



# Rest Assured Interview Question Banks

## • What is REST Assured?

- REST Assured is a Java library designed for simplifying and automating the testing of RESTful APIs. It provides a domain-specific language (DSL) for writing readable and expressive tests for API endpoints, making it easier to validate responses, headers, and more.

## • How do you add REST Assured to a project?

- To add REST Assured to a project, you can include its dependency in your project's build configuration file, such as Maven or Gradle. For example, in Maven, you'd add the following dependency.

```
<dependency>
  <groupId>io.rest-assured</groupId>
  <artifactId>rest-assured</artifactId>
  <version>4.4.0</version>
  <scope>test</scope>
</dependency>
```

## • How do you send GET requests using REST Assured?

- To send a GET request using REST Assured, you can use the `get` method from the `RestAssured` class, specifying the API endpoint. For example:

```
import static io.restassured.RestAssured.*;
import io.restassured.response.Response;

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

#### • **How do you validate a response using REST Assured?**

- REST Assured provides various methods for response validation. You can use the `then` on a `Response` object to apply assertions on status codes, headers, response body, etc. For example:

```
response.then().statusCode(200).contentType("application/json");
```

#### • **What is JSONPath in REST Assured?**

- JSONPath is a query language used to traverse and query elements within JSON responses. REST Assured allows you to use JSONPath to extract specific data from JSON responses for validation or further processing.

#### • **How do you extract data using JSONPath with REST Assured?**

- In REST Assured, you can use the `extract().jsonPath()` method on a `Response` object to extract data using JSONPath expressions. For example:

```
String value = response.then().extract().jsonPath().getString("key.subKey");
```

### • **How do you send POST requests using REST Assured?**

- To send a POST request using REST Assured, you can use the `given()` method to set request specifications and then use the `post()` method to send the request. For example:

```
given()
    .contentType("application/json")
    .body(requestBody)
.when()
    .post("https://api.example.com/resource")
.then()
    .statusCode(201);
```

### • **How do you send PUT and DELETE requests with REST Assured?**

Similar to POST requests, you can use the `put()` and `delete()` methods from the `RestAssured` class to send PUT and DELETE requests. For example:

```
// Sending a PUT request
given()
    .contentType("application/json")
    .body(requestBody)
.when()
    .put("https://api.example.com/resource/{id}", resourceId)
.then()
    .statusCode(200);

// Sending a DELETE request
given()
```

```
.when()
    .delete("https://api.example.com/resource/{id}", resourceId)
    .then()
        .statusCode(204);
```

### • ♀ **What are Request Specifications in REST Assured?**

- Request Specifications in REST Assured are pre-defined configurations for requests. They can include headers, query parameters, authentication, and more. Using request specifications helps maintain consistency and reusability across multiple requests.

### • 🐶 **How do you authenticate using REST Assured?**

- REST Assured supports various authentication methods. For example, to perform basic authentication, you can use the `auth()` method along with `basic()` or `preemptive()` authentication. For example:

```
given()
    .auth().basic("username", "password")
    .when()
        .get("https://api.example.com/resource")
    .then()
        .statusCode(200);
```

### • • ♀ **What is BDD in REST Assured?**

- BDD in REST Assured refers to writing tests in a human-readable language that closely resembles natural language. It involves using

methods like `given()`, `when()`, and `then()` to describe the behavior of the API under test, making tests more expressive.

### • • ♢ **How do you perform query parameter validation in REST Assured?**

- You can use the `queryParam()` method within the `given()` section to add query parameters to the request. For validation, you can use the `then()` section to assert the presence or value of the query parameter in the response. For example:

```
javaCopy code
given()
    .queryParam("key", "value")
.when()
    .get("https://api.example.com/resource")
.then()
    .statusCode(200)
    .body("queryParamkey", equalTo("expectedValue"));
```

### • • ♢ **What is Response Specification in REST Assured?**

- Response Specification in REST Assured is a pre-defined configuration for response validation. It allows you to specify assertions and expectations that are applied to the response, ensuring consistent validation across multiple tests.

### • • ♢ **How do you use Response Specification in REST Assured?**

- To use Response Specification, you can define it using the `ResponseSpecBuilder` class and then apply it using the `expect()` method in the `then` section of a request. This way, the specified assertions will be applied to the response. For example:

```

javaCopy code
ResponseSpec responseSpec = new ResponseSpecBuilder()
    .expectStatusCode(200)
    .expectContentType(ContentType.JSON)
    .build();

given()
    .when()
        .get("https://api.example.com/resource")
    .then()
        .spec(responseSpec);

```

#### • • ♪ **How do you handle timeouts in REST Assured?**

- REST Assured allows you to set timeout values using the `given()` section. For example, you can use the `timeouts()` method to define the connection and request timeouts. For instance:

```

javaCopy code
given()
    .timeouts().connectTimeout(5000).readTimeout(5000)
    .when()
        .get("https://api.example.com/resource");

```

#### • • ♪ **What is the "RootPath" feature in REST Assured?**

- The "RootPath" feature in REST Assured allows you to specify a default path that is used as the starting point for extracting data using JSONPath expressions. This can simplify data extraction when dealing with complex JSON structures.

### • • ♪ How do you handle SSL in REST Assured?

- REST Assured handles SSL by default. However, if you need to work with self-signed certificates or specific truststores, you can configure SSL using the `trustStore()` and `relaxedHTTPSValidation()` methods within the `given()` section.

### • • ♪ What is "log()" in REST Assured?

- The `log()` method in REST Assured is used for logging request and response details. It can help diagnose issues and provide insight into the interactions with the API.

### • • ♪ How do you set headers in REST Assured?

- You can use the `header()` method within the `given()` section to set headers for a request. For example:

```
javaCopy code
given()
    .header("Authorization", "Bearer token")
    .header("Content-Type", "application/json")
.when()
    .get("https://api.example.com/resource");
```

### • • ♪ What is the difference between "given()", "when()", and "then()" in REST Assured?

- In REST Assured, `given()` is used to set up preconditions for a request, `when()` is used to execute the request, and `then()` is used to perform assertions on the response. These methods help structure tests in a clear and expressive manner.

## • • ♢ What is BDD in REST Assured?

- BDD in REST Assured refers to writing tests in a human-readable language that closely resembles natural language. It involves using methods like `given()`, `when()`, and `then()` to describe the behavior of the API under test, making tests more expressive.

## • • ♢ How can you effectively manage dynamic values, such as timestamps, when validating responses using REST Assured?

- To handle dynamic values like timestamps during response validation with REST Assured, you can utilize a range of JSONPath functions, such as `"matches()"` or `"contains()"`.

```
given().  
    get("https://api.example.com/resource").  
then().  
    assertThat().  
        body("timestamp", matchesRegex("\\d{4}-\\d{2}-\\d{2}T\\d
```

## • • ♢ How can you validate a numeric response using REST Assured?

To validate a numeric response value using REST Assured, you can employ Hamcrest matchers. For instance, you can utilize `"equalTo()"` to ascertain if a response value aligns with an anticipated numeric value.

```
import static org.hamcrest.Matchers.equalTo;  
  
public void validateNumericResponse() {  
  
    RestAssured.given()  
        .when()  
        .get("https://api.example.com/numeric")  
        .then()
```



```
.assertThat()
    .statusCode(200)
    .body("numericField", equalTo(42)); // Validate that the res
}
```

- • ♪ How does REST Assured handle Unicode characters when testing APIs?

REST Assured automatically manages Unicode characters in both requests and responses. Typically, no additional configuration or handling is necessary.....

```
public static void main(String[] args) {
    // Setting up base URI
    RestAssured.baseURI = "https://api.example.com";
    // Sending GET request
    Response response = RestAssured.get("/endpoint");
    // Retrieving response body
    String responseBody = response.getBody().asString();
    // Printing response body
    System.out.println("Response Body: " + responseBody);
}
```

- • ♪ Describe how to perform `HEAD` requests using REST Assured?

```
public static void main(String[] args) {
    // Setting up base URI
    RestAssured.baseURI = "<https://api.example.com>";

    // Sending HEAD request
    Response response = RestAssured.head("/resource");
}
```

```
// Retrieving status code
int statusCode = response.getStatusCode();

// Printing status code
System.out.println("Status Code: " + statusCode);
}
```

- • Describe how to extract response time information from a response using REST Assured?
  - You can use the `time()` method on a Response object to extract the response time in milliseconds. For example:

```
// Send a request and capture the response
Response response = RestAssured.get("https://api.example.com")
// Extract the response time in milliseconds
long responseTime = response.time();
```