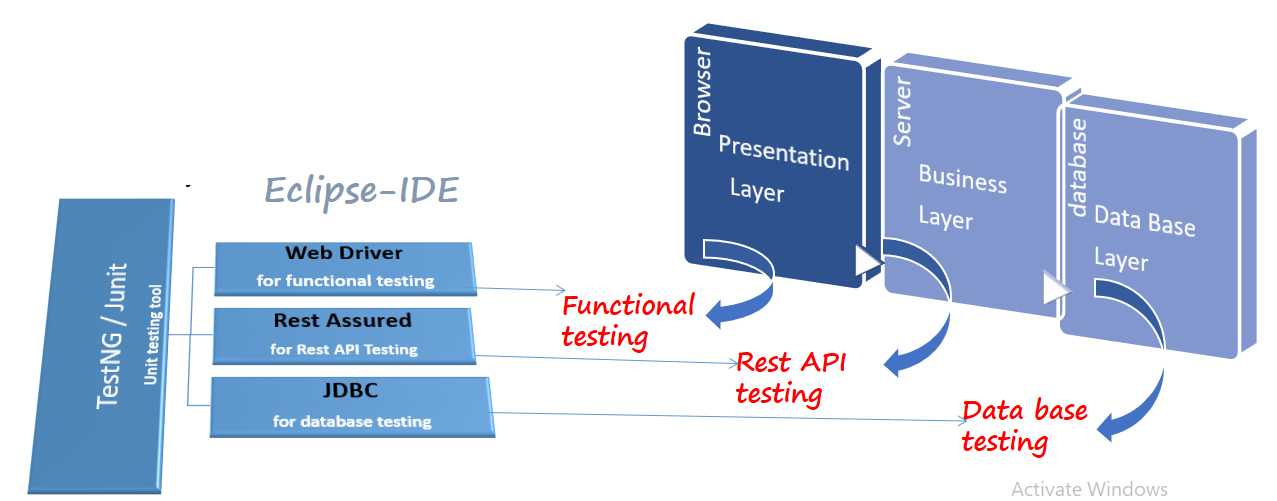
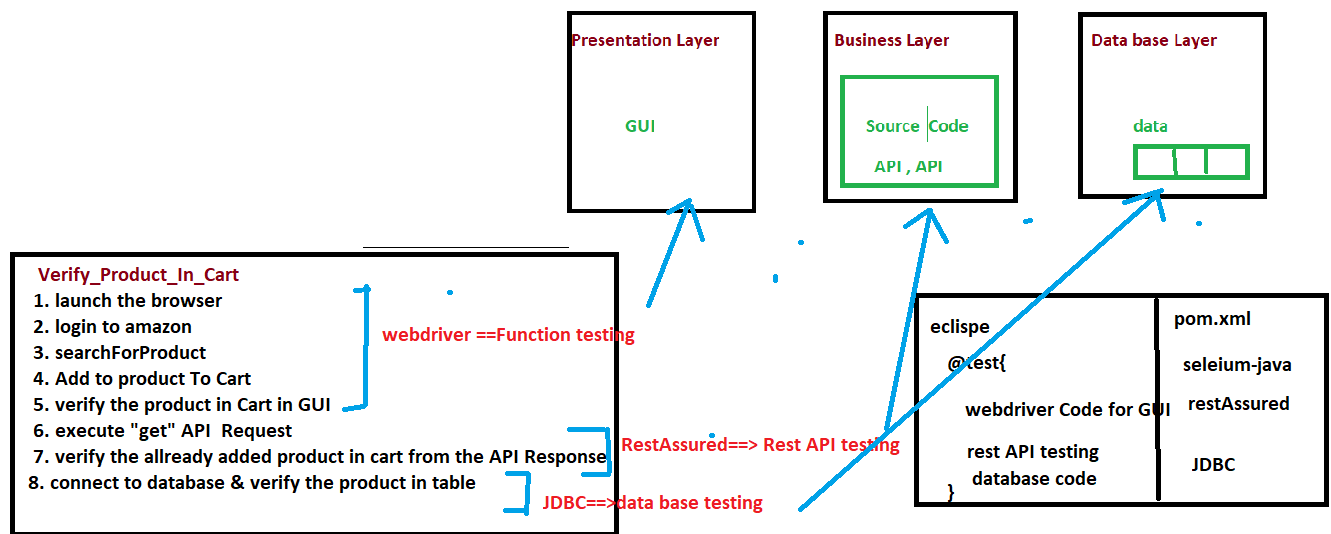
Rest Assured introduction

* *Rest-Assured is a Java based library that is used to test RESTful Web Services. This library behaves like a headless Client to access REST web services*
* Rest Assured simple java libraries for testing of Rest WebServcie API
* *Rest-Assured library also provides ability to validate the Rest-HTTP Responses received from server. For e.g. we can verify the Status code, Status message, Headers and even the Body of the response. This makes Rest-Assured a very flexible library that can be used for testing.*
* It support both XML and JSON

**Why RestAssured is popular?**

****

****

* It Can Integrate seamlessly with existing java based framework like
* Testing/junit/BDD
* Selenium Webdriver
* JDBC
* So that We can Automate E2E business workflow which include all the layers

**Rest assured Advantages**

* Support for all http methods
* Support for BDD
* Use of Hamcrest matches for validation /Assertions
* Provide inbuilt method to create a request header & body
* Provide inbuilt method to validate response header & body
* Handles various authentication like Token AUTH, Basic auth , oauth1.0 , oauth2.0
* Send request over the network using existing protocol like **http**
* Integrate seamlessly with existing java based framework like selenium, testing , jdbc , poi etc
* Used for complete backend automation
* Framework can integrate with CI/CD pipeline
* Open source headless client

**Rest assured Project Setup**

* In order to use Rest assured client ,we should create a maven project & add below dependency

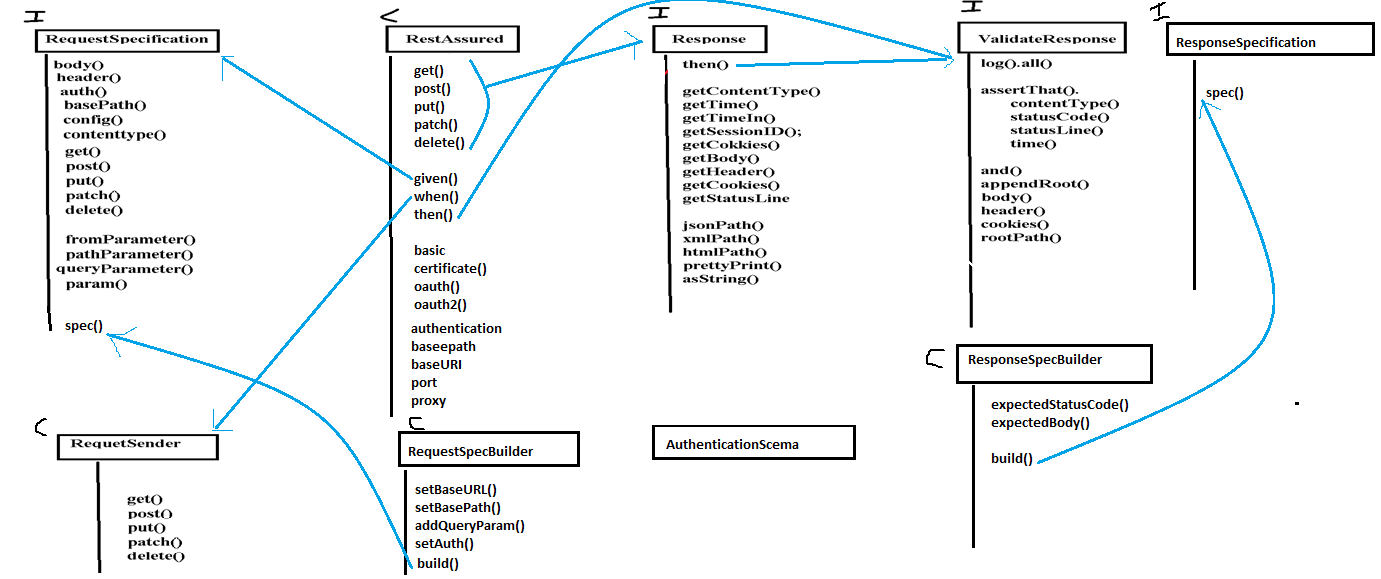
**Step to create maven Project**

* open Eclipse
* create maven project
* Add below dependency in POM.xml

save pom.xml file

|  |  |
| --- | --- |
| **Tool** | **Description** |
| **rest-assured-3.0.2** | **Rest api headless client** |
| **json-simple-1.1.1** | **Used to create Json object** |
| **jackson-mapper-asl-1.9.13** | **Used to create pojo & parser class** |
| **testng-6.8** | **Unit testing tool** |

**Rest assured Class diagram**



Description of Every API

|  |  |
| --- | --- |
| Rest Assured class | Description |
| RestAssured | Main class of restassured , where we can start the Rest API test using  BDD like given(), when() , then() |
| RequestSpecification | Used to set the specification of the request like authentication & parameter etc. |
| RequestBuilder | Used to create common reusable precondition code for the request |
| RequestSpecSender | Used to set precondition for the request |
| Response | Used to receive the response of the request |
| ResponseSpecification | Used to set the specification of the response |
| ResponseSpecBiuilder | Used to create common reusable precondition code for the request |
| ValidateResponse | Used to validate the response |
|  |  |
| JsonObject | used to create JSONObject |
| ObjectMapper | used for serialization |

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.*

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");

## 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();

## 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());

}

}

## 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.

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

//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

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

//validate response using Assert and checking with response StatusLineAssert.assertEquals("HTTP/1.1 200 OK",employeesResponse.getStatusLine());

## 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: 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);

}

What are API Query Parameters?

API Query parameters can be defined as the optional key-value pairs that appear after the question mark in the URL. Basically, they are extensions of the URL that are utilized to help determine specific content or action based on the data being delivered. Query parameters are appended to the end of the URL, using a ‘?’. The question mark sign is used to separate path and query parameters.

If you want to add multiple query parameters, an ‘&’ sign is placed in between them to form what is known as a query string. It can feature various object types with distinct lengths such as arrays, strings, and numbers.

It critical to note that query parameters can play a pivotal role in attribution, however, it is vitally essential to ensure that the attribution strategy is in the cross-platform, and it is performing everything it can.

Query Parameter Examples

1. https://example.com/articles?sort=ASC&page=2

In this URL, there are two query parameters, sort, and page, with ASC and 2 being their values, respectively.

2. http//www.techopedia.com/search.aspx?q=database&ion-all

In the URL above, the bolded values after the ‘?’ are the query parameters, q=database&ion-all (query string).

Query vs. Path Parameters (differences)

The first difference between query and [path parameters](https://rapidapi.com/blog/api-glossary/parameters/path/) is their position in the URL. While the query parameters appear on the right side of the ‘?’ in the URL, path parameters come before the question mark sign.

Secondly, the query parameters are used to sort/filter resources. On the other hand, path parameters are used to identify a specific resource or resources.

You can’t omit values in path parameters since they are part of the URL. On the other hand, query parameters are added at the end of the URL, and thus can allow omission of some values as long as the serializing standards are followed.

Query parameters have unique attributes which help to define resources in a better way. Path parameters, on the other hand, have dynamic resources, which act upon more granular objects of the resource.

When Should You Use Query Parameters?

There are several cases that warrant the use of query parameters. They can be used in filtering criteria, sorting criteria, or to represent the current page number in a paginated collection. Additionally, query parameters can be used in API requests that retrieve data. They are ideal to be passed to various methods to extend the functionality of the API.