



REST Assured - Basic

Request Types, Response, Serialization & Deserialization, Json and JsonPath



Introduction

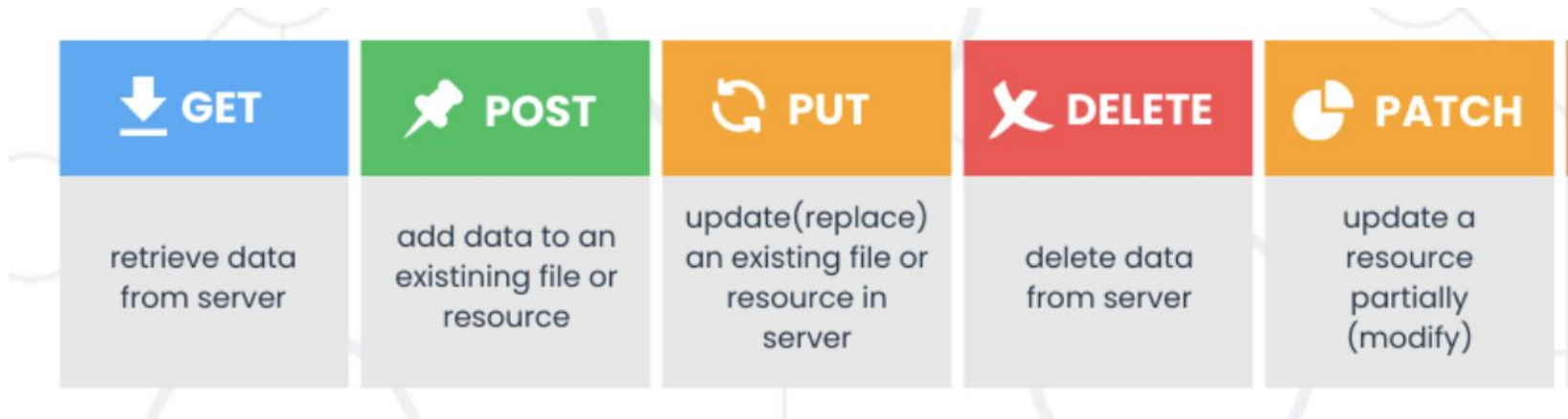
What is REST Assured?

REST Assured is an open-source Java library that is used for testing **RESTful** web services.

Features:

- Open source API Testing framework
- BDD like syntax, readable
- Seamless integration with other frameworks like TestNG and Cucumber

Http Request Methods





REST Assured Syntax

- **Given()**
 - The precondition for the request, for example, headers, contentType, body
- **When()**
 - The action for sending the request for example, request method, url, etc.
- **Then()**
 - Processing the response, assertion, extraction, logging, etc.

```
this.userId = given() RequestSpecification
    .contentType(ContentType.JSON)
    .body(payload)
    .when()
    .post(s: "https://regres.in/api/users") Response
    .then() ValidatableResponse
    .log().body()
    .statusCode(i: 201)
    .body(s: "name", equalTo(userName))
    .extract().body().jsonPath().getInt(path: "id");
```



Query & Path Parameter

Query Parameter

Rest Assured Demo / To do List / **Get List**

GET

▼

https://gorest.co.in/public/v2/todos?page=1&per_page=100

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Query Params

| | Key | Value |
|-------------------------------------|----------|-------|
| <input checked="" type="checkbox"/> | page | 1 |
| <input checked="" type="checkbox"/> | per_page | 100 |
| | Key | Value |

Path Parameter

Rest Assured Demo / To do List / **Get Detail**

GET

▼

https://gorest.co.in/public/v2/todos/14117

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Query Params

| | | |
|--|-----|-------|
| | Key | Value |
| | Key | Value |



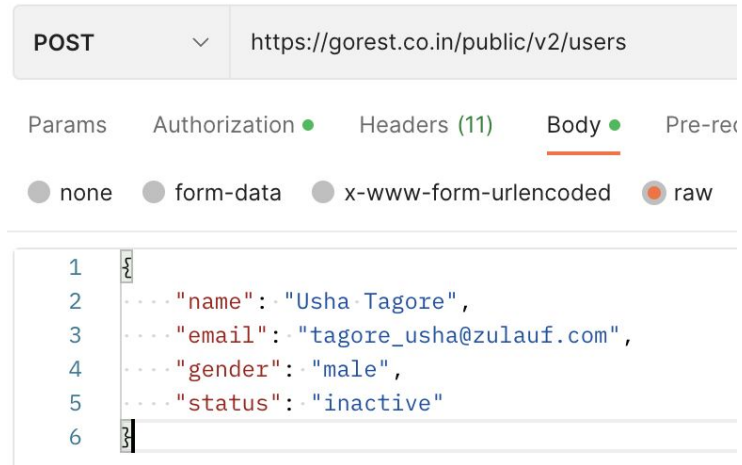
Sending a Request Body

What is Serialization?

Serialization is the process of converting an object into a byte stream, which can be written to a file, sent over a network, or stored in a database.

When does Serialization happen?

In REST Assured, when we pass an object as the request body, serialization happens automatically.



File Upload

<https://the-internet.herokuapp.com/upload>

POST

▼

https://the-internet.herokuapp.com/upload

Params

Authorization

Headers (10)

Body ●

Pre-request Script

Tests

Settings

☐ none

☒ form-data

☐ x-www-form-urlencoded

☐ raw

☐ binary

☐ GraphQL

| | Key | Value |
|-------------------------------------|------|-------------------|
| <input checked="" type="checkbox"/> | file | File System.png × |
| | Key | Value |



Parsing a Json Response

Deserialization

- What is deserialization ?
 - Deserialization is the process of converting a byte stream into an object.
- How to perform deserialization in REST Assured?
 - POJO
 - `JsonObject (Map)`
 - `Tree (JsonPath)`

POJO

```
1  "id": 1,  
2  "title": "Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops",  
3  "price": 109.95,  
4  "description": "Your perfect pack for everyday use and walks in the forest.  
5  "category": "men's clothing",  
6  "image": "https://fakestoreapi.com/img/81fPKd-2AYL._AC_SL1500_.jpg",  
7  "rating": {  
8    "rate": 3.9,  
9    "count": 120  
10 }  
11  
12
```

```
product = {TestPOJO$Product@4211} "TestPOJO.Product(id=1, title=Fjallraven -  
> f id = {Long@4218} 1  
> f title = "Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops"  
> f price = {Double@4220} 109.95  
> f category = "men's clothing"  
> f description = "Your perfect pack for everyday use and walks in the forest. Sta  
> f image = "https://fakestoreapi.com/img/81fPKd-2AYL._AC_SL1500_.jpg"  
> f rating = {TestPOJO$Rating@4224} "TestPOJO.Rating(rate=3.9, count=120)"  
    f rate = 3.9  
    f count = 120
```

JsonObject (Map)

```
1  "id": 1,  
2  "title": "Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops",  
3  "price": 109.95,  
4  "description": "Your perfect pack for everyday use and walks in the forest.  
5  "category": "men's clothing",  
6  "image": "https://fakestoreapi.com/img/81fPKd-2AYL._AC_SL1500_.jpg",  
7  "rating": {  
8      "rate": 3.9,  
9      "count": 120  
10 }  
11  
12
```

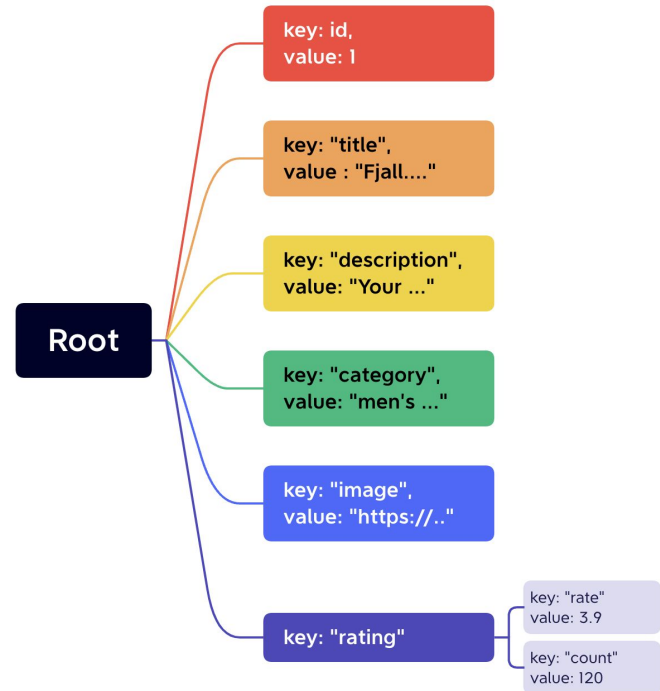
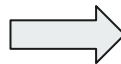
```
✓ js = {JsonObject@3923} {"image":"https://fakestoreapi.com/img/81fPKd-2AYL._AC_SL1500_.jpg"}  
  ✓ f map = {HashMap@3933} size = 7  
    > "image" -> "https://fakestoreapi.com/img/81fPKd-2AYL._AC_SL1500_.jpg"  
    > "price" -> {BigDecimal@3946} "109.95"  
    ✓ "rating" -> {JsonObject@3948} {"rate":3.9,"count":120}  
      > key = "rating"  
      ✓ value = {JsonObject@3948} {"rate":3.9,"count":120}  
        ✓ f map = {HashMap@4485} size = 2  
          > "rate" -> {BigDecimal@4491} "3.9"  
          > "count" -> {Integer@4493} 120  
        > "description" -> "Your perfect pack for everyday use and walks in the forest. Stash"  
        > "id" -> {Integer@3952} 1  
        > "title" -> "Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops"  
        > "category" -> "men's clothing"
```

JsonPath

Objects and Maps are good, but they are too cumbersome. What if I just want to know a specific value from the response?

JsonPath (View the json as a tree!)

```
1  {
2    "id": 1,
3    "title": "Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops",
4    "price": 109.95,
5    "description": "Your perfect pack for everyday use and walks in the forest.",
6    "category": "men's clothing",
7    "image": "https://fakestoreapi.com/img/81fPKd-2AYL._AC_SL1500_.jpg",
8    "rating": {
9      "rate": 3.9,
10     "count": 120
11   }
12 }
```



Challenge: Json Array

```
1  {
2    {
3      "id": 1,
4      "title": "Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops",
5      "price": 109.95,
6      "description": "Your perfect pack for everyday use and walks in the forest. Stash your
7      "category": "men's clothing",
8      "image": "https://fakestoreapi.com/img/81fPKd-2AYL._AC_SL1500_.jpg",
9      "rating": {
10         "rate": 3.9,
11         "count": 120
12     }
13 },
14 {
15     "id": 2,
16     "title": "Mens Casual Premium Slim Fit T-Shirts ",
17     "price": 22.3,
18     "description": "Slim-fitting style, contrast raglan long sleeve, three-button henley pl
19         wearing. And Solid stitched shirts with round neck made for durability and a great
20         Henley style round neckline includes a three-button placket.",
21     "category": "men's clothing",
22     "image": "https://fakestoreapi.com/img/71-3HjGNDUL._AC_SY879._SX._UX._SY._UY_.jpg",
23     "rating": {
24         "rate": 4.1,
25         "count": 259
26     }
27 }
```

