REST Assured explanation

give me the deep and detailed explaination of Rest assured in java so that i can able to understand it well and do not have any type of doubt in this.

so give me the deep and detailed explaination

Alright Piyush 👍
Since you like **deep, doubt-free explanations**, I'll explain **REST Assured in Java** from **absolute basics → internals → real-world usage**, step by step, in **very simple language**, just like a mentor sitting beside you.

🔷 What is REST Assured in Java?

REST Assured is a **Java library** used to **test RESTful APIs**.

In simple words:

> **REST Assured allows you to send HTTP requests (GET, POST, PUT, DELETE, etc.) and verify the responses using Java code.**

It is mainly used for:

- **API Automation Testing**
- **Backend Testing**
- **Integration Testing**

It is widely used with **JUnit/TestNG** and fits perfectly with **Spring Boot APIs**, which is great for your backend goals.

🔷 Why Do We Need REST Assured?

Before REST Assured, API testing in Java was **painful** 😵:

- Using `HttpURLConnection`
- Writing lots of boilerplate code
- Manual JSON parsing

- Hard-to-read assertions

REST Assured solves all of this.

Without REST Assured (painful)

```java
HttpURLConnection conn = ...
BufferedReader reader = ...
parse JSON manually
```

With REST Assured (clean & readable)

```java
given()
   .when()
   .get("/users")
   .then()
   .statusCode(200);
```

👉 **Readable like English**

🔷 What Exactly Is REST Assured?

REST Assured is:

- A **Domain Specific Language (DSL)** for API testing
- Built **on top of Java**
- Uses **Hamcrest matchers**
- Supports **JSON & XML**
- Integrates with **JUnit/TestNG**

🔷 What Is REST (Quick Refresher)

REST = **Representational State Transfer**

REST APIs work using:

- **HTTP Methods**

- **URLs**
- **Headers**
- **Body**
- **Status Codes**

Example:

```
bash

GET /users/1
POST /users
PUT /users/1
DELETE /users/1
```

REST Assured helps you **test all these parts**.

🔷 REST Assured Architecture (VERY IMPORTANT)

REST Assured internally follows this **Given–When–Then** pattern:

```
less

Given → When → Then
```

1️⃣ Given (Request Preparation)

- Headers
- Query params
- Path params
- Request body
- Authentication

2️⃣ When (Action)

- HTTP method (GET, POST, PUT, DELETE)

3️⃣ Then (Validation)

- Status code
- Response body
- Headers
- Time

- Schema

🔷 Setting Up REST Assured

Maven Dependency

```xml
<dependency>
    <groupId>io.rest-assured</groupId>
    <artifactId>rest-assured</artifactId>
    <version>5.4.0</version>
    <scope>test</scope>
</dependency>
```

Usually used with:

- JUnit
  or
- TestNG

🔷 First REST Assured Test (Hello World)

```java
import static io.restassured.RestAssured.*;

public class FirstTest {

    @Test
    public void testGetUsers() {
        given()
        .when()
            .get("https://reqres.in/api/users/2")
        .then()
            .statusCode(200);
    }
}
```

What happens internally?

1. REST Assured sends a **GET request**
2. Server responds with **JSON**
3. REST Assured checks:
   - Status code = 200
4. Test passes ✔️

🔷 Understanding `given()`

```java
given()
    .header("Content-Type", "application/json")
    .queryParam("page", 2)
```

Used for:

- Headers
- Query parameters
- Path parameters
- Request body
- Authentication

🔷 Understanding `when()`

```java
when()
    .get("/users");
```

This is where the **actual HTTP request is fired**.

Supported methods:

- `get()`
- `post()`
- `put()`
- `patch()`
- `delete()`

🔷 Understanding `then()`

```java
then()
   .statusCode(200)
   .body("data.id", equalTo(2));
```

Used for **assertions**:

- Status code
- Response body
- Headers
- Cookies
- Response time

🔷 Validating JSON Response (VERY IMPORTANT)

Example JSON

```json
{
  "data": {
    "id": 2,
    "email": "janet@example.com"
  }
}
```

Validation

```java
.then()
   .body("data.id", equalTo(2))
   .body("data.email", equalTo("janet@example.com"));
```

👉 REST Assured uses **GPath expressions** to navigate JSON.

🔷 Sending POST Request (With Body)

```java
given()
   .header("Content-Type", "application/json")
   .body("""
      {
         "name": "Piyush",
         "job": "Backend Developer"
      }
   """)
.when()
   .post("https://reqres.in/api/users")
.then()
   .statusCode(201);
```

Internally:

- Body → converted to JSON
- Sent as HTTP POST
- Response validated

🔷 Path Parameters

```java
given()
   .pathParam("id", 2)
.when()
   .get("/users/{id}")
.then()
   .statusCode(200);
```

🔷 Query Parameters

```java
```

```
given()
    .queryParam("page", 2)
.when()
    .get("/users")
.then()
    .statusCode(200);
```

🔷 Authentication in REST Assured

Basic Auth

java

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

Bearer Token (JWT)

java

```
given()
    .header("Authorization", "Bearer " + token)
```

Very useful for **Spring Security + JWT**, which you're already learning.

🔷 Extracting Response

java

```
Response response =
given()
.when()
    .get("/users/2")
.then()
    .extract().response();

String email = response.jsonPath().getString("data.email");
```

Used when:

- One API depends on another
- Token generation
- Chained requests

🔷 REST Assured with POJO (Advanced but Clean)

```java
User user = new User("Piyush", "Developer");

given()
    .body(user)
.when()
    .post("/users")
.then()
    .statusCode(201);
```

REST Assured:

- Converts Java object → JSON
- Uses Jackson internally

🔷 Common Assertions

```java
.statusCode(200)
.statusLine("HTTP/1.1 200 OK")
.header("Content-Type", "application/json")
.time(lessThan(2000L))
```

🔷 Logging (VERY USEFUL)

```java
given()
    .log().all()
```

```
.when()
    .get("/users")
.then()
    .log().all();
```

Helps in:

- Debugging
- Interview explanation
- Understanding request/response flow

🔷 REST Assured vs Postman

Feature	REST Assured	Postman
Automation	✅ Yes	❌ Limited
CI/CD	✅ Yes	❌
Java Integration	✅ Native	❌
Code Reuse	✅ Yes	❌

👉 **Postman for manual testing**
👉 **REST Assured for automation**

🔷 Real-World Usage (Industry)

REST Assured is used in:

- Microservices testing
- Spring Boot backend testing
- CI/CD pipelines
- Regression testing
- Integration testing

Common combo:

```
css
```

Spring Boot + REST Assured + JUnit + Maven

🔷 Interview-Level Summary (Must Remember)

> **REST Assured is a Java-based DSL used to automate REST API testing.**
> **It simplifies HTTP calls, JSON parsing, and assertions using a fluent Given–When–Then syntax.**

If you want, next I can explain:

1️⃣ REST Assured **with Spring Boot real project**

2️⃣ REST Assured **interview questions**

3️⃣ REST Assured **end-to-end flow with JWT**

4️⃣ REST Assured **framework structure (industry level)**

Just tell me 🔥