Rest Assured advanced concepts

OAuth 2.0

API Contract download

****
Authorization Server EndPoint:
https://rahulshettyacademy.com/oauthapi/oauth2/resourceOwner/token

HTTP Method : POST

Form parameters :
client_id:692183103107-
p0m7ent2hk7suguv4vq22hjcfhcr43pj.apps.googleusercontent.com
client_secret:erZOWM9g3UtwNRj340YYaK_W
grant_type:client_credentials
scope:trust
****

GetCourseDetails EndPoint (Secured by OAuth) :
https://rahulshettyacademy.com/oauthapi/getCourseDetails

HTTP Method : GET
Query Parameter : access_token

> First we will make call to Authorization Server(AS) Endpoint and this server will give us the token which will be used to access GET endpoint to fetch results

RSA_ClientCredentialsOauth.post-man_collection.json
JSON · 8 KB

Serialization & Deserialization of Requests/ Responses with POJO classes

>Serialization in Rest Assured context is a process of converting a Java object into Request body (Payload).

>Rest Assured also supports **deserialization** by converting Response body back to Java object.

>**Advantages:**

- Easy to parse and extract response (Json/XML) values if they are

wrapped as Java object.
- User friendly methods can be created which makes code more readable.

>**Design Approach:**
- Java object is constructed with the support of POJO classes
- POJO classes are created based on the request/ Response payload

>**What additional libraries required?**
For JSON you need to have either Jackson, Jackson2, GSon or Johnzon in the class path and for XML you need JAXB.

POJO - Plan Old Java Object

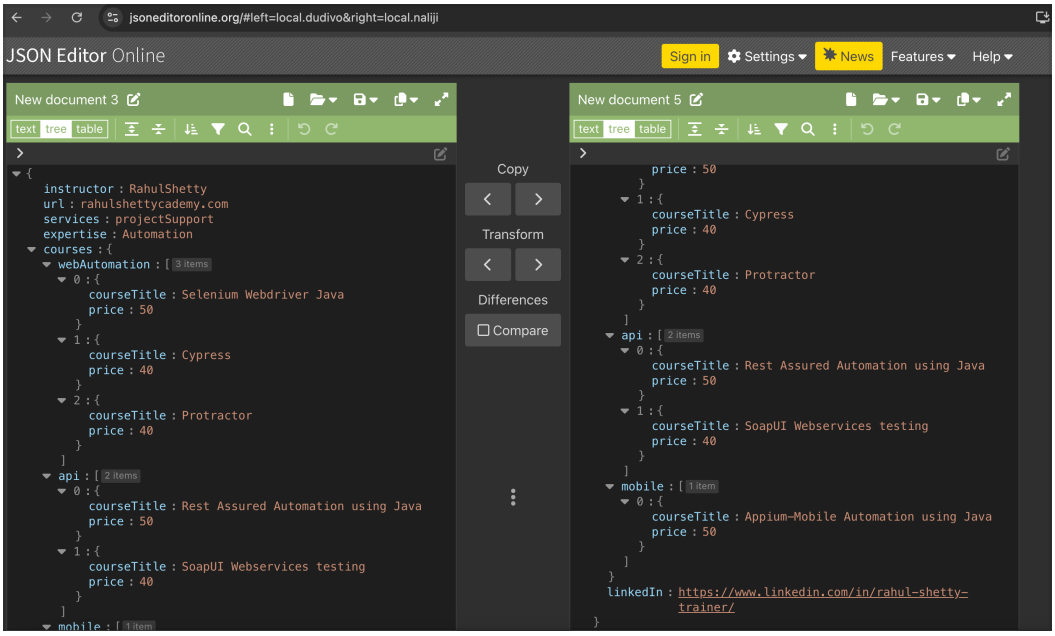
Plain old Java object (POJO) is a class definition that is not tied to any Java framework so any Java program can use it. A POJO has no particular naming convention for properties and methods, or any other special restrictions. Their primary advantage is their reusability and simplicity

Jackson databind maven dependency -> This give capability of serialization & deserialization

Example Json:
https://jsonformatter.org/json-editor

```
{
   "instructor": "RahulShetty",
   "url": "rahulshettycademy.com",
   "services": "projectSupport",
   "expertise": "Automation",
   "courses": {
      "webAutomation": [
         {
            "courseTitle": "Selenium Webdriver Java",
            "price": "50"
         },
         {
            "courseTitle": "Cypress",
            "price": "40"
         },
         {
            "courseTitle": "Protractor",
            "price": "40"
         }
      ],
      "api": [
         {
```

```
            "courseTitle": "Rest Assured Automation using Java",
            "price": "50"
        },
        {
            "courseTitle": "SoapUI Webservices testing",
            "price": "40"
        }
    ],
    "mobile": [
        {
            "courseTitle": "Appium-Mobile Automation using Java",
            "price": "50"
        }
    ]
  },
  "linkedIn": "https://www.linkedin.com/in/rahul-shetty-trainer/"
}
```

Deserialization: It is used after we get response from JSON and then we deserialize to get values

Serialization: In this we convert Java object to JSON.
We create one java object and set all the values with setters and we will give it to Rest Assured code, so that Rest Assured convert java object into JSON body to submit to the API

ADD-DeletePlaceAPIs.docx
Word Document · 12 KB

builder.docx
Word Document · 15 KB

Request and Response Spec Builders:

builder.docx
Word Document · 15 KB

ADD PLACE :

RestAssured.*baseURI*="XXXX";

Response res=*given*().queryParam("key", "qaclick123").header("Content-Type","application/json")

.body(add_place_json)
.when().post("/maps/api/place/add/json").
then().assertThat().statusCode(200). contentType("application/json")
extract().response();

Request Spec Builder

RequestSpecBuilder class is used to build some request and we will use that request again and again.

```
RequestSpecBuilder req = new
RequestSpecBuilder().setContentType(ContentType.JSON)
        .setBaseUri("XXXX")
        .addQueryParam("key","qaclick123")
        .build();
```

given().spec (req).body(add_place_json) .post("/maps/api/place/add/json).

Response Spec Builder:

ResponseSpecBuilder class is used to build some response and we will use that response again and again.

```
ResponseSpecBuilder res = new
```

```
ResponseSpecBuilder().expectStatusCode(200).expectContentTyp
e(ContentType.JSON). build();
```

then().spec(res).extract().response();

Rewriting Test with Request and Response Spec Builder :

```
given().spec(req).body(add_place_json).post("/maps/api/
place/add/json).
then().spec(res).extract().response();
```

NOTE:

```
 ResponseSpecification responseSpec = new
ResponseSpecBuilder().expectStatusCode(200).build();
 RequestSpecification requestSpec = new
RequestSpecBuilder().addParam("parameter1",
"value1").build();
```