# Table of Contents

[**What is Rest Assured?        3**](#h.5y1134nz11ix)

[**How do you add Rest Assured to your project?        3**](#h.unmmtfbnqisb)

[**Explain the main components of a Rest Assured test.        3**](#h.weq0nd8pek12)

[**How do you perform a simple GET request using Rest Assured?        4**](#h.5wiracac04h4)

[**Explain the usage of path parameters in Rest Assured.        4**](#h.4jcftl92r12d)

[**How do you validate JSON responses in Rest Assured?        4**](#h.rkspoema9tgc)

[**Explain the concept of authentication in Rest Assured.        4**](#h.6w4qha2gs8c4)

[**How can you handle cookies in Rest Assured?        5**](#h.4gyoux8583cl)

[**How do you handle query parameters in a Rest Assured request?        5**](#h.6ihmjxb9j94k)

[**Explain the difference between given(), when(), and then() in a Rest Assured test.        5**](#h.wn0mjv2gia9c)

[**How can you extract values from the response and use them in subsequent requests?        6**](#h.shanhqxrwb66)

[**Explain the usage of the log() method in Rest Assured.        6**](#h.bo3e43dytv0n)

[**How do you handle timeouts in Rest Assured requests?        6**](#h.xm0w4bilrmeq)

[**Explain the concept of filters in Rest Assured.        6**](#h.wd929w38hj9v)

[**How can you handle multipart requests using Rest Assured?        7**](#h.uvj62haqurnl)

[**Explain how you handle authentication tokens in Rest Assured for API testing.        7**](#h.ohd27un898xk)

[**How do you handle dynamic values in Rest Assured, such as timestamps in request payloads?        7**](#h.lphdciqayqeh)

[**How can you handle dynamic or changing endpoints in Rest Assured tests?        8**](#h.xz9uvgebvcdp)

[**Explain the usage of the expect() method in Rest Assured.        8**](#h.rl203wdvh5xy)

[**How do you handle response headers in Rest Assured tests?        8**](#h.m91izfpsr6pc)

[**What is the purpose of the Matchers class in Rest Assured?        8**](#h.agvkbcs42bow)

[**How do you perform a POST request with a JSON payload in Rest Assured?        9**](#h.y65ku4i7m3ga)

[**Explain the concept of a Root Path in Rest Assured.        9**](#h.30r03dqj58ge)

[**What is the purpose of the 'basePath()' method in Rest Assured?        9**](#h.nw0hgfawq30e)

[**Explain the concept of Request Specification and Response Specification in Rest Assured.        9**](#h.5ctk0tgmya15)

[**How can you handle authentication using OAuth 2.0 in Rest Assured?        10**](#h.mkraga4mysxx)

[**Explain the purpose of the 'formParam()' method in Rest Assured.        10**](#h.hm33qcj4n09a)

[**How do you handle request and response logging in Rest Assured?        10**](#h.iese96909i84)

[**What is the significance of the 'statusCode()' method in Rest Assured assertions?        10**](#h.zfn8maqrve1d)

[**How can you perform a DELETE request using Rest Assured?        10**](#h.pokv6nryj7ni)

[**Explain the purpose of the 'extract().jsonPath()' method in Rest Assured.        11**](#h.c2714v9ju5a8)

[**Explain the use of the with() method in Rest Assured.        11**](#h.s5pwa625bkj3)

[**How can you validate the response time in Rest Assured?        11**](#h.hn7xlv5wvb8o)

[**Explain the purpose of the config() method in Rest Assured.        11**](#h.ffug6b6s8vtj)

[**Explain the difference between hasItem() and hasItems() in Rest Assured assertions.        12**](#h.qycn9gko7ant)

[**What is the purpose of the 'prettyPrint()' method in Rest Assured?        12**](#h.10cefd8mbmyr)

[**How can you handle SSL certificates in Rest Assured?        12**](#h.avthriqw7bu4)

[**What is the purpose of the 'relaxedHTTPSValidation()' method in Rest Assured?        12**](#h.ednhkh2mramf)

[**How can you set a proxy for your Rest Assured tests?        13**](#h.yogfyth6adoe)

[**What is the purpose of the 'config(JsonConfig.jsonConfig())' method in Rest Assured?        13**](#h.3wcqmfi1pirt)

[**How do you handle XML payloads in Rest Assured requests?        13**](#h.4tnhuie0k5j)

[**Explain the purpose of the 'filter(ResponseLoggingFilter.logResponseTo())' method in Rest Assured.        14**](#h.i80hkqwh8dmj)

[**How can you handle response caching in Rest Assured?        14**](#h.6fpq718qixpy)

[**Explain the purpose of the 'relaxedContentType()' method in Rest Assured.        14**](#h.v8zs88rvatgo)

[**How can you handle dynamic authentication tokens in Rest Assured?        14**](#h.gmj4zrwg022g)

[**Explain the purpose of the spec() method in Rest Assured.        15**](#h.k9u6hh86afni)

[**How do you handle request authentication using a username and password in Rest Assured?        15**](#h.f7mqakmbvhse)

[**How can you set a global timeout for all requests in Rest Assured?        15**](#h.ggt9lu29fny8)

[**How can you perform data-driven testing using Rest Assured?        16**](#h.mxvuzefrzr7h)

[**Explain the purpose of the 'auth().oauth2AuthorizationCodeFlow()' method in Rest Assured.        16**](#h.9xhfnpebovt)

[**How do you handle cookies between multiple requests in Rest Assured?        16**](#h.dmeyripadizw)

[**What is the purpose of the auth().none() method in Rest Assured?        16**](#h.2ckhb1s2voql)

[**How can you handle query parameters with special characters in Rest Assured?        17**](#h.pdkto5c748rn)

[**How do you handle dynamic authentication tokens obtained from a login request in subsequent requests using Rest Assured?        17**](#h.idnhvg3tl3hk)

[**Explain the purpose of the 'config(JsonConfig.jsonConfig().numberReturnType())' method in Rest Assured.        17**](#h.d13h5nniq147)

[**What is the purpose of the 'extract().xmlPath()' method in Rest Assured?        18**](#h.z2rkzle81yil)

[**Explain the purpose of the 'extract().asString()' method in Rest Assured.        18**](#h.1vwh5asd0gaz)

[**How can you handle a scenario where an API response returns paginated data in Rest Assured?        18**](#h.h1k7vmyjc2r)

[**How do you handle dynamic payloads in Rest Assured when the request or response structure varies?        18**](#h.kznf54s153a8)

[**How can you handle assertion failures gracefully in Rest Assured to continue with the execution of subsequent test steps?        19**](#h.jymcuj62uskh)

[**How do you handle dynamic response content where the data is subject to frequent changes, such as timestamps or random values?        19**](#h.sfjhgxq0qzxe)

[**How can you handle parallel execution of Rest Assured tests?        19**](#h.1hiejoxsdq7h)

[**Explain the purpose of the 'extract().statusLine()' method in Rest Assured.        20**](#h.4yr2pvyqrgfz)

[**How can you handle JSON response validation in Rest Assured when the expected JSON structure is complex and contains nested elements?        20**](#h.iw6ia0iul47p)

# What is Rest Assured?

A Java-based library for testing RESTful web services is called Rest Assured. It makes it easier to send HTTP queries and validate the results, which facilitates Java API testing. (Rest Assured supports various HTTP methods, assertions, and JSON/XML parsing capabilities to facilitate comprehensive API testing within Java projects.)

# How do you add Rest Assured to your project?

To incorporate Rest Assured into your project, put the following dependency in the pom.xml file of your Maven project:

io.rest-assured rest-assured 4.4.0 test

# Explain the main components of a Rest Assured test.

A typical Rest Assured test consists of the following components:

Request Specification: Describes the request, including the base URI, authentication, headers, etc.

Request: Represents the HTTP request (GET, POST, PUT, DELETE, etc.).

Response Specification: Describes the expected response, including status codes, headers, and body expectations.

Response: Represents the actual HTTP response received.

A Rest Assured test typically consists of several main components that work together to interact with RESTful APIs, send requests, validate responses, and perform assertions.

# How do you perform a simple GET request using Rest Assured?

Here’s an example of a simple GET request:

Response response = given() .baseUri("https://api.example.com") .when() .get("/resource");

**Explain the usage of path parameters in Rest Assured.**

Variable values can be included in the URL using path parameters. You can specify path parameters in Rest Assured by using the ‘pathParam’ method. As an illustration, consider this:

given() .pathParam("userId", 123) .when() .get("/users/{userId}");

# How do you validate JSON responses in Rest Assured?

To extract and validate values from JSON responses, you can utilize Rest Assured’s ‘JsonPath’ functionality. As an illustration, consider this:

given() .when() .get("/endpoint") .then() .assertThat() .body("key", equalTo("expectedValue"));

# Explain the concept of authentication in Rest Assured.

Rest Assured offers ways to manage several authentication formats, including OAuth and simple authentication. For basic authentication, for instance:

given() .auth() .basic("username", "password") .when() .get("/secured/resource");

given().auth().basic("username", "password")

.when().get("/api/endpoint")

.then().statusCode(200);

# How can you handle cookies in Rest Assured?

The ‘cookies’ method allows you to take cookies out of the response and add them back into next requests. As an illustration, consider this:

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

Map cookies = response.getCookies();

given() .cookies(cookies) .when() .get("/nextEndpoint");

# How do you handle query parameters in a Rest Assured request?

 To add query parameters to a Rest Assured request, use the ‘queryParam’ method. As an illustration, consider this:

given() .queryParam("parameter1", "value1") .queryParam("parameter2", "value2") .when() .get("/endpoint");

# Explain the difference between given(), when(), and then() in a Rest Assured test.

‘given()’ establishes the preconditions or request specifications for Rest Assured,

‘when()’ performs the HTTP request, and

‘then()’ verifies the response. As an illustration:

given() .baseUri("https://api.example.com") .when() .get("/resource") .then() .statusCode(200);

given() sets up the request specification, .auth() specifies the authentication method, and .when() initiates the HTTP request. The .then() section is used for making assertions on the response received.

# How can you extract values from the response and use them in subsequent requests?

You can use the ‘extract()’ method along with ‘path()’ or ‘jsonPath()’ to extract values from the response. Here’s an example:

String userId = given() .when() .get("/users") .then() .extract() .path("users[0].id");

given() .pathParam("userId", userId) .when() .get("/users/{userId}");

# Explain the usage of the log() method in Rest Assured.

The ‘log()’ method is used for logging request and response details. It can be applied to different parts of a request, such as ‘log().all()’, ‘log().body()’, or ‘log().ifValidationFails()’. This is useful for debugging and understanding the details of the HTTP communication.

given() .log().all() .when() .get("/endpoint") .then() .log().body();

# How do you handle timeouts in Rest Assured requests?

You can use the ‘timeout()’ method to set timeouts for requests. For example, setting a timeout of 5000 milliseconds:

given() .timeout(5000) .when() .get("/endpoint");

# Explain the concept of filters in Rest Assured.

Filters in Rest Assured allow you to customize and intercept the request and response. For example, you can use filters to log requests and responses, modify headers, or handle cookies. Here’s a simple example:

given() .filter(new RequestLoggingFilter()) .filter(new ResponseLoggingFilter()) .when() .get("/endpoint");

# How can you handle multipart requests using Rest Assured?

Rest Assured provides the ‘multiPart’ method to handle multipart requests. Here’s an example for sending a file in a multipart request:

given() .multiPart(new File("/path/to/file.txt")) .when() .post("/upload");

# Explain how you handle authentication tokens in Rest Assured for API testing.

You can use the ‘header’ method to include authentication tokens in the request headers. For example, handling a Bearer token:

given() .header("Authorization", "Bearer yourAccessToken") .when() .get("/secured/endpoint");

# How do you handle dynamic values in Rest Assured, such as timestamps in request payloads?

You can use the ‘RestAssured’ class to generate dynamic values. For example, adding a current timestamp to a request payload:

long currentTime = System.currentTimeMillis(); given() .body("{\"timestamp\":" + currentTime + "}") .when() .post("/endpoint");

# How can you handle dynamic or changing endpoints in Rest Assured tests?

You can use the ‘String.format’ method or concatenate strings to dynamically construct endpoints. For example:

String dynamicEndpoint = "/users/" + userId; given() .when() .get(dynamicEndpoint);

# Explain the usage of the expect() method in Rest Assured.

The ‘expect()’ method is used to apply additional expectations on the response, such as status code expectations or time-based expectations. For example:

given() .when() .get("/endpoint") .then() .expect().statusCode(200) .expect().time(lessThan(1000L));

# How do you handle response headers in Rest Assured tests?

You can use the ‘header’ method to validate specific headers in the response. For example, checking the ‘Content-Type’ header:

given() .when() .get("/endpoint") .then() .header("Content-Type", "application/json");

# What is the purpose of the Matchers class in Rest Assured?

 Answer: The Matchers class in Rest Assured provides various static methods for performing different types of assertions on the response. For example, Matchers.equalTo(value) is used to check if a response value is equal to the expected value.

# How do you perform a POST request with a JSON payload in Rest Assured?

Answer: To perform a POST request with a JSON payload, you can use the body() method to include the JSON content. For example:

given() .body("{\"key\": \"value\"}") .when() .post("/endpoint") .then() .statusCode(201);

# Explain the concept of a Root Path in Rest Assured.

Answer: The Root Path in Rest Assured is a base path that is prefixed to all resource paths in a request. It can be set using the rootPath() method. This is useful when you have a common base path for multiple API endpoints.

# What is the purpose of the 'basePath()' method in Rest Assured?

 Answer: The basePath() method is used to set a common base path for all requests in a given test class. It allows you to avoid repeating the base path in each request, making the test class more concise.

# Explain the concept of Request Specification and Response Specification in Rest Assured.

 Answer: Request Specification and Response Specification are reusable sets of conditions that can be applied to multiple requests or responses. They allow you to define common settings, such as base URI, authentication, and assertions, and reuse them across different test cases.

# How can you handle authentication using OAuth 2.0 in Rest Assured?

Answer: Rest Assured provides the auth().oauth2(accessToken) method to handle OAuth 2.0 authentication. You can pass the access token obtained through the OAuth 2.0 flow to this method.

# Explain the purpose of the 'formParam()' method in Rest Assured.

Answer: The formParam() method is used to specify form parameters in a request. It is commonly used in POST requests where the data is sent as form parameters rather than in the request body.

# How do you handle request and response logging in Rest Assured?

Answer: You can use the log().all() method to log all details of the request and response. Additionally, you can use log().ifValidationFails() to log details only if the response validation fails.

# What is the significance of the 'statusCode()' method in Rest Assured assertions?

Answer: The statusCode() method is used to assert the HTTP status code of the response. It ensures that the actual status code matches the expected status code.

# How can you perform a DELETE request using Rest Assured?

Answer: To perform a DELETE request, you can use the delete() method. For example:

given() .when() .delete("/endpoint") .then() .statusCode(204);

# Explain the purpose of the 'extract().jsonPath()' method in Rest Assured.

Answer: The extract().jsonPath() method is used to parse the response body as JSON and allows you to perform assertions on specific JSON elements. It provides a fluent interface for working with JSON responses.

# Explain the use of the with() method in Rest Assured.

Answer: The with() method is used to apply configuration options to a specific request. For example, you can use with().queryParam(“key”, “value”) to add query parameters to a request.

# How can you validate the response time in Rest Assured?

Answer: You can use the time() method to assert the response time. For example:

given() .when() .get("/endpoint") .then() .time(Matchers.lessThan(1000)); // Response time less than 1000 milliseconds

# Explain the purpose of the config() method in Rest Assured.

Answer: The config() method is used to set configuration options for a request. For example, you can use config(RestAssured.config().sslConfig(…)) to configure SSL settings.

# Explain the difference between hasItem() and hasItems() in Rest Assured assertions.

Answer: hasItem() is used to assert the presence of a single item in a collection, while hasItems() is used to assert the presence of multiple items in a collection.

# What is the purpose of the 'prettyPrint()' method in Rest Assured?

Answer: The prettyPrint() method is used to log the request and response in a more human-readable format, making it easier to debug and understand the details.

# How can you handle SSL certificates in Rest Assured?

Answer: To handle SSL certificates, you can use the config() method with SSL configuration. For example:

given() .config(RestAssured.config().sslConfig(new SSLConfig().allowAllHostnames())) .when() .get("/secure-endpoint") .then() .statusCode(200);

# What is the purpose of the 'relaxedHTTPSValidation()' method in Rest Assured?

Answer: The relaxedHTTPSValidation() method is used to disable strict SSL certificate validation, allowing you to make requests to HTTPS endpoints without validating the SSL certificate.

# How can you set a proxy for your Rest Assured tests?

Answer: You can set a proxy using the proxy() method. For example:

given() .proxy("proxy-host", 8080) .when() .get("/endpoint") .then() .statusCode(200);

# What is the purpose of the 'config(JsonConfig.jsonConfig())' method in Rest Assured?

Answer: The config(JsonConfig.jsonConfig()) method is used to configure JSON serialization and deserialization settings. It allows you to customize how JSON data is processed during requests and responses.

# How do you handle XML payloads in Rest Assured requests?

Answer: You can use the contentType() method to set the content type to XML and provide the XML payload using the body() method. For example:

given() .contentType(ContentType.XML) .body("John Doe") .when() .post("/user") .then() .statusCode(201);

# Explain the purpose of the 'filter(ResponseLoggingFilter.logResponseTo())' method in Rest Assured.

Answer: The filter(ResponseLoggingFilter.logResponseTo()) method is used to log the response to a specific output stream. It can be helpful for debugging and analyzing the response details. How do you handle request retries in Rest Assured? Answer: You can use the retry() method to specify the number of times a request should be retried in case of a failure. For example:

given() .retry(3) .when() .get("/endpoint") .then() .statusCode(200);

# How can you handle response caching in Rest Assured?

Answer: You can use the cacheControl() method to handle response caching settings. For example:

given() .cacheControl(CacheControl.maxAge(60, TimeUnit.SECONDS)) .when() .get("/cached-endpoint") .then() .statusCode(200);

# Explain the purpose of the 'relaxedContentType()' method in Rest Assured.

Answer: The relaxedContentType() method is used to perform assertions on the response without considering the exact match of the Content-Type header. It allows for more flexible content type validation.

# How can you handle dynamic authentication tokens in Rest Assured?

Answer: You can use variables to store dynamic authentication tokens and include them in the request. For example:

String authToken = "your\_dynamic\_token";

given() .header("Authorization", "Bearer " + authToken) .when() .get("/secured-endpoint") .then() .statusCode(200);

# Explain the purpose of the spec() method in Rest Assured.

Answer: The spec() method is used to apply a predefined set of request or response specifications to a given request. It helps in reusing common configurations across multiple requests.

# How do you handle request authentication using a username and password in Rest Assured?

Answer: You can use the auth().basic(username, password) method to handle basic authentication. For example:

given() .auth().basic("your\_username", "your\_password") .when() .get("/secured-endpoint") .then() .statusCode(200);

# How can you set a global timeout for all requests in Rest Assured?

Answer: You can use the config() method to set a global timeout for all requests. For example:

RestAssured.config = RestAssured.config().httpClient(HttpClientConfig.httpClientConfig().setParam("http.connection.timeout", 5000));

# How can you perform data-driven testing using Rest Assured?

 Answer: Data-driven testing in Rest Assured can be achieved by iterating over a dataset and using variables in the requests. For example, using a loop to iterate over a list of parameters and making requests for each set of data.

# Explain the purpose of the 'auth().oauth2AuthorizationCodeFlow()' method in Rest Assured.

Answer: The auth().oauth2AuthorizationCodeFlow() method is used for OAuth 2.0 authentication using the authorization code flow. It helps in handling the authentication process with the authorization server.

# How do you handle cookies between multiple requests in Rest Assured?

Answer: You can use the cookies() method to capture cookies from a response and then include them in subsequent requests. For example:

Response response = given() .when() .get("/login") .then() .extract().response();

Map allCookies = response.getCookies();

given() .cookies(allCookies) .when() .get("/secured-endpoint") .then() .statusCode(200);

# What is the purpose of the auth().none() method in Rest Assured?

Answer: The auth().none() method is used to indicate that no authentication is required for a particular request. It’s helpful when dealing with public endpoints that do not require authentication.

# How can you handle query parameters with special characters in Rest Assured?

Answer: You can use the URLEncoder class to encode special characters in query parameters. For example:

String encodedValue = URLEncoder.encode("special@character", StandardCharsets.UTF\_8.toString());

given() .queryParam("param", encodedValue) .when() .get("/endpoint") .then() .statusCode(200);

# How do you handle dynamic authentication tokens obtained from a login request in subsequent requests using Rest Assured?

Answer: You can capture the authentication token from the login response using the extract().path() method and then use it in subsequent requests. For example:

String authToken = given() .body("{\"username\":\"your\_username\", \"password\":\"your\_password\"}") .when() .post("/login") .then() .extract().path("token");

given() .header("Authorization", "Bearer " + authToken) .when() .get("/secured-endpoint") .then() .statusCode(200);

# Explain the purpose of the 'config(JsonConfig.jsonConfig().numberReturnType())' method in Rest Assured.

Answer: The config(JsonConfig.jsonConfig().numberReturnType()) method is used to configure how JSON numeric values are handled. It allows you to specify whether to parse them as integers or floating-point numbers.

# What is the purpose of the 'extract().xmlPath()' method in Rest Assured?

Answer: The extract().xmlPath() method is used to create an XMLPath object from the response. It allows for advanced XML parsing and querying of specific elements in the response.

# Explain the purpose of the 'extract().asString()' method in Rest Assured.

Answer: The extract().asString() method is used to retrieve the response body as a string. It allows you to capture the entire response body for further processing or validation.

# How can you handle a scenario where an API response returns paginated data in Rest Assured?

Answer: You can handle paginated data by making multiple requests, each fetching a page of data using query parameters like page and pageSize. Iterate through the pages until you retrieve all the required data.

# How do you handle dynamic payloads in Rest Assured when the request or response structure varies?

Answer: You can use variables and dynamic data generation to handle dynamic payloads. Additionally, you can leverage tools like Jackson or Gson to dynamically serialize and deserialize Java objects for complex payloads.

# How can you handle assertion failures gracefully in Rest Assured to continue with the execution of subsequent test steps?

Answer: You can use the softAssertions() method from the AssertJ library to create soft assertions, which allow the test to continue even if there are assertion failures. For example:

SoftAssertions softAssert = new SoftAssertions();

softAssert.assertThat(response.getStatusCode()).isEqualTo(200); softAssert.assertThat(response.getBody().jsonPath().getString("name")).isEqualTo("John Doe");

softAssert.assertAll();

# How do you handle dynamic response content where the data is subject to frequent changes, such as timestamps or random values?

 Answer: For dynamic content, you can use relaxed assertions or ignore specific dynamic parts during validation. Regular expressions or custom matchers can also be employed to handle variability in the response.

# How can you handle parallel execution of Rest Assured tests?

Answer: Rest Assured tests can be executed in parallel by using testing frameworks that support parallel execution, such as TestNG or JUnit. Additionally, you can leverage parallel streams or execute tests in separate threads.

# Explain the purpose of the 'extract().statusLine()' method in Rest Assured.

Answer: The extract().statusLine() method is used to retrieve the entire status line from the response. It allows you to capture and validate the complete status line, including the status code and description.

# How can you handle JSON response validation in Rest Assured when the expected JSON structure is complex and contains nested elements?

Answer: You can use nested JSONPath expressions to navigate through the complex structure and perform assertions on specific elements. Additionally, tools like JSON schema validation can be employed for more comprehensive validations.