a reference for RequestSpecification, we must use the given() function of the RestAssured class. We can't make an object out of RequestSpecification because it's an interface. Its implemented class is RequestSpecificationImpl.*

Q: How to use Request Specification in Rest Assured?  
Ans:

RequestSpecification reqSpec = RestAssured.given();

reqSpec.baseUri("http://localhost:8080")

reqSpec.basePath("/employees");

Alternatively, rather than repeatedly invoking RequestSpecification Reference, we may utilise the **builder pattern** as seen below:-

RequestSpecification reqSpec =

RestAssured.given()

.baseUri("http://localhost:8080")

.basePath("/employees");

*Refer*[*Spring Boot Rest Assured Example*](https://www.techgeeknext.com/spring-boot/spring-boot-rest-assured)*to understand how RequestSpecification is used.*

Q: Why do we use static import in Rest Assured?  
Ans:

*Static import is a Java programming language feature that allows members (fields and methods) that have been scoped as public static within their container class to be used in Java code****without mentioning the class in which the field has been defined****.*

package com.techgeeknext.controller;

import org.testng.annotations.Test;

/\*\*

\* this is static import to avoid writing

\* into front of every method call of RestAssured

\*/

import static io.restassured.RestAssured.\*;

public class EmpControllerTest {

@Test

public void testGetEmployees() {

// with static import

given();

// without static import

/\*\*

\* import io.restassured.RestAssured;

\* RestAssured.given();

\*/

}

}

Q: What is rest assured method chaining?  
Ans:

*In object-oriented programming languages, method chaining is a typical syntax for invoking numerous method calls. Each method returns an object, allowing****multiple calls to be chained together****in a single line without the need for variables to hold interim results.*

For example in rest assured all methods chained together with dots.

given()

.baseUri(baseUri)

.queryParam(parameterName, parameterValues)

.accept(contentType).

.when()

.then();

*Refer*[*Spring Boot Rest Assured Example*](https://www.techgeeknext.com/spring-boot/spring-boot-rest-assured)*to understand how static imports are used.*

Q: What is client server architecture?  
Ans:

*The client-server model defines how a server gives resources and services to one or more clients. Web servers, mail servers, and file servers are examples of servers. So, the Client requests something, and the Server fulfils the request.*

Q: What is REST?  
Ans:

*REpresentational State Transfer is the acronym for REpresentational State Transfer. It means that when a RESTful API is invoked, the server will send a representation of the status of the requested resource to the client. The operation you want the server to perform on that resource, expressed as an HTTP method.*

Q: What is JSON?  
Ans:

*JSON (JavaScript Object Notation) is a text-based standard for describing structured data that is based on JavaScript object syntax. It's often used in web applications to send data to server and client.*

Key-Value pairs are used to represent data in JSON Object as below:

{

"id" : 1,

"username" : "TechGeekNextUser",

"role" : "Admin"

}

Q: How to configure Rest-Assured with Eclipse or any other IDE?  
Ans:

Below are the steps to configure Rest-Assured in IDE.

1. Download and setup Java.
2. Setup [TestNG in Eclipse](https://www.techgeeknext.com/testng/install-testng-in-eclipse" \t "_blank) OR [TestNG in IntelliJ](https://www.techgeeknext.com/testng/install-testng-in-intellij" \t "_blank)
3. [Download Rest-Assured jar files](https://github.com/rest-assured/rest-assured/wiki/Downloads) -> Under Current direct downloads -> Download rest-assured-\*-dist.zip
4. In Eclipse, IntelliJ, create new Java Project and add External Jars ( Rest-Assured downloaded jar)

Q: Write a code to test REST API using Rest Assured?  
Ans:

import org.testng.annotations.Test;

import io.restassured.RestAssured;

import io.restassured.http.Method;

import io.restassured.response.Response;

import io.restassured.specification.RequestSpecification;

public class EmployeesTest {

@Test

public void GetAllEmoloyees()

{

// base URL to call

RestAssured.baseURI = "http://localhost:8080/employees/get";

//Provide HTTP method type - GET, and URL to get all employees

//This will give respose

Response employeesResponse = RestAssured.given().request(Method.GET, "/all");

// Print the response in string format

System.out.println(employeesResponse.getBody().asString());

}

}

*Refer*[*Spring Boot Rest Assured Example*](https://www.techgeeknext.com/spring-boot/spring-boot-rest-assured)*to understand it's implementation.*

Q: What are the ways and how to validate the response of REST API in Rest Assured?  
Ans:

Response is an interface available io.restassured.response package.This interface contains many methods, majority of which can be used to extract parts from the received response. Below are some method of response which we can use to validate the received response:

1. Status Code: getStatusCode() method can be used to validate the response. It is an integer values, if it valid response will return 200.
2. Response employeesResponse = RestAssured.given().request(Method.GET, "/all");
3. //validate the resonse using Assert

Assert.assertEquals(200, employeesResponse.getStatusCode());

1. Status Line: getStatusLine() method can be used to validate the response. It contains 3 part i.e. Http Protocol version, Status Code (Integer) and Status Code (String) like HTTP/1.1 200 OK
2. Response employeesResponse = RestAssured.given().request(Method.GET, "/all");
3. //validate response using Assert and checking with response StatusLine

Assert.assertEquals("HTTP/1.1 200 OK",employeesResponse.getStatusLine());

*Refer*[*Spring Boot Rest Assured Example*](https://www.techgeeknext.com/spring-boot/spring-boot-rest-assured)*to understand how to extract and validate the response.*

Q: What is the best way to keep sensitive data out of the log in rest assured?

*It is now available to****blacklist headers****using blacklistHeader method in REST Assured 4.2.0 ensuring that they are not displayed in the request or response log. One or more headers can be blacklisted. A blacklist is used to prevent sensitive data from being included in the log.*

Set<String> headers = new HashSet<String>();

headers.add("X-REGION");

headers.add("content-type");

given().

baseUri("http://localhost:8080").

header("X-REGION", "NAM").

// blacklist headers

config(

config.logConfig(LogConfig.logConfig().blacklistHeaders(headers)))

// blacklist multiple headers

//config(config().logConfig(LogConfig.logConfig().blacklistHeader("Accept","set-cookie"))).

log().all().

when().

get("/employees").

then().

assertThat().

statusCode(200);

Q: What is jsonPath in Rest Assured?  
Ans:

*JsonPath (io.restassured.path.json.JsonPath) is a simple way to get values from an Object document without having to use XPath. When retrieving an object from the document, it follows the Groovy GPath syntax. It can be thought of as a JSON-specific version of XPath. As an example, consider the following Object document.*

{ "company": {

"employee": [

{ "id": 1,

"name": "TechGeekNextUser1",

"role": "Admin"

},

{ "id": 2,

"name": "TechGeekNextUser2",

"role": "User"

},

{ "id": 3,

"name": "TechGeekNextUser3",

"role": "User"

}

]

}

}

Response employeesResponse = RestAssured.given().request(Method.GET, "/all");

JsonPath jsonPathObj = employeesResponse.jsonPath();

//get a list of all employees id:

List<String> employeeIds = jsonPathObj.get("company.employee.id");

//get the first employee name:

String empName = jsonPathObj.get("company.employee[0].name");

*Refer*[*Spring Boot Rest Assured Example*](https://www.techgeeknext.com/spring-boot/spring-boot-rest-assured)*to understand how to use JsonPath to extract the specific object from the response.*

Q: How to log the request and response in case of validation failed in Rest Assured?  
Ans:

If a test validation fails, log().ifValidationFails() logs everything in the request and response.

/\*\*

\* Log the request and response details if validation fails.

\*/

@Test

public void testIfValidationFails() {

given().

baseUri("http://localhost:8080").

header("X-REGION", "NAM").

log().ifValidationFails().

when().

get("/employees").

then().

log().ifValidationFails().

assertThat().

statusCode(200);

}

Q: How to use Path Variable with GET rest endpoint in Rest Assured?  
Ans:

Considering id as path variable in GET Rest end point url - http://localhost:8080/employee/{id}.

Example : http://localhost:8080/employee/33  
Output:

{

"id": 33,

"name": "User-1",

"role": "Admin"

}

Pass 33 value to path variable id in the given program.

@Test

public void testGetEmployeeWithPathParam() {

Response empResponse = given().

baseUri("http://localhost:8080").

contentType(ContentType.JSON).

pathParam("id", "33").

when().

get("/employee/{id}").

then().

log().all().

assertThat().

statusCode(200).

extract().

response();

JsonPath jsonPathObj = empResponse.jsonPath();

Assertions.assertEquals(jsonPathObj.getLong("id"), 33);

Assertions.assertEquals(jsonPathObj.getString("name"), "User-1");

Assertions.assertEquals(jsonPathObj.getString("role"), "Admin");

}

*Refer*[*Spring Boot Rest Assured Example*](https://www.techgeeknext.com/spring-boot/spring-boot-rest-assured)*to understand how to use path variable with Get Rest Endpoint.*

Q: How to find all employees ids from 15 to 300 using Rest Assured jsonPath?  
Ans:

Response employeesResponse = RestAssured.given().request(Method.GET, "/all");

JsonPath jsonPathObj = employeesResponse.jsonPath();

//get all employees id between 15 and 300

List<Map> employees = jsonPathObj.get("company.employee

.findAll { employee -> employee.id >= 15 && employee.id <= 300 }");

Q: How to send a POST Request in Rest Assured?  
Ans:

@Test

public void testPostEmployee() throws JSONException {

JSONObject empParams = new JSONObject();

empParams.put("name", "TechGeekNextUser44");

empParams.put("role", "Supervisor");

given()

.contentType(ContentType.JSON)

.body(empParams.toString())

.log().all()

.when()

.post("http://localhost:8080/employee")

.then()

.assertThat().statusCode(200)

.body("name", equalTo("TechGeekNextUser44"))

.body("role", equalTo("Supervisor"))

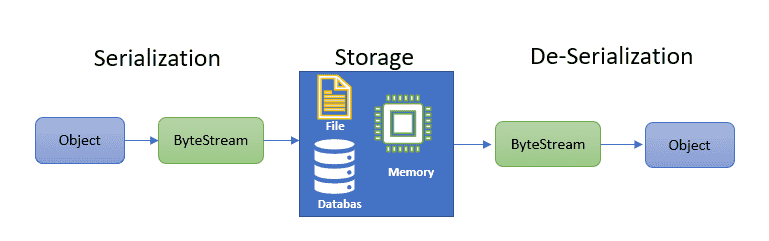
.log().all();

}

*Refer*[*Spring Boot Rest Assured Example*](https://www.techgeeknext.com/spring-boot/spring-boot-rest-assured)*to understand how to implement and test POST Rest Endpoint using Rest Assured.*

Q: What is Serialization and Deserialization in Java?  
Ans:

*Serialization is the process of****converting an object's state into a byte stream****. Deserialization is the process of****recreating the actual Java object in memory using the byte stream****. The object is kept alive through this approach.*

[](https://www.techgeeknext.com/img/serialization.PNG)

Q: How to deserialize a response JSON as List of POJO in Rest assured?  
Ans:

In below ways we can deserialize a response JSON as List of POJO in Rest assured:

1. List<Employee> returnedEmployees = Arrays.asList(response.getBody().as(Employee[].class));
2. In version 3.0.2 (io.restassured):
3. JsonPath jsonPath = RestAssured.given()
4. .when()
5. .get("/employee/get/all")
6. .then()
7. .assertThat()
8. .statusCode(Response.Status.OK.getStatusCode())
9. .assertThat()
10. .extract().body().jsonPath();
11. List<Employee> Employees = jsonPath.getList("", Employee.class);
12. Google's Gson library
13. Gson gson = new Gson();
14. List<Employee> returnedEmployees = gson.fromJson(jsonStr, new TypeToken<List<Employee>>(){}.getType());

Q: What is the Array slice operator in JsonPath in Rest Assured?  
Ans:*The array slice operator is a brilliant way to extract certain objects from Json. What if, in the case of employees, we wanted to get every alternate employee in the Json? We'll need the Array, Slice operator for this. [StartIndex: EndIndex: Steps] is the syntax of the Array Slice operator.*

$..employee[1,4,2]

$..['employee'][1,4,2]

Q: How can we get size of JSON array in Rest assured?  
Ans:

JSON Response:

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

{

"Status": 200,

"ORG": {

"EMPLOYEES": [

{

"id": 1,

"name": XYZ,

"role": "ADMIN"

},

{

"id": 2,

"name": ABC,

"role": "USER"

},

{

"id": 3,

"name": AAA,

"role": "USER"

}

]

}

}

// base URL to call

RestAssured.baseURI = "http://localhost:8080/employees/get";

//Provide HTTP method type - GET, and URL to get all employees

//This will give respose

Response employeesResponse = RestAssured.given().request(Method.GET, "/all");

//use JsonPath from Rest-Assured to get list of employee id

List<String> employees = employeesResponse.jsonPath().getList("ORG.EMPLOYEES.id");

System.out.println(employees.size());

Q: How to log in case of error in response in Rest assured?  
Ans:

In Rest Assured, there is way to logs everything with log().ifError() if there is an error in the response.

/\*\*

\* Log if Error exist

\*/

@Test

public void testLogIfError() {

given().

baseUri("http://localhost:8080").

header("X-REGION", "NAM").

log().all().

when().

get("/employees").

then().

log().ifError().

assertThat().

statusCode(200);

}

API Automation:

Introduction in API and where it is used in project structure.

Real time usage of API's with Examples.

Understanding GET, POST, PUT, DELETE Http CRUD operations of API.

What are the Path, Query Parameters and Headers in Rest API.

Explained Marriot hotel Examples

=>Api is an interface or communication protocol between client and server intended of simplify the building of client side softwares.x

End point: Address where API is hosted on the Server.

HTTP methods which are commonly used to communicate with Rest API’s are

GET, POST, PUT, and DELETE

GET- The GET method is used to extract information from the given server using a given URI. While using GET request, it should only extract data and should have no other effect on the data. No Payload/Body required

How to send input data in GET?

Ans: Using Query Parameters

POST- A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.

How to send input data in POST?

Ans: Using Form Parameters /Body Payload

PUT- Replaces all current representations of the target resource with the uploaded content.

DELETE- Removes all current representations of the target resource given by a URI.

Resources:

Resources represent API/Collection which can be accessed from the Server

Google.com/maps

google.com/search

google.com/images

Path Parameters:

Path parameters are variable parts of a URL path. They are typically used to point to a specific resource within a collection, such as a user identified by ID

<https://www.google.com/Images/1123343>

<https://www.google.com/docs/1123343>

<https://amazon.com/orders/112>

<https://www.google.com/search?q=newyork&oq=newyork&aqs=chrome..69i57j0l7.2501j0j7&sourceid=chrome&ie=UTF-8>

Query Parameters:

Query Parameter is used to sort/filter the resources.

Query Parameters are identified with?””

<https://amazon.com/orders?sort_by=2/20/2020>

Headers/Cookies:

Headers represent the meta-data associated with the API request and response. In layman terms, we were sending Additional details to API to process our request.

Example : Authorization details

End Point Request URL can be constructed as below

Base URL/resource/(Query/Path)Parameters

**RestApi Automation Framework from Scratch**

--> Create Maven Project with Cucumber and RestAssured Dependies

--> Define Project Structure with cucumber framework setup

--> Create a feature file with the testcases you want to automate

--> Implement the smart step Definition files with supported code

--> Build Utils file to define all reusable request and response specifications

--> Build pojo classes for serializing and de-serializing json Payload

--> Implement logging into Framework to request and response details

--> Develop End to End Functinality Test with all Validations and Assersions

--> Define Global Variable and drive all Global variables from the properties file

--> Define enum class with constants to centralize all resouces details

--> Implement Data Driven Mechanism to drive dynamically from feature files

--> Implement Parameterization to run tests with multiple data sets using cucmber example keyword

--> Add More Tests and implement tagging mechanism to run selected tests form Test Runner file

--> Implement Pre and Postconditions for tests with cucmber Hooks

--> Execute Complete framework from Maven Commands

To run on command Line --> Go to project folder --> mvn test

-->mvn test verify -Dcucumber-options="--tags @AddPlace"

--> Implement Maven driven global values into test for dynamic execution

--> Generate Report for Tst Execution results

--> Integrate framework into jenkins CI/CD tool

Top 25 Interview Questions:

1. What all challenges are included under API?

API Documentation

Access to DB

Authorization overhead