REST API

Rest API Syllabus:

1. Introduction
2. Configure local environment for Restful API
    a) Install MY SQL
    b) Tomcat configuration
    c) Deploy .war file in TOMCAT server
3. Launch swagger file
4. Execute available End point in swagger
5. JSON document
6. Install postman tool
7. Execute available End points in postman and understanding all status codes
8. Validate Responses in postman
9. Create collection in postman
10. Schema validation in postman

How to create a data base in MYSQL?

Click on this link https://dev.mysql.com/downloads/installer/ **or** go to any website type MYSQL download you can download from there also → install the MySQL→(give the port number as 3306 if you facing any issue with port no type **ctrl+shift+esc** find mysql.exe click on that click on start server) once click on start server it's running normally → once software is installed type "create database Santosh" → click on schemas → click on refresh to check database is created or not.

How to create TOMCAT server in your local machine?

Click on this link https://tomcat.apache.org/download-90.cgi or go to any website type tomcat → click on official site → select tomcat 9 → scroll down →click on 64-bit windows zip (it's start downloading)→ copy that downloaded file → go to any drive in your laptop → create a folder called (Server) → paste it →right click and extract that file (Extract here) → after completion of extract remove zip file→ rename the folder as (apache-tomcat).

How to set up the environment for tomcat?

Open your system environment variable → in system variable → create

Step 1: create CATALINA_HOME

Variable name: CATALINA_HOME

Variable value: c:\server\apache-tomcat

Step 2: Edit path variable

Select **“**path” variable and click on edit button in new windows click on “new” button

And provide this path (C:\Sever\apache-tomcat\bin)

How to change the tomcat port number?

Go to Apache-tomcat → click on “conf” folder → click on server.xml → right click and open this on notepad++ → in line number 69(change the port no as 8083 or any other) →save it and close it.

How to make sure the tomcat has configured properly or not?

Go to cmd → write “startup” → enter→ open the browser → type http:// local host:8083→ see the tomcat page home page.

How to stop the server?

Open server command prompt → press ctrl+c.

How to deployee .war file?

Go to below link to download .war file→

https://drive.google.com/drive/folders/1tUg0lUQGSgEtupj67_NzkosdyCe-_9Eb?usp=drive_link

or

Go to temp folder → clean your temp folder → right click and click on git bash here → and type git clone with below linkhttps://github.com/pgudi/ExampleOctober3rd2023Repository.git

Open the folder(spring boot deployment) → click on .war file and copy war file→ go to tomcat configure path there is a folder called web apps under web apps paste it → open cmd prompt → type → startup→ open sever folder inside server folder click on web apps folder → copy that folder name (springboot-sgsoftwaretestinginstitute-sgtesting)→ paste it in notepad → and add local host

name also like (http://localhost:8083/springboot-sgsoftwaretestinginstitute-sgtesting/swagger-ui.html) → copy this link and paste it in any browser.

You can modify data base from above web site or using postman.

How to download and install postman tool?

Open any browser navigate the postman url and download the 64 bit of postman software. Or click on https://www.postman.com/downloads/ this links to download postman tool. (create an account in postman)

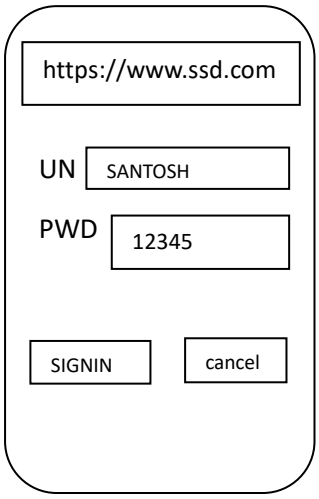
How to create a collection in postman?

Click on collection tab in left hand side and click on '+' symbol and provide the collection name and click outside it create the collection kclose it. Expand collection name create and click on 3 dots and add request (click) provide name as Create_user_request.

We have to provide URL(to get the URL go to swagger except swagger select every thing copy and add which we have to execute the tests <name> like (http://localhost:8083/springboot-sgsoftwaretestinginstitute-sgtesting/users)

Choose from dropdown and choose JSON.

In post of swaggers copy paste details and provide the appropriate value and click on send.



<credentials>

<login>

<UN>Santosh</un>

<pwd>12345</pwd>

</login>

</credentials>

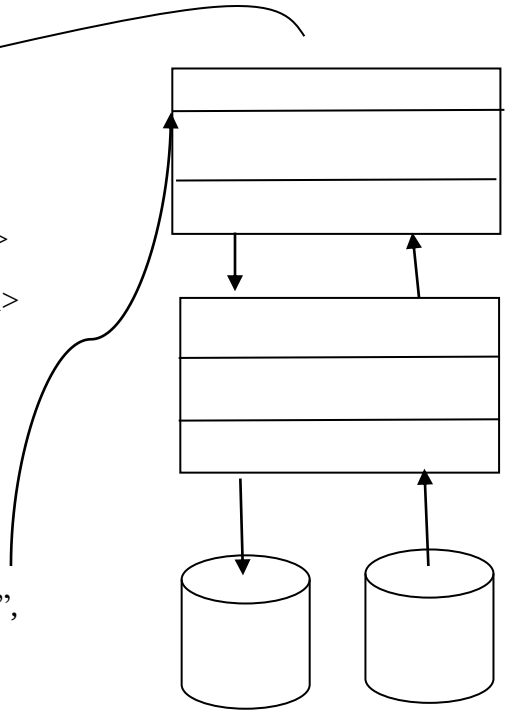
Or

{

"UN" =" Santosh",

"pwd"="12345"

}



Case 1: -

```
<html>
  <head>
    <title> Sample web page</title>
      <script>
            Var Student = {
                        “fname”:”santosh”
                        “age”:21;
                        “city name”:”bangalore”
                    };
             document.write(“First name: ”+student.fname+”<br/>”);
             document.write(“Age: ”+student.age+”<br/>”);
             document.write(“City name: ”+student.cityname+”<br/>”);
             document.write(“First name: ”+student[“fname”]+”<br/>”);
             document.write(“Age: ”+student[“age”]+”<br/>”);
             document.write(“City name: ”+student[“cityname”]+”<br/>”);
    </script>
  </head>
<body>
        <h1 align =”centre”>Test web page</h1>
</body>
</html>
```

Case 2: -

```
<html>
  <head>
     <title> Sample web page</title>
       <script>
              Var Student = {
                                  “fname”:”santosh”
                                  “age”:21;
                                  “cityname”:”bangalore”
                                },
                               {
                                  “fname”:”darshan”
                                  “age”:19;
                                  “cityname”:”mangalore”
                                },
                                };
                document.write(“First name: ”+student[0].fname+”<br/>”);
                document.write(“Age: ”+student[0].age+”<br/>”);
                document.write(“City name: ”+student[0].cityname+”<br/>”);
                document.write(“First name: ”+student[1].fname+”<br/>”);
                document.write(“Age: ”+student[1].age+”<br/>”);
                document.write(“City name: ”+student[1].cityname+”<br/>”);
    </script>
  </head>
<body>
        <h1 align =” centre”>Test web page</h1>
        <p> This web page describes JSON object </p>
</body>
</html>
```

Case 3: -

```
<html>
  <head>
     <title> Sample web page</title>
       <script>
              Var Student = {
                                  “fname”:”santosh”
                                  “age”:21;
                                  “cityname”:”bangalore”
                                },
                               {
                                  “fname”:”darshan”
                                  “age”:19;
                                  “cityname”:”mangalore”
                                },
                                };
                document.write(“First name: ”+student.santu.fname+”<br/>”);
                document.write(“Age: ”+student.santu.age+”<br/>”);
                document.write(“City name: ”+student.santu.cityname+”<br/>”);
                document.write(“First name: ”+student.darsh.fname+”<br/>”);
                document.write(“Age: ”+student.darsh.age+”<br/>”);
                document.write(“City name: ”+student.darsh.cityname+”<br/>”);
    </script>
  </head>
<body>
         <h1 align =” centre”>Test web page</h1>
         <p> This web page describes JSON object </p>
</body>
</html>
```

JSON Tutorial:

JSON stands for JavaScript Object Notation. JSON objects are used for transferring data between server and client, XML serves the same purpose. However JSON objects have several advantages over XML and we are going to discuss them in this tutorial along with JSON concepts and its usages.

Let's have a look at the piece of a JSON data: It basically has key-value pairs.

```
var chaitanya = {
   "firstName" : "Chaitanya",
   "lastName" : "Singh",
   "age" :  "28"
};
```

Features of JSON:

- It is light-weight
- It is language independent
- Easy to read and write
- Text based, human readable data exchange format

Why use JSON?

Standard Structure: As we have seen so far that JSON objects are having a standard structure that makes developers job easy to read and write code, because they know what to expect from JSON.

Light weight: When working with AJAX, it is important to load the data quickly and asynchronously without requesting the page re-load. Since JSON is light weighted, it becomes easier to get and load the requested data quickly.

Scalable: JSON is language independent, which means it can work well with most of the modern programming language. Let's say if we need to change the server side language, in that case it would be easier for us to go ahead with that change as JSON structure is same for all the languages.

JSON vs. XML

Let see how JSON and XML look when we store the records of 4 students in a text based format so that we can retrieve it later when required.

JSON style:

```
{"students":[
   {"name":"John", "age":"23", "city":"Agra"},
   {"name":"Steve", "age":"28", "city":"Delhi"},
   {"name":"Peter", "age":"32", "city":"Chennai"},
   {"name":"Chaitanya", "age":"28", "city":"Bangalore"}
]}
```

XML style:

```
<students>
  <student>
    <name>John</name> <age>23</age> <city>Agra</city>
  </student>
  <student>
    <name>Steve</name> <age>28</age> <city>Delhi</city>
  </student>
  <student>
    <name>Peter</name> <age>32</age> <city>Chennai</city>
  </student>
  <student>
    <name>Chaitanya</name> <age>28</age> <city>Bangalore</city>
  </student>
</students>
```

As you can clearly see JSON is much more **light-weight** compared to XML. Also, in JSON we take advantage of arrays that is not available in XML.

JSON data structure types and how to read them:

1. JSON objects
2. JSON objects in array
3. Nesting of JSON objects

1) JSON objects:

```
var chaitanya = {
  "name" : "Chaitanya Singh",
  "age" : "28",
  "website" : "beginnersbook"
};
```

The above text creates an object that we can access using the variable chaitanya. Inside an object we can have any number of key-value pairs like we have above. We can access the information out of a JSON object like this:

```
document.writeln("The name is:  " +chaitanya.name);
document.writeln("his age is: " + chaitanya.age);
document.writeln("his website is: "+ chaitanya.website);
```

2) JSON objects in array

In the above example we have stored the information of one person in a JSON object suppose we want to store the information of more than one person; in that case we can have an array of objects.

```
var students = [{
   "name" : "Steve",
   "age" :  "29",
   "gender" : "male"

},
{
   "name" : "Peter",
   "age" : "32",
   "gender" : "male"

},
{
   "name" : "Sophie",
   "age" : "27",
   "gender" : "female"
}];
```

To access the information out of this array, we do write the code like this:

```
document.writeln(students[0].age); //output would be: 29
document.writeln(students[2].name); //output: Sophie
```

You got the point, right? Let's carry on with the next type.

3) Nesting of JSON objects:

Another way of doing the same thing that we have done above.

```
var students = {
  "steve" : {
  "name" : "Steve",
  "age" :  "29",
  "gender" : "male"
},

"pete" : {
  "name" : "Peter",
  "age" : "32",
  "gender" : "male"
},

"sop" : {
  "name" : "Sophie",
  "age" : "27",
  "gender" : "female"
}
}
```

Do like this to access the info from above nested JSON objects:

```
document.writln(students.steve.age); //output: 29
document.writeln(students.sop.gender); //output: female
```

JSON & JavaScript:

JSON is considered as a subset of JavaScript but that does not mean that JSON cannot be used with other languages. In fact it works well with PHP, Perl, Python, Ruby, Java, Ajax and many more.

Just to demonstrate how JSON can be used along with JavaScript, here is an example:
If you have gone though the above tutorial, you are familiar with the JSON structures. A JSON file type is .json

How to read data from json file and convert it into a JavaScript object?
We have two ways to do this.
1) Using eval function, but this is not suggested due to security reasons (malicious data can be sent from the server to the client and then eval in the client script with harmful effects).
2) Using JSON parser: No security issues plus it is faster than eval. Here is how to use it:

```
var chaitanya = {
"name" : "Chaitanya Singh",
"age" : "28",
"website" : "beginnersbook"
};
```

We are converting the above JSON object to javascript object using JSON parser:

```
var myJSObject = JSON.parse(chaitanya);
```

How to convert JavaScript object to JSON text?
By using method stringify

```
var jsonText= JSON.stringify(myJSObject);
```

How to Validate the JSON Response:

In Postman Tool click on Test tab

In Right side we can view JavaScript SNIPPETS

the folowing options are avilable to validate the JSON Response

1.Status code: Code is 200

2. Response Body: JOSN Value Check

Sample Code:

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Get a PArticular User", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.first_name).to.eql('Welcome');
    pm.expect(jsonData.last_name).to.eql('User1');
    pm.expect(jsonData.address).to.eql('Upparwadi Sindhanur');
});
```

++

Status Codes in API Testing:

2XX : Successful

1). 200 OK	The request is OK (this is the standard response for successful HTTP requests)
2). 201 Created	The request has been fulfilled, and a new resource is created
3). 202 Accepted	The request has been accepted for processing, but the processing has not been completed
4). 204 No Content	The request has been successfully processed, but is not returning any content

4xx: Client Error

1). 400 Bad Request The request cannot be fulfilled due to bad syntax

2). 401 Unauthorized The request was a legal request, but the server is refusing to respond to it

3). 404 Not Found The requested page could not be found but may be available again in the future

4). 408 Request Timeout The server timed out waiting for the request

5xx: Server Error

1) 500 Internal Server Error A generic error message, given when no more specific message is suitable

2) 502 Bad Gateway The server was acting as a gateway or proxy and received an invalid response from the upstream server

3) 503 Service Unavailable The server is currently unavailable

4). 505 HTTP Version Not SupportedThe server does not support the HTTP protocol version used in the request

HTTP Methods Description:

GET: The GET http method used to FETCH or retrive resource from Server

POST: the POST http method is used to create a new Resource

PUT: the PUT http method is used to modify or update the existing resource

if resource is not available then it creates the Resource

PATCH: the PATCH http method is used to modify or update the existing resource, if resource is not available then it displays error message.

DELETE: the DELETE http method is used to remove or erase the resource from the server permanently

***How to Send the Request and get the Response ?

In Postman in Body section select Raw and File content type as JSON and we have to create the JOSN Request, suppose if the request need for SignIn then it can be as follows:

```
{
  "email":"santoshubhale@gmail.com",
  "password":"India111"
}
```

Once after creating the Request provide the required Authorization and Run the Request.

How to Create Collections and How to run the collections?

Ans:

In Postman for each request, there is a Save option then we can save each request in a Collection [Collection is a container,it has multiple requests]

How to run the Collection?

Ans:

In Postman there is a "Runner" option once we clicked on that option, it displays

Collection Runner window. there select the specific collection Name. And Run the Collection.

Head:- head is a HTTP method. It validates or verify the existing of resources in server. Its acts as similar to the get method

How to configure the environment variable in postman

Click on Environment quick look → click on global → click on Add → click on URL → except users copy everything and paste it initial value → click on save → in address bar remove everything just type {{URL}}/users.

Schema validation:

QA never generate the JSON Schema

Go to website → navigate the URL (JSONSchema.net)

JSON Schema:

JSON Schema is a contract for JSON document that defines the expected data types and format of each field in the response.

Why is JSON Schema Validation required?

JSON Schema Validation is required because:

1. We monitor API responses and ensure that the format that we are getting is same as the expected one.

2. We get alert whenever there is any breaking change in JSON response.

3. We use JSON Schema to construct a model of API response and it makes easier to validate that API is returning the valid data.

How to generate the JSON Schema for Postman:

---

1. Navigate to the URL:

https://jsonschema.net/app/schemas/0

2. Paste the JSON REsponse on left hand side and click on submit button

3. It generates json schema on right handside.

4. Copy the JSON Schema and paste it into the Postman Tests tab

5. We have to write the below code to valida the json schema

```
pm.test("Validate schema", () => {
        pm.response.to.have.jsonSchema(schema);
});
```

++

const schema=


```
{
  "type": "array",
   "default": [],
   "title": "Root Schema",
   "items": {
      "type": "object",
      "default": {},
      "title": "A Schema",
      "required": [
         "id",
         "first_name",
         "last_name",
         "email",
         "phone_number",
         "address",
         "state",
         "zipcode",
         "created_at",
         "updated_at",
         "student_id"
      ],
      "properties": {
         "id": {
            "type": "integer",
            "default": 0,
            "title": "The id Schema",
            "examples": [
```

```
        326569
    ]
},
"first_name": {
    "type": "string",
    "default": "",
    "title": "The first_name Schema",
    "examples": [
        "DemoUser1"
    ]
},
"last_name": {
    "type": "string",
    "default": "",
    "title": "The last_name Schema",
    "examples": [
        "User1"
    ]
},
"email": {
    "type": "string",
    "default": "",
    "title": "The email Schema",
    "examples": [
        "demo@gmail.com"
    ]
},
"phone_number": {
    "type": "string",
    "default": "",
```

```
    "title": "The phone_number Schema",
    "examples": [
        "9800012345"
    ]
},
"address": {
    "type": "string",
    "default": "",
    "title": "The address Schema",
    "examples": [
        "RPC Layout"
    ]
},
"state": {
    "type": "string",
    "default": "",
    "title": "The state Schema",
    "examples": [
        "Karnataka"
    ]
},
"zipcode": {
    "type": "string",
    "default": "",
    "title": "The zipcode Schema",
    "examples": [
        "560040"
    ]
},
"created_at": {
```

```
        "type": "string",
        "default": "",
        "title": "The created_at Schema",
        "examples": [
            "2022-05-04T05:39:46.000Z"
        ]
    },
    "updated_at": {
        "type": "string",
        "default": "",
        "title": "The updated_at Schema",
        "examples": [
            "2022-05-04T05:39:46.000Z"
        ]
    },
    "student_id": {
        "type": "integer",
        "default": 0,
        "title": "The student_id Schema",
        "examples": [
            1449
        ]
    }
},
"examples": [{
    "id": 326569,
    "first_name": "DemoUser1",
    "last_name": "User1",
    "email": "demo@gmail.com",
    "phone_number": "9800012345",
```

```
            "address": "RPC Layout",
            "state": "Karnataka",
            "zipcode": "560040",
            "created_at": "2022-05-04T05:39:46.000Z",
            "updated_at": "2022-05-04T05:39:46.000Z",
            "student_id": 1449
        }]
    },
    "examples": [
        [{
            "id": 326569,
            "first_name": "DemoUser1",
            "last_name": "User1",
            "email": "demo@gmail.com",
            "phone_number": "9800012345",
            "address": "RPC Layout",
            "state": "Karnataka",
            "zipcode": "560040",
            "created_at": "2022-05-04T05:39:46.000Z",
            "updated_at": "2022-05-04T05:39:46.000Z",
            "student_id": 1449
        }]
    ]
}

pm.test("Validate schema", () => {
            pm.response.to.have.jsonSchema(schema);
});
```

Validate the JSON Response

Below are the validation we can add in Tests Options:

Naviagte to snippets option in the right hand side.

1)Select Status code is 200 option below snippet gets added

```
pm.test('Status code is 200', function () {
    pm.response.to.have.status(200);
})
```

2)Select Response Body :JSON Value Check below snippet gets added

```
pm.test("Your test name", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.data[0].email).to.eql('michael.lawson@reqres.in');
});
```

3)Select Response Body:Contains String

```
pm.test("Body matches string", function () {
    pm.expect(pm.response.text()).to.include("Ferguson");
});
```

Json Body Validation for below 3 json Responses

1)Json Object in Array

Below are the reponse for the Request:https://reqres.in/api/users?page=2&per_page=2


```
{
    "page": 2,
    "per_page": 2,
    "total": 12,
    "total_pages": 6,
```

```
    "data": [
        {
            "id": 3,
            "email": "emma.wong@reqres.in",
            "first_name": "Emma",
            "last_name": "Wong",
            "avatar": "https://reqres.in/img/faces/3-image.jpg"
        },
        {
            "id": 4,
            "email": "eve.holt@reqres.in",
            "first_name": "Eve",
            "last_name": "Holt",
            "avatar": "https://reqres.in/img/faces/4-image.jpg"
        }
    ],
    "support": {
        "url": "https://reqres.in/#support-heading",
        "text": "To keep ReqRes free, contributions towards server costs are appreciated!"
    }
}
```


Below are the validation:

```
pm.test('Status code is 200', function () {
    pm.response.to.have.status(200);
})

pm.test("Your test name", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.data[0].email).to.eql('emma.wong@reqres.in');
});
```

```
pm.test("Body matches string", function () {
    pm.expect(pm.response.text()).to.include("Eve");
});
```

2)Nested Json Object

Below are the reponse for the request https://reqres.in/api/users?id=2

Reponse:


```
{
    "data": {
        "id": 2,
        "email": "janet.weaver@reqres.in",
        "first_name": "Janet",
        "last_name": "Weaver",
        "avatar": "https://reqres.in/img/faces/2-image.jpg"
    },
    "support": {
        "url": "https://reqres.in/#support-heading",
        "text": "To keep ReqRes free, contributions towards server costs are appreciated!"
    }
}
```


Json Validation:

```
pm.test('Status code is 200', function () {
    pm.response.to.have.status(200);
})

pm.test("Your test name", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.support.url).to.eql('https://reqres.in/#support-heading');
```

});

3)Json Object :

https://reqres.in/api/users

Create api:

Request body:

```
{
    "name": "Santosh “,
    "job": "QA",
    "id": "598",
    "createdAt": "2024-01-24T07:40:09.483Z"
}
```

Response for the above request:

```
{
    "name": "Suhas “,
    "job": "QA",
    "id": "598",
    "createdAt": "2024-01-24T07:40:09.483Z"
}
```

Json validation:

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(201);
pm.test("Your test name", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.job).to.eql('QA');
});
});
pm.test("Body matches string", function () {
```

```
    pm.expect(pm.response.text()).to.include("Santosh");
});
```

Rest assured/ API Automation.

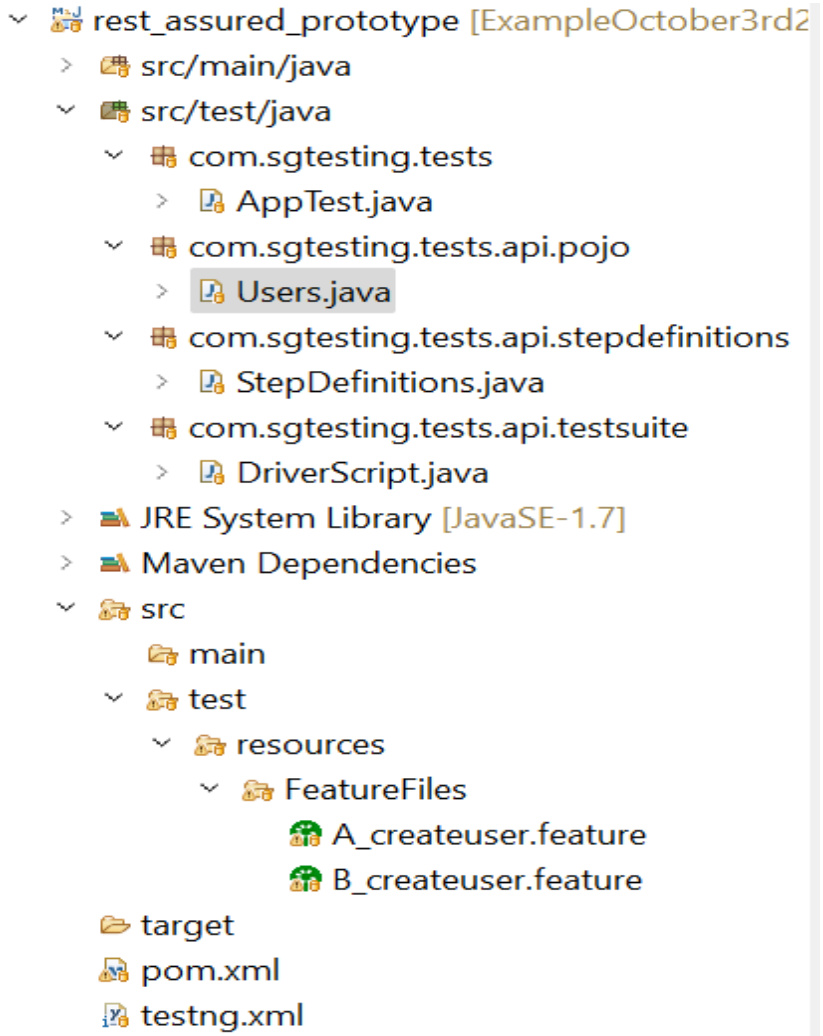
POJO (plain old java object) class in Rest assured

It stands for plain old java object. By using POJO class object along with Jackson Library we can create JSON object programmatically.

Note :- Jackson file can download in MVN repository → type Jackson→ com.fasterxml.jacksoncoe>> Jackson -data bind

Create a pojo class: -

Program: -



User.java:

```
package com.sgtesting.tests.api.pojo;

public class Users {

	private String address;
	private String emailId;
	private String firstName;
	private String  lastName;
	private String  phoneNumber;
	private String stateName;
	private String userRole;
	private Integer zipcode;

	public Users() {

	}

	public Users(String address, String emailId, String firstName, String
lastName, String phoneNumber,
			String stateName, String userRole, Integer zipcode) {
		super();
		this.address = address;
		this.emailId = emailId;
		this.firstName = firstName;
		this.lastName = lastName;
		this.phoneNumber = phoneNumber;
		this.stateName = stateName;
		this.userRole = userRole;
		this.zipcode = zipcode;
	}
	public String getAddress() {
		return address;
	}
	public void setAddress(String address) {
		this.address = address;
	}
	public String getEmailId() {
		return emailId;
	}
	public void setEmailId(String emailId) {
		this.emailId = emailId;
	}
	public String getFirstName() {
		return firstName;
	}
	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}
	public String getLastName() {
		return lastName;
	}
	public void setLastName(String lastName) {
		this.lastName = lastName;
	}
	public String getPhoneNumber() {
		return phoneNumber;
	}
	public void setPhoneNumber(String phoneNumber) {
		this.phoneNumber = phoneNumber;
```

```
	}
	public String getStateName() {
		return stateName;
	}
	public void setStateName(String stateName) {
		this.stateName = stateName;
	}
	public String getUserRole() {
		return userRole;
	}
	public void setUserRole(String userRole) {
		this.userRole = userRole;
	}
	public Integer getZipcode() {
		return zipcode;
	}
	public void setZipcode(Integer zipcode) {
		this.zipcode = zipcode;
	}
}
```

Step definition:

```
package com.sgtesting.tests.api.stepdefinitions;

import org.testng.Assert;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.sgtesting.tests.api.pojo.Users;

import cucumber.api.java.en.And;
import cucumber.api.java.en.Given;
import cucumber.api.java.en.Then;
import cucumber.api.java.en.When;
import io.restassured.RestAssured;
import io.restassured.path.json.JsonPath;
import io.restassured.response.Response;
import io.restassured.specification.RequestSpecification;

public class StepDefinitions {
	public static RequestSpecification httpRequest=null;
	public static Users userobj=null;
	public static String userjsonobject=null;
	public static Response response=null;
	public static String user_id=null;
	/**
	 * Step Name : I specify the post restful api endpoint url
	 */
	@Given("^I specify the post restful api endpoint url$")
	public void I_specify_the_post_restful_api_endpoint_url()
	{
		try
		{
	RestAssured.baseURI="http://localhost:8082/springboot-
	sgsoftwaretestinginstitute-sgtesting/users";
		}catch (Exception e)
		{
			e.printStackTrace();
		}
	}
```

```
/**
 * Step Name: I access given method to get RequestSpecification
object
 */
@And("^I access given method to get RequestSpecification object$")
public voidI_access_given_method_to_get_RequestSpecification_object()
{
	try
	{
		httpRequest=RestAssured.given();
	}catch (Exception e)
	{
		e.printStackTrace();
	}
}

/**
 * Step Name:I create user object using user pojo class
 */
@And("^I create user object using user pojo class$")
public void I_create_user_object_using_user_pojo_class()
{
	try
	{
		userobj=new Users();
		userobj.setAddress("12th Main,7th Cross Vijayanagar");
		userobj.setEmailId("vinu@gmail.com");
		userobj.setFirstName("vinith");
		userobj.setLastName("B");
		userobj.setPhoneNumber("9010203040");
		userobj.setStateName("Karnataka");
		userobj.setUserRole("QA Engineer");
		userobj.setZipcode(560098);
	}catch (Exception e)
	{
		e.printStackTrace();
	}
}

/**
 * Step Name: I convert user object into user json object
 */
@And("^I convert user object into user json object$")
public void I_convert_user_object_into_user_json_object()
{
	try
	{
		ObjectMapper mapper=new ObjectMapper();
userjsonobject=mapper.writerWithDefaultPrettyPrinter().writeValueAsSt
ring(userobj);
	}catch (Exception e)
	{
		e.printStackTrace();
	}
}

/**
 * Step Name: I provide header and body contents
 */
@And("^I provide header and body contents$")
```

```
public void I_provide_header_and_body_contents()
{
	try
	{
		httpRequest.header("Content-Type", "application/json");
		httpRequest.body(userjsonobject);
	}catch (Exception e)
	{
		e.printStackTrace();
	}
}

/**
 * Step Name: I access post http method to send request to server
 */
@When("^I access post http method to send request to server$")
public void I_access_post_http_method_to_send_request_to_server()
{
	try
	{
		response=httpRequest.post();
		JsonPath jpath=response.jsonPath();
		user_id=jpath.getString("id");
	}catch (Exception e)
	{
		e.printStackTrace();
	}
}

/**
 * Step Name: I get status code as 201
 */
@Then("^I get status code as 201$")
public void I_get_status_code_as_201()
{
	try
	{
		int statuscode=response.getStatusCode();
		Assert.assertEquals(201, statuscode);
	}catch (Exception e)
	{
		e.printStackTrace();
	}
}

/**
 * Step Name: I specify the get restful api endpoint url
 */
@Given("^I specify the get restful api endpoint url$")
public void I_specify_the_get_restful_api_endpoint_url()
{
	try
	{
RestAssured.baseURI="http://localhost:8082/springboot-
sgsoftwaretestinginstitute-sgtesting/users/"+user_id;

	}catch (Exception e)
	{
		e.printStackTrace();
	}
}
```

```
	/**
	 * Step Name: I access get http method to retrive resource from
server
	 */
	@When("^I access get http method to retrive resource from server$")
	public voidI_access_get_http_method_to_retrive_resource_from_server()
	{
		try
		{
			response=httpRequest.get();
			String content=response.asPrettyString();
			System.out.println(content);
		}catch (Exception e)
		{
			e.printStackTrace();
		}
	}

	/**
	 * Step Name: I validate content in response object
	 */
	@Then("^I validate content in response object$")
	public void I_validate_content_in_response_object()
	{
		try
		{
			JsonPath jpath=response.jsonPath();
			String address=jpath.getString("address");
	Assert.assertEquals("12th Main,7th Cross Vijayanagar", address);

			String emailid=jpath.getString("emailId");
			Assert.assertEquals("vinu@gmail.com", emailid);

			String userrole=jpath.getString("userRole");
			Assert.assertEquals("QA Engineer", userrole);
		}catch (Exception e)
		{
			e.printStackTrace();
		}
	}

	/**
	 * Step Name: I get status code as 200
	 */
	@And("^I get status code as 200$")
	public void I_get_status_code_as_200()
	{
		try
		{
			int statuscode=response.getStatusCode();
			Assert.assertEquals(200, statuscode);
		}catch (Exception e)
		{
			e.printStackTrace();
		}
	}

	/**
	 * Step Name: I specify the delete restful api endpoint url
	 */
```

```
	@Given("^I specify the delete restful api endpoint url$")
	public void I_specify_the_delete_restful_api_endpoint_url()
	{
		try
		{
	RestAssured.baseURI="http://localhost:8082/springboot-
	sgsoftwaretestinginstitute-sgtesting/users/"+user_id;
		}catch (Exception e)
		{
			e.printStackTrace();
		}
	}

	/**
	 * Step Name: I access delete http method to delete resource from
server
	 */
	@When("^I access delete http method to delete resource from server$")
public void I_access_delete_http_method_to_delete_resource_from_server()
	{
		try
		{
			response=httpRequest.delete();
			String content=response.asString();
			System.out.println(content);
		}catch (Exception e)
		{
			e.printStackTrace();
		}
	}

	@Given("^I connect to the Mysql Database$")
	public void I_connect_to_the_Mysql_Database()
	{
		System.out.println("I connect to the Mysql Database!!!!!1");
	}

	@And("^I execute config database scripts$")
	public void I_execute_config_database_scripts()
	{
		System.out.println("I execute config database scripts!!!!!1");
	}
}
```

Driver script:

```
package com.sgtesting.tests.api.testsuite;


import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

import cucumber.api.CucumberOptions;
import cucumber.api.testng.CucumberFeatureWrapper;
import cucumber.api.testng.TestNGCucumberRunner;

@CucumberOptions(
            features = "src/test/resources/FeatureFiles",
            glue = "com.sgtesting.tests.api.stepdefinitions"
            )


public class DriverScript {
      private TestNGCucumberRunner testngCucumberRunner=null;

      @BeforeClass
      public void setUpMethod()
      {
            try
            {
                  testngCucumberRunner=new
TestNGCucumberRunner(this.getClass());
            }catch (Exception e) {
                  e.printStackTrace();
            }
      }

      @Test(dataProvider = "features")


      public void feature(CucumberFeatureWrapper cucumberFeature)
      {
            try
            {

      testngCucumberRunner.runCucumber(cucumberFeature.getCucumberFeature()
);
            }catch (Exception e) {
                  e.printStackTrace();
            }
      }

      @DataProvider
      public Object[][] features()
      {
            if(testngCucumberRunner==null)
            {
                  testngCucumberRunner=new
TestNGCucumberRunner(this.getClass());
            }
            return testngCucumberRunner.provideFeatures();
      }

      @AfterClass
```

```
	public void tearDownMethod()
	{
		try
		{
			testngCucumberRunner.finish();
		}catch (Exception e) {
			e.printStackTrace();
		}
	}


}
```

Feature file:

A_createuser.feature

```
# Feature Name:
# User Story Name:
# User Story Number:
# Author:
# Description:
Feature: Verify the create user restful api endpoint

Scenario: Create user by using restful api post endpoint
Given I specify the post restful api endpoint url
And I access given method to get RequestSpecification object
And I create user object using user pojo class
And I convert user object into user json object
And I provide header and body contents
When I access post http method to send request to server
Then I get status code as 201
Given I specify the get restful api endpoint url
And I access given method to get RequestSpecification object
When I access get http method to retrive resource from server
Then I validate content in response object
And I get status code as 200
Given I specify the delete restful api endpoint url
And I access given method to get RequestSpecification object
When I access delete http method to delete resource from server
Then I get status code as 200
```

B_createuser. Feature:

```
# Feature Name:
# User Story Name:
# User Story Number:
# Author:
# Description:
Feature: Verify the create user restful api endpoint New

Scenario: Create user by using restful api post endpoint
Given I specify the post restful api endpoint url
And I access given method to get RequestSpecification object
And I create user object using user pojo class
And I convert user object into user json object
And I provide header and body contents
When I access post http method to send request to server
Then I get status code as 201

Scenario: Display New Created User
Given I specify the get restful api endpoint url
And I access given method to get RequestSpecification object
When I access get http method to retrive resource from server
Then I validate content in response object
And I get status code as 200

Scenario: Delete New Created User
Given I specify the delete restful api endpoint url
And I access given method to get RequestSpecification object
When I access delete http method to delete resource from server
Then I get status code as 200
```

Pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
      <modelVersion>4.0.0</modelVersion>

      <groupId>com.sgtesting</groupId>
      <artifactId>rest_assured_prototype</artifactId>
      <version>0.0.1-SNAPSHOT</version>

      <name>rest_assured_prototype</name>
      <!-- FIXME change it to the project's website -->
      <url>http://www.example.com</url>

      <properties>
            <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
            <maven.compiler.source>1.7</maven.compiler.source>
            <maven.compiler.target>1.7</maven.compiler.target>
      </properties>

      <dependencies>
            <dependency>
                  <groupId>junit</groupId>
                  <artifactId>junit</artifactId>
                  <version>4.11</version>
```

```
			<scope>test</scope>
		</dependency>
		<dependency>
			<groupId>info.cukes</groupId>
			<artifactId>cucumber-java</artifactId>
			<version>1.2.5</version>
		</dependency>
		<dependency>
			<groupId>io.cucumber</groupId>
			<artifactId>cucumber-jvm-deps</artifactId>
			<version>1.0.6</version>
			<scope>provided</scope>
		</dependency>
		<dependency>
			<groupId>info.cukes</groupId>
			<artifactId>cucumber-jvm-deps</artifactId>
			<version>1.0.5</version>
			<scope>provided</scope>
		</dependency>
		<dependency>
			<groupId>info.cukes</groupId>
			<artifactId>cucumber-junit</artifactId>
			<version>1.2.5</version>
			<scope>test</scope>
		</dependency>
		<dependency>
			<groupId>info.cukes</groupId>
			<artifactId>cucumber-testng</artifactId>
			<version>1.2.5</version>
		</dependency>
		<dependency>
			<groupId>org.testng</groupId>
			<artifactId>testng</artifactId>
			<version>7.8.0</version>
			<scope>test</scope>
		</dependency>
		<dependency>
			<groupId>io.rest-assured</groupId>
			<artifactId>rest-assured</artifactId>
			<version>5.3.0</version>
			<scope>test</scope>
		</dependency>
		<dependency>
			<groupId>com.fasterxml.jackson.core</groupId>
			<artifactId>jackson-databind</artifactId>
			<version>2.15.2</version>
		</dependency>
	</dependencies>

	<build>
		<pluginManagement><!-- lock down plugins versions to avoid
using Maven
			defaults (may be moved to parent pom) -->
			<plugins>
				<!-- clean lifecycle, see
				https://maven.apache.org/ref/current/maven-
core/lifecycles.html#clean_Lifecycle -->
				<plugin>
					<artifactId>maven-clean-plugin</artifactId>
					<version>3.1.0</version>
				</plugin>
```

```
                        <!-- default lifecycle, jar packaging: see
                        https://maven.apache.org/ref/current/maven-
core/default-bindings.html#Plugin_bindings_for_jar_packaging -->
                        <plugin>
                              <artifactId>maven-resources-
                        plugin</artifactId>
                              <version>3.0.2</version>
                        </plugin>
                        <plugin>
                              <artifactId>maven-compiler-
                        plugin</artifactId>
                              <version>3.8.0</version>
                        </plugin>
                        <plugin>
                              <artifactId>maven-surefire-
                        plugin</artifactId>
                              <version>2.22.1</version>
                        </plugin>
                        <plugin>
                              <groupId>org.apache.maven.plugins</groupId>
                              <artifactId>maven-surefire-
                              plugin</artifactId>
                              <version>3.2.3</version>
                              <configuration>
                                    <suiteXmlFiles>

      <suiteXmlFile>testng.xml</suiteXmlFile>
                                    </suiteXmlFiles>
                              </configuration>
                        </plugin>
                        <plugin>
                              <artifactId>maven-jar-plugin</artifactId>
                              <version>3.0.2</version>
                        </plugin>
                        <plugin>
                              <artifactId>maven-install-plugin</artifactId>
                              <version>2.5.2</version>
                        </plugin>
                        <plugin>
                              <artifactId>maven-deploy-plugin</artifactId>
                              <version>2.8.2</version>
                        </plugin>
                        <!-- site lifecycle, see
                        https://maven.apache.org/ref/current/maven-
core/lifecycles.html#site_Lifecycle -->
                        <plugin>
                              <artifactId>maven-site-plugin</artifactId>
                              <version>3.7.1</version>
                        </plugin>
                        <plugin>
                              <artifactId>maven-project-info-reports-
plugin</artifactId>
                              <version>3.0.0</version>
                        </plugin>
                  </plugins>
            </pluginManagement>
      </build>
</project>
```

TestNG.xml:

```
<?xml version="1.0" encoding="UTF-8"?>

<suite name="API Testcases">
	<test name="testcases">
		<classes>
			<class
name="com.sgtesting.tests.api.testsuite.DriverScript"></class>
		</classes>
	</test>
</suite>
```

To download above project, go to (https://github.com/santoshubhale/RestAPI) this website