Serialization and Deserialization in Rest Assured Framework

# What is Serialization?

Serialization is the process of converting a Java object into JSON so that it can be sent in the body of an API request (mostly in POST or PUT).

Example: Creating a new user by sending JSON to an API.

# POJO Class Example:

public class User {

private String name;

private int age;

public User(String name, int age) {

this.name = name;

this.age = age;

}

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public int getAge() { return age; }

public void setAge(int age) { this.age = age; }

}

# POST Request using Serialization:

User user = new User("Kumar", 30);

given()

.contentType("application/json")

.body(user)

.when()

.post("https://api.example.com/users")

.then()

.statusCode(201);

# What is Deserialization?

Deserialization is the process of converting JSON response into a Java object so that we can easily read and validate values.

# GET Request using Deserialization:

Response response = given()

.when()

.get("https://api.example.com/users/1");

User userFromResponse = response.as(User.class);

System.out.println(userFromResponse.getName());

System.out.println(userFromResponse.getAge());

# What is a POJO?

POJO = Plain Old Java Object

A simple Java class with private variables and public getters/setters.

Used for mapping data between Java and JSON (both serialization and deserialization).

# Summary Table:

Concept | Purpose | Involves JSON? | Direction | Real Use Case

----------------|----------------------------------------|----------------|-----------------|-----------------------------

Serialization | Convert Java Object → JSON | Yes | Java → JSON | Sending a POST request

Deserialization | Convert JSON → Java Object | Yes | JSON → Java | Reading a GET/POST response

POJO | Acts as data structure for mapping | Yes | Used for both | Structure for both flows

# Real-Life Scenario:

User user = new User("Adwaitha", 5);

Response postResponse = given()

.contentType("application/json")

.body(user)

.when()

.post("https://api.example.com/users")

.then()

.statusCode(201)

.extract().response();

User userCreated = postResponse.as(User.class);

System.out.println("Name from Response: " + userCreated.getName());

System.out.println("Age from Response: " + userCreated.getAge());

# Why Use These Concepts in Testing:

Use Case | Concept | Benefit

----------------------------------|-----------------|-------------------------------

POST/PUT requests with objects | Serialization | Avoid writing JSON manually

GET/POST response validation | Deserialization | Easily extract and assert

API chaining | Both | Clean and reusable code