# 100 REST ASSURED INTERVIEW QUESTIONS AND ANSWERS

## General API Testing Questions

**Q: What is API testing?**

A: API testing involves testing application programming interfaces directly and as part of integration testing to determine if they meet expectations for functionality, reliability, performance, and security.

---

**Q: Why is API testing important?**

A: API testing is important because it ensures that the API performs as expected, is secure, and can handle the expected load and stress. It is also essential for ensuring the reliability of the communication between different software systems.

---

**Q: What is the difference between API testing and unit testing?**

A: Unit testing focuses on individual components of the software, while API testing focuses on the entire API, ensuring that endpoints function correctly and meet the requirements.

---

**Q: What are the common types of API testing?**

A: Common types include functional testing, load testing, runtime error detection, security testing, UI testing, penetration testing, and fuzz testing.

---

**Q: What tools are commonly used for API testing?**

A: Common tools include Postman, SoapUI, JMeter, RestAssured, Swagger, and Katalon Studio.

---

## REST Assured Specific Questions

**Q: What is REST Assured?**

A: REST Assured is a Java library used for testing and validating RESTful APIs. It simplifies the process of testing APIs by providing a domain-specific language (DSL) for writing tests.

Trailhead Titans

**Q: How do you set up REST Assured in a project?**

A: Add the REST Assured dependency to your Maven or Gradle project. For Maven:

```
<dependency>
    <groupId>io.rest-assured</groupId>
    <artifactId>rest-assured</artifactId>
    <version>4.3.3</version>
    <scope>test</scope>
</dependency>
```

---

**Q: How do you perform a GET request using REST Assured?**

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

Response response = RestAssured.get("https://api.example.com/resource");
System.out.println(response.getStatusCode());
```

---

**Q: How do you validate the status code of a response using REST Assured?**

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

---

**Q: How do you send a POST request with a JSON body using REST Assured?**

```
given().
   contentType("application/json").
   body("{ \"key\": \"value\" }").
when().
   post("https://api.example.com/resource").
then().
   statusCode(201);
```

**HTTP Methods and Status Codes**

**Q: What are the main HTTP methods used in RESTful APIs?**

A: The main methods are GET, POST, PUT, DELETE, PATCH, OPTIONS, and HEAD.

---

**Q: What does the status code 200 mean?**

A: 200 OK means the request has succeeded.

---

**Q: What does the status code 201 mean?**

A: 201 Created means the request has been fulfilled and resulted in a new resource being created.

---

**Q: What does the status code 400 mean?**

A: 400 Bad Request means the server could not understand the request due to invalid syntax.

---

**Q: What does the status code 401 mean?**

A: 401 Unauthorized means the client must authenticate itself to get the requested response.

---

**Q: What does the status code 404 mean?**

A: 404 Not Found means the server cannot find the requested resource.

---

**Q: What does the status code 500 mean?**

A: 500 Internal Server Error means the server encountered an unexpected condition that prevented it from fulfilling the request.

Trailhead Titans

**Q: How do you validate the response content type using REST Assured?**

```
given().
when().
    get("https://api.example.com/resource").
then().
    assertThat().
    contentType("application/json");
```

---

**Q: How do you validate JSON response data using REST Assured?**

```
given().
when().
    get("https://api.example.com/resource").
then().
    body("key", equalTo("value"));
```

---

**Q: How do you log request and response details in REST Assured?**

```
given().
    log().all().
when().
    get("https://api.example.com/resource").
then().
    log().all();
```

---

**Authentication and Authorization**

**Q: How do you handle basic authentication in REST Assured?**

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

Trained Titans

**Q: How do you handle OAuth2 authentication in REST Assured?**

```
given().
    auth().
    oauth2("your_access_token").
when().
    get("https://api.example.com/secure-resource").
then().
    statusCode(200);
```

---

**Q: What is the difference between authentication and authorization?**

A: Authentication verifies the identity of a user, while authorization determines what an authenticated user is allowed to do.

---

**Request and Response Specification**

**Q: What is a request specification in REST Assured?**

A: A request specification is a reusable set of request configurations like base URI, headers, and authentication that can be applied to multiple requests.

---

**Q: How do you create and use a request specification in REST Assured?**

```
RequestSpecification requestSpec = new RequestSpecBuilder().
    setBaseUri("https://api.example.com").
    build();

given().
    spec(requestSpec).
when().
    get("/resource").
then().
    statusCode(200);
```

**Q: What is a response specification in REST Assured?**

A: A response specification is a reusable set of response verifications like status code, headers, and body content that can be applied to multiple responses.

Thread Titans

**Q: How do you create and use a response specification in REST Assured?**

```
ResponseSpecification responseSpec = new ResponseSpecBuilder().
    expectStatusCode(200).
    build();

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

---

**Headers and Parameters**

**Q: How do you add headers to a request in REST Assured?**

```
given().
    header("Content-Type", "application/json").
when().
    get("https://api.example.com/resource").
then().
    statusCode(200);
```

---

**Q: How do you add query parameters to a request in REST Assured?**

```
given().
    queryParam("param1", "value1").
when().
    get("https://api.example.com/resource").
then().
    statusCode(200);
```

Tra oad Titans

**Q: How do you send cookies with a request in REST Assured?**

```
given().
    cookie("session_id", "abc123").
when().
    get("https://api.example.com/resource").
then().
    statusCode(200);
```

---

**Q: How do you validate response headers using REST Assured?**

```
given().
when().
    get("https://api.example.com/resource").
then().
    assertThat().
    header("Content-Type", "application/json");
```

---

**Data Handling**

**Q: How do you validate a JSON array in the response using REST Assured?**

```
given().
when().
    get("https://api.example.com/resource").
then().
    body("data", hasSize(5));
```

---

**Q: How do you extract a value from the JSON response using REST Assured?**

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

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

---

**Q: How do you handle dynamic data in API tests using REST Assured?**

A: You can use placeholders in your request body and replace them with dynamically generated data during test execution.

Skilled Titans

**Q: How do you validate that a key is present in the JSON response using REST Assured?**

```
given().
when().
    get("https://api.example.com/resource").
then().
    body("$", hasKey("key"));
```

---

**Q: How do you handle nested JSON responses in REST Assured?**

```
given().
when().
    get("https://api.example.com/resource").
then().
    body("parent.child", equalTo("value"));
```

---

**Q: How do you validate the response time of an API request using REST Assured?**

```
given().
when().
    get("https://api.example.com/resource").
then().
    time(lessThan(2000L));
```

---

**Q: How do you parse XML responses using REST Assured?**

```
given().
when().
    get("https://api.example.com/resource").
then().
    body("response.element", equalTo("value"));
```

Thread Titans

**Q: How do you validate that a list in a JSON response contains specific values using REST Assured?**

```
given().
when().
    get("https://api.example.com/resource").
then().
    body("data", hasItems("value1", "value2"));
```

---

**Advanced REST Assured Features**

**Q: How do you configure REST Assured to use relaxed HTTPS validation?**

A: RestAssured.useRelaxedHTTPSValidation();

---

**Q: How do you reuse request and response specifications across multiple tests in REST Assured?**

A: Define the specifications in a setup method annotated with @BeforeClass (JUnit) or @BeforeAll (TestNG).

---

**Q: How do you perform file upload in REST Assured?**

```
given().
    multiPart(new File("/path/to/file")).
when().
    post("https://api.example.com/upload").
then().
    statusCode(200);
```

---

**Q: How do you perform multipart form data requests in REST Assured?**

```
given().
    multiPart("formField", "value").
    multiPart("fileField", new File("/path/to/file")).
when().
    post("https://api.example.com/resource").
then().
    statusCode(200);
```

**Q: How do you handle custom deserialization in REST Assured?**

A: Implement a custom deserializer and register it with the ObjectMapper in REST Assured.

Squad Titans

---

**Q: How do you extract cookies from a response in REST Assured?**

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

Map<String, String> cookies = response.getCookies();
```

---

**Q: How do you handle redirects in REST Assured?**

A: RestAssured.followRedirects = true;

---

**Q: How do you capture request and response logs to a file in REST Assured?**

A: Configure a logger using RestAssured.config().logConfig() and specify the log file.

---

**Error Handling**

**Q: How do you handle exceptions in REST Assured?**

A: Use try-catch blocks to catch exceptions and handle them appropriately in your test cases.

---

**Q: What strategies can you use to retry failed requests in REST Assured?**

A: Implement a retry mechanism using loops or third-party libraries like resilience4j.

---

**Q: How do you handle rate limiting in API tests using REST Assured?**

A: Implement logic to check response headers for rate limit information and pause execution if limits are reached.

Trailhead Titans

**Performance Testing**

**Q: How do you perform load testing using REST Assured?**

A: REST Assured is not designed for load testing. Use tools like JMeter or Gatling for load testing APIs.

---

**Q: How do you measure response times in REST Assured?**

```
long responseTime = given().
when().
    get("https://api.example.com/resource").
then().
    extract().
    time();
```

---

**Q: How do you verify performance metrics like throughput and latency in API tests?**

A: Use dedicated performance testing tools like JMeter or Gatling to measure throughput and latency.

---

**API SECURITY TESTING**

**Q: What is API security testing?**

A: API security testing involves validating the security mechanisms of an API, including authentication, authorization, and data protection measures.

---

**Q: How do you perform SQL injection testing on an API using REST Assured?**

A: Send payloads that include SQL injection strings and validate the API's response to ensure it is properly sanitized.

---

**Q: How do you perform cross-site scripting (XSS) testing on an API using REST Assured?**

A: Send payloads that include XSS attack strings and validate the API's response to ensure it is properly sanitized.

**Q: How do you test for broken authentication vulnerabilities in an API?**

A: Test API endpoints with and without proper authentication credentials to ensure that unauthorized access is prevented.

---

Trailhead Titans

**Q: How do you validate JWT tokens in REST Assured?**

```
given().
    auth().
    oauth2("your_jwt_token").
when().
    get("https://api.example.com/secure-resource").
then().
    statusCode(200);
```

---

**Best Practices**

**Q: What are some best practices for writing API tests with REST Assured?**

A:

- Use request and response specifications to reduce code duplication.
- Validate both positive and negative scenarios.
- Use data-driven testing to cover various input combinations.
- Keep tests independent and isolated.
- Maintain clear and concise assertions.

**Q: How do you handle versioning in API tests?**

A: Include version information in the request URL or headers and maintain separate test cases for different API versions.

---

**Q: How do you ensure your API tests are maintainable?**

A: Use page object pattern, reusable methods, and keep tests modular and independent.

---

**Q: How do you integrate REST Assured tests with CI/CD pipelines?**

A: Use build tools like Maven or Gradle to run tests in CI/CD pipelines, and configure your CI/CD tool (e.g., Jenkins) to execute the tests as part of the build process.

---

**Data-Driven Testing**

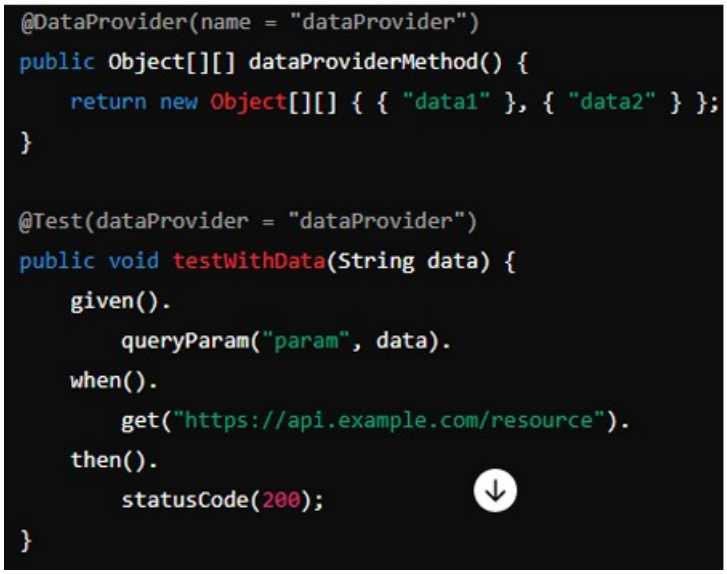
**Q: What is data-driven testing?**

A: Data-driven testing is a methodology where test data is separated from test logic, allowing you to run the same test with different input data sets.

---

Trailhead Titans

**Q: How do you implement data-driven testing in REST Assured using TestNG?**

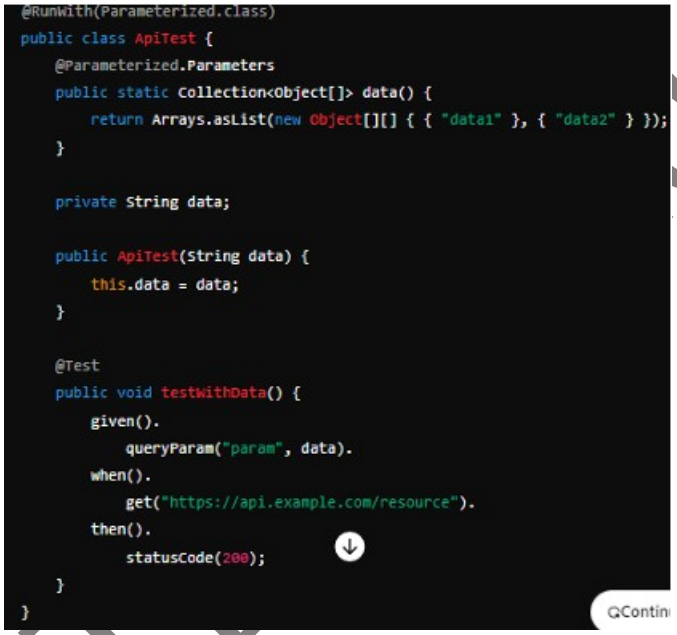


```
@DataProvider(name = "dataProvider")
public Object[][] dataProviderMethod() {
    return new Object[][] { { "data1" }, { "data2" } };
}

@Test(dataProvider = "dataProvider")
public void testWithData(String data) {
    given().
        queryParam("param", data).
    when().
        get("https://api.example.com/resource").
    then().
        statusCode(200);
}
```

---

**Q: How do you implement data-driven testing in REST Assured using JUnit?**



```
@RunWith(Parameterized.class)
public class ApiTest {
    @Parameterized.Parameters
    public static Collection<Object[]> data() {
        return Arrays.asList(new Object[][] { { "data1" }, { "data2" } });
    }

    private String data;

    public ApiTest(String data) {
        this.data = data;
    }

    @Test
    public void testWithData() {
        given().
            queryParam("param", data).
        when().
            get("https://api.example.com/resource").
        then().
            statusCode(200);
    }
}
```

**Mocking and Stubbing**

**Q: What is the purpose of mocking in API testing?**

A: Mocking allows you to simulate the behavior of a real API, enabling you to test how your system interacts with the API without needing the actual API to be available.

---

**Q: How do you mock an API endpoint using REST Assured?**

A: REST Assured does not support mocking directly. Use tools like WireMock to create mock servers and simulate API responses.

---

**Q: How do you integrate REST Assured with WireMock?**

```
import static com.github.tomakehurst.wiremock.client.WireMock.*;

WireMockServer wireMockServer =
```

---

**Q: How do you set up and start a WireMock server for testing with REST Assured?**

```
import com.github.tomakehurst.wiremock.WireMockServer;
import static com.github.tomakehurst.wiremock.core.WireMockConfiguration.wireMockConfig;

WireMockServer wireMockServer = new WireMockServer(wireMockConfig().port(8080));
wireMockServer.start();
```

---

**Q: How do you configure a mock response in WireMock?**

```
import static com.github.tomakehurst.wiremock.client.WireMock.*;

wireMockServer.stubFor(get(urlEqualTo("/resource"))
    .willReturn(aResponse()
        .withStatus(200)
        .withHeader("Content-Type", "application/json")
        .withBody("{\"key\": \"value\"}")));
```

**Q: How do you use REST Assured to send requests to the WireMock server?**

```
given().
    baseUri("http://localhost:8080").
when().
    get("/resource").
then().
    statusCode(200).
    body("key", equalTo("value"));
```

---

**Q: How do you stop the WireMock server after tests are completed?**

A: wireMockServer.stop()

---

**Q: What are the benefits of using mocking in API testing?**

A: Mocking allows you to:

- Isolate the system under test from external dependencies.
- Test scenarios that are hard to reproduce in a real environment.
- Speed up testing by avoiding network delays.
- Control the responses from the dependencies.

---

**Data Validation**

**Q: How do you validate XML schema in REST Assured?**

```
import static io.restassured.module.jsv.XmlSchemaValidator.*;

given().
when().
    get("https://api.example.com/resource").
then().
    body(matchesXsdInClasspath("schema.xsd"));
```

**Q: How do you validate JSON schema in REST Assured?**

```
import static io.restassured.module.jsv.JsonSchemaValidator.*;

given().
when().
    get("https://api.example.com/resource").
then().
    body(matchesJsonSchemaInClasspath("schema.json"));
```

---

**Q: How do you handle optional fields in JSON validation using REST Assured?**

**A: Use conditional assertions or hasKey to check for the presence of optional fields.**

```
given().
when().
    get("https://api.example.com/resource").
then().
    body("$", hasKey("optionalKey"));
```

---

Trainer Kartans

**Versioning and Environment Management**

**Q: How do you handle API versioning in REST Assured tests?**

A: Parameterize your tests to include the API version in the URL or headers.

```
String apiVersion = "v1";
given().
    header("API-Version", apiVersion).
when().
    get("https://api.example.com/" + apiVersion + "/resource").
then().
    statusCode(200);
```

**Q: How do you manage different environments (dev, staging, prod) in REST Assured?**

A: Use environment-specific configuration files or system properties to set the base URI and other environment-specific settings.

**Example:**

String baseUri = System.getProperty("baseUri", "https://api.dev.example.com");

RestAssured.baseURI = baseUri;

---

**Error Handling and Resilience**

**Q: How do you test for error responses in REST Assured?**

A: Send requests with invalid data or to non-existent endpoints and verify the error responses.

```
given().
when().
    get("https://api.example.com/non-existent").
then().
    statusCode(404);
```

---

**Q: How do you handle intermittent failures in API tests?**

A: Implement retry logic or use a library like resilience4j to retry failed requests.

```
int maxRetries = 3;
for (int i = 0; i < maxRetries; i++) {
    Response response = given().get("https://api.example.com/resource");
    if (response.getStatusCode() == 200) {
        break;
    }
}
```

---

Tra ead Titans

**Q: How do you simulate network failures in API tests?**

A: Use tools like Chaos Monkey or network simulation tools to simulate network issues and test the API's resilience.

**Miscellaneous**

**Q: How do you test APIs that require multipart file uploads using REST Assured?**

```
given().
    multiPart("file", new File("path/to/file")).
when().
    post("https://api.example.com/upload").
then().
    statusCode(200);
```

---

**Q: How do you handle different content types in REST Assured?**

**A: Set the Content-Type and Accept headers as needed for the request.**

```
given().
    contentType("application/json").
    accept("application/json").
when().
    post("https://api.example.com/resource").
then().
    statusCode(200);
```

---

**Q: How do you handle pagination in API testing using REST Assured?**

**A: Loop through the pages and validate the responses.**

**Q: How do you validate deeply nested JSON structures in REST Assured?**

```
given().
when().
    get("https://api.example.com/resource").
then().
    body("parent.child.grandchild", equalTo("value"));
```

Trailblazed Titans

**Q: How do you validate the response of a POST request with a payload in REST Assured?**

```
given().
    contentType("application/json").
    body("{ \"key\": \"value\" }").
when().
    post("https://api.example.com/resource").
then().
    statusCode(201).
    body("key", equalTo("value"));
```

---

**Q: How do you use REST Assured to validate the status code and headers of a response?**

```
given().
when().
    get("https://api.example.com/resource").
then().
    statusCode(200).
    header("Content-Type", equalTo("application/json"));
```

---

**Q: How do you send a PUT request with REST Assured?**

```
given().
    contentType("application/json").
    body("{ \"key\": \"new_value\" }").
when().
    put("https://api.example.com/resource/1").
then().
    statusCode(200);
```

**Q: How do you validate a DELETE request response using REST Assured?**

```
given().
when().
    delete("https://api.example.com/resource/1").
then().
    statusCode(204);
```

**Q: How do you handle path parameters in REST Assured?**

```
given().
    pathParam("id", 1).
when().
    get("https://api.example.com/resource/{id}").
then().
    statusCode(200).
    body("id", equalTo(1));
```

---

**Q: How do you validate complex JSON responses that contain lists and objects in REST Assured?**

```
given().
when().
    get("https://api.example.com/resource").
then().
    body("items.size()", greaterThan(0)).
    body("items[0].name", equalTo("Item 1"));
```

---

**Q: How do you handle basic authentication in REST Assured?**

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

**Q: How do you set a custom request timeout in REST Assured?**

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

Trained Titans

**Q: How do you validate multiple assertions in a single REST Assured request?**

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

body("key1", equalTo("value1")). body("key2", equalTo("value2"));

---

**API Contract Testing**

**What is API contract testing?**

A: API contract testing ensures that an API adheres to a defined contract (e.g., OpenAPI/Swagger specification) in terms of request and response formats, parameters, and data types.

---

**Q: Why is API Contract Testing important?**

A: API Contract Testing is important because it:

- Ensures that changes to the API do not break existing functionality.
- Provides a clear agreement between API providers and consumers.
- Helps catch integration issues early in the development cycle.
- Improves the reliability and stability of the API.

---

**Q: What are some tools commonly used for API Contract Testing?**

A: Some commonly used tools for API Contract Testing include:

- Swagger/OpenAPI: Provides a specification for documenting APIs.
- Postman: Offers features for defining and testing API contracts.
- Pact: A tool specifically designed for contract testing between services.
- Dredd: Validates API documentation against the actual API.

---

**Q: What is Pact, and how does it work in API Contract Testing?**

A: Pact is a contract testing tool that ensures compatibility between service providers and consumers by creating "pacts" or agreements. It works by having the consumer tests generate contract files (pacts) that describe the expected interactions. These pacts are then used to verify the provider to ensure it meets the consumer's expectations.

**Q: What are the key components of an API contract?**

A: Key components of an API contract include:

- Endpoint URLs: The paths to access resources.
- HTTP Methods: The operations allowed (GET, POST, PUT, DELETE, etc.).
- Request Parameters: Query parameters, headers, and body content required for the request.
- Response Format: The structure of the response, including status codes, headers, and body content.

Trailhead Titans

- Data Types: The data types of request and response fields.
- Constraints: Validation rules such as required fields, formats, and value ranges.

---

**Q: What are some challenges associated with API Contract Testing?**

A: Challenges in API Contract Testing include:

- **Versioning:** Managing different versions of the API and their respective contracts.
- **Tooling Compatibility:** Ensuring tools used for contract testing integrate well with the development workflow.
- **Dynamic Data:** Handling dynamic or non-deterministic data in requests and responses.
- **Maintenance:** Keeping the contract definitions up-to-date as the API evolves.
- **Consumer-Driven Contracts:** Ensuring that the contract reflects the needs of all consumers, not just one.

---

**Q: How do you ensure backward compatibility in API Contract Testing?**

A: To ensure backward compatibility:

- Version the API: Clearly version the API and maintain contracts for each version.
- Deprecate Gradually: Mark old features as deprecated before removal.
- Run Regression Tests: Continuously run tests for both old and new versions of the API.
- Consumer Feedback: Regularly collect feedback from API consumers to understand the impact of changes.

---

**Q: What is the difference between schema validation and API Contract Testing?**

A:

Schema Validation: Focuses on ensuring that the data structures (such as JSON or XML) conform to predefined schemas. It validates the format and data types of the API responses.

- API Contract Testing: Encompasses schema validation but goes beyond to include validating the entire interaction between the API provider and consumer. It ensures the API meets the specified contract in terms of request/response structure, status codes, headers, and data integrity.

Trailhead Titans