RestAssured

RestAssured is an API/Library through which we can automate RestAPI’s.

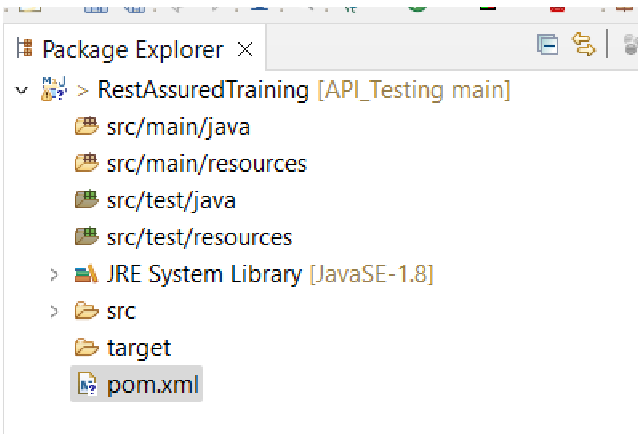
Pre-requisites:

1) Java 9+ & Eclipse
2) TestNG Framework
3) Maven (comes with Eclipse)

Install TestNG from Eclipse marketplace.

Specify all dependencies in pom.xml

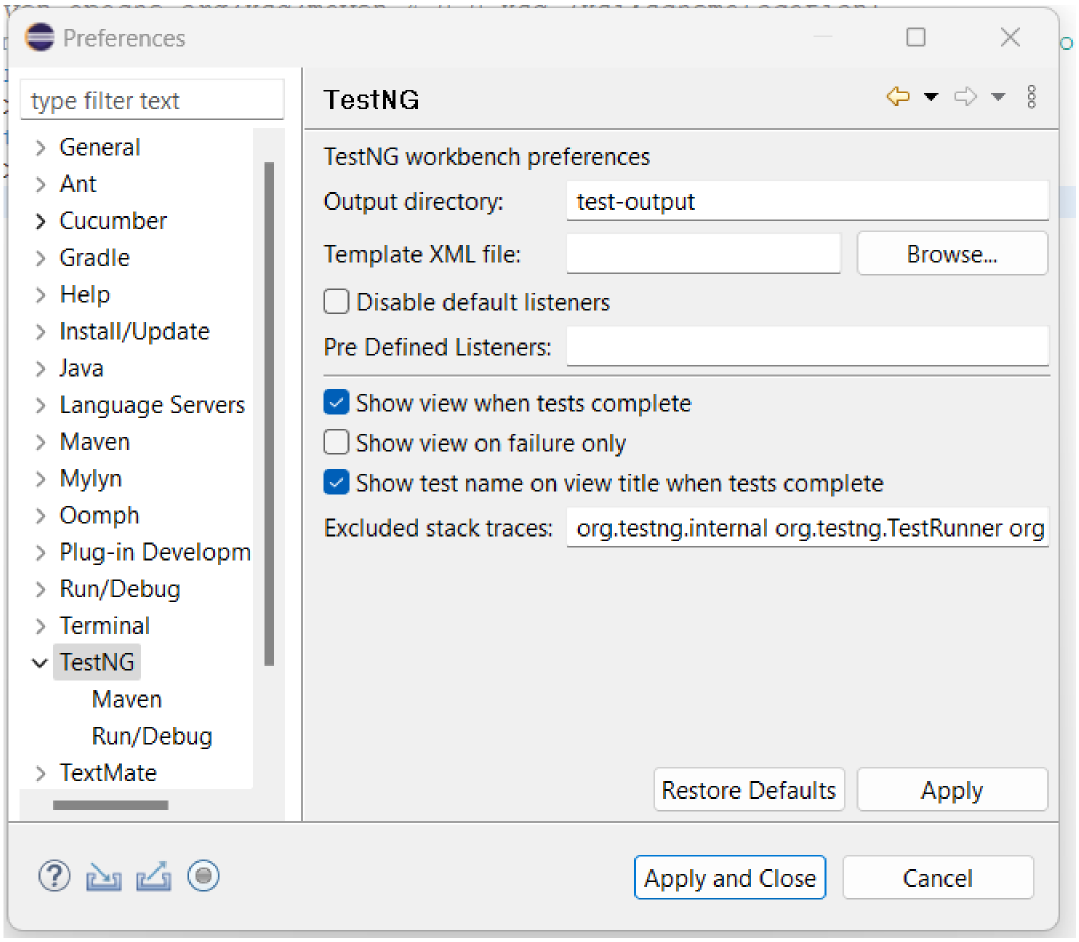
Create Maven Project.



Src/main/java & src/main/resources => used by developers

Src/test/java & src/test/resources => used by testers

Check TestNG is present:


Preferences

type filter text

- General
- Ant
- Cucumber
- Gradle
- Help
- Install/Update
- Java
- Language Servers
- Maven
- Mylyn
- Oomph
- Plug-in Developm
- Run/Debug
- Terminal
- TestNG
  - Maven
  - Run/Debug
- TextMate

TestNG

TestNG workbench preferences

Output directory: test-output

Template XML file: Browse...

[ ] Disable default listeners

Pre Defined Listeners:

[✓] Show view when tests complete

[ ] Show view on failure only

[✓] Show test name on view title when tests complete

Excluded stack traces: org.testng.internal org.testng.TestRunner org

Restore Defaults | Apply

Apply and Close | Cancel

Dependencies:

RestAssured: https://rest-assured.io/

Dependencies: https://github.com/rest-assured/rest-assured/wiki/GettingStarted

Add dependencies: Maven, Gradle, XmlPath, etc.

Dependencies to add in pom.xml:

1) io.rest-assured (io is package name) => rest-assured.
2) json-path
3) json
4) gson
5) testng
6) scribejava-apis
7) json-schema-validator
8) xml-schema-validator

Get dependencies from https://mvnrepository.com/.

1) io.rest-assured (io is package name) => rest-assured: https://mvnrepository.com/artifact/io.rest-assured/rest-assured/5.4.0.
2) json-path: https://mvnrepository.com/artifact/io.rest-assured/json-path/5.4.0

3) json: https://mvnrepository.com/artifact/org.json/json/20240303.
4) gson: https://mvnrepository.com/artifact/com.google.code.gson/gson/2.10.1.
5) testng: https://mvnrepository.com/artifact/org.testng/testng/7.9.0 .
6) scribejava-apis: https://mvnrepository.com/artifact/com.github.scribejava/scribejava-apis/8.3.1 .
7) json-schema-validator: https://mvnrepository.com/artifact/com.github.java-json-tools/json-schema-validator/2.2.14 .
8) xml-schema-validator:

HTTP Requests:

1) get
2) post
3) put
4) delete

Ex: https://reqres.in/

Gherkin -keywords

By default, restassured supports BDD Ghekin langusge so no need to add dependencies like cucumber

TestNG style

- given():
  Content type, set cookies, add auth i.e. authentication, add param i.e. parameters, set headers info i.e. information, etc.
- when():
  get, post, put, delete requests.
- then():
  validate status code, extract response, extract headers cookies & response body

Add static Packages: https://github.com/rest-assured/rest-assured/wiki/GettingStarted#static-imports

io.restassured.RestAssured.*

io.restassured.matcher.RestAssuredMatchers.*

org.hamcrest.Matchers.*

For which ever method starts first we don't need to specify dot '.' Else start with dot.

Ex: given()

____

________________.when()

.then()

Given ex:

GET List user: https://reqres.in/api/users?page=2 . Response code: 200

GET user: https://reqres.in/api/users/2 . Response code: 200

POST Create user: https://reqres.in/api/users. Response code: 201

```
{
  "name": "morpheus",
  "job": "leader"
}
```

PUT Update user: https://reqres.in/api/users/2 . Response code: 200

```
{
  "name": "morpheus",
  "job": "zion resident"
}
```

DELETE user: https://reqres.in/api/users/2 . Response code: 204

Static packages, ex:

```
package day1HTTPMatehods;

import org.testng.annotations.Test;

import static io.restassured.RestAssured.*;
import static io.restassured.matcher.RestAssuredMatchers.*;
import static org.hamcrest.Matchers.*;



/*
given():
 Content type, set cookies, add auth i.e. authentication, add param i.e. parameters, set headers info
i.e. information, etc.
when():
 get, post, put, delete requests.
then():
 validate status code, extract response, extract headers cookies & response body
*/

public class HTTPRequests {


	//using testng test
	//for single user


	@Test
	void getUser() {
		given()


		.when()
		 .get("https://reqres.in/api/users?page=2")


		.then()
		 .statusCode(200)
		 .body("page", equalTo(2))
		 .log().all();
	}




}
```

Example:

```
package day1HTTPMatehods;

import org.testng.annotations.Test;

import static io.restassured.RestAssured.*;
import static io.restassured.matcher.RestAssuredMatchers.*;
import static org.hamcrest.Matchers.*;

import java.util.HashMap;



/*
given():
 Content type, set cookies, add auth i.e. authentication, add param i.e. parameters, set headers info i.e. information, etc.
when():
 get, post, put, delete requests.
then():
 validate status code, extract response, extract headers cookies & response body
*/

public class HTTPRequests {



	int id;

	//using testng test
	//for single user
```

```
@Test(priority=1)
void getUser() {
        given()

        .when()
         .get("https://reqres.in/api/users?page=2")

        .then()
         .statusCode(200)
         .body("page", equalTo(2))
         .log().all();
}

@Test(priority=2)
void createUser() {
        HashMap data = new HashMap(); //hashmap is in key value pair
        data.put("name", "rutuja");
        data.put("job", "sdet");

        id = given()
           .contentType("application/json")
           .body(data)

        .when()
           .post("https://reqres.in/api/users")
           .jsonPath().getInt("id");

/*
        .then()
```

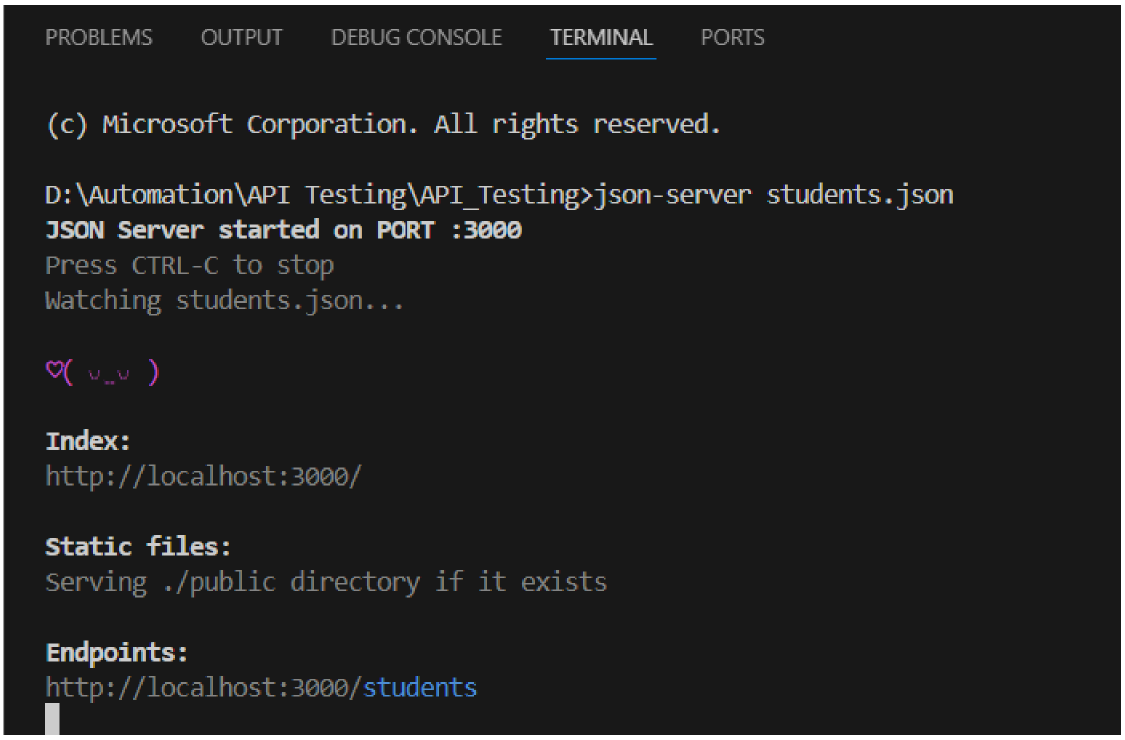
```
                    .statusCode(201)
                    .log().all();
*/

        }

        @Test(priority=3,dependsOnMethods = {"createUser"})
        void updateUser() {
                HashMap data = new HashMap();
                data.put("name", "John");
                data.put("job", "SDE");

                given()
                        .contentType("application/json")
                        .body(data)
                .when()
                  .put("https://reqres.in/api/users/"+id)
                .then()
                  .statusCode(200)
                  .log().all();



        }

        @Test(priority=4)
        void deleteUser() {
                given()
                .when()
                  .delete("https://reqres.in/api/users/"+id)
                .then()
                  .statusCode(204)
                  .log().all();
        }




}
```

Creating Post Request Payloads in Multiple Ways

How many ways we create request body:

1) Hashmap
2) Using org.json
3) Using POJO (Plain Old Java Object)
4) Using external json file

Start local server



```
(c) Microsoft Corporation. All rights reserved.

D:\Automation\API Testing\API_Testing>json-server students.json
JSON Server started on PORT :3000
Press CTRL-C to stop
Watching students.json...

♡( ◡‿◡ )

Index:
http://localhost:3000/

Static files:
Serving ./public directory if it exists

Endpoints:
http://localhost:3000/students
```

Different ways to create POST request body

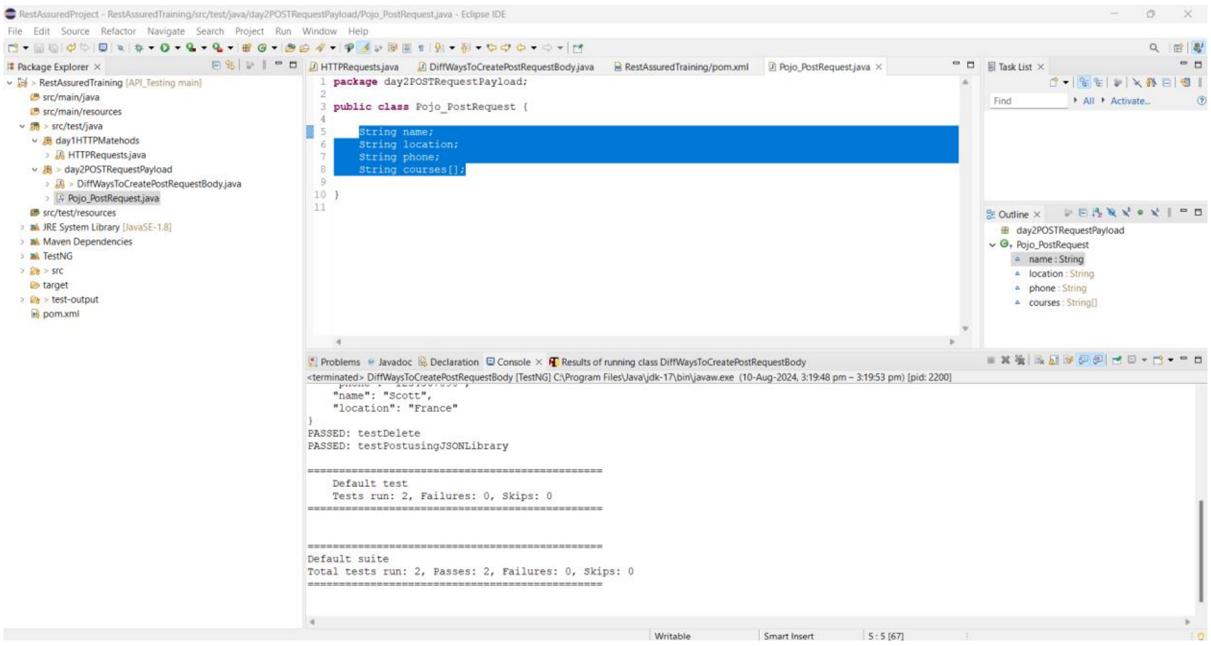
1) Post request body using Hashmap.

2) Post request body creating using Org.JSON

3) Post request body creating using POJO class

4) Post request using external json file data

POJO => Encapsulation followed here.


RestAssuredProject - RestAssuredTraining/src/test/java/day2POSTRequestPayload/Pojo_PostRequest.java - Eclipse IDE

```java
package day2POSTRequestPayload;

public class Pojo_PostRequest {

    String name;
    String location;
    String phone;
    String courses[];

}
```

```
<terminated> DiffWaysToCreatePostRequestBody [TestNG] C:\Program Files\Java\jdk-17\bin\javaw.exe (10-Aug-2024, 3:19:48 pm – 3:19:53 pm) [pid: 2200]
    "name": "Scott",
    "location": "France"
}
PASSED: testDelete
PASSED: testPostusingJSONLibrary

===============================================
    Default test
    Tests run: 2, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 2, Passes: 2, Failures: 0, Skips: 0
===============================================
```

Source -> Generate Getters and Setters

EX:

```java
package day2POSTRequestPayload;



import static io.restassured.RestAssured.given;

import static org.hamcrest.Matchers.equalTo;


import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.util.HashMap;


import org.json.JSONObject;

import org.json.JSONTokener;


/*

Different ways to create POST request body

1) Post request body using Hashmap.

2) Post request body creating using Org.JSON
```

```
3) Post request body creating using POJO class
4) Post request using external json file data
*/


import org.testng.annotations.Test;


public class DiffWaysToCreatePostRequestBody {


        //1) POST request body using Hashmap
/*
        //create method
        @Test(priority=1)
        void testPostusingHashmap() {
                HashMap data = new HashMap();


                data.put("name", "Scott");
                data.put("location", "France");
                data.put("phone", "1234567890");


                String courseArr[] = {"C", "C++"};


                data.put("courses", courseArr);


                given()
                  .contentType("application/json")
                  .body(data)
                .when()
                  .post("http://localhost:3000/students")
                .then()
                  .statusCode(201)
```

```
                    .body("name", equalTo("Scott"))
                    .body("location", equalTo("France"))
                    .body("phone", equalTo("1234567890"))
                    .body("courses[0]", equalTo("C"))
                    .body("courses[1]", equalTo("C++"))
                    //verify header
                    .header("Content-Type", "application/json; charset=utf-8")
                    //print entire response body
                    .log().all();

        }
*/



        //2) Post request body creating using Org.JSON
/*
                //create method
                @Test(priority=1)
                void testPostusingJSONLibrary() {
                        JSONObject data = new JSONObject();

                        data.put("name", "Scott");
                        data.put("location", "France");
                        data.put("phone", "1234567890");

                        String courseArr[] = {"C", "C++"};

                        data.put("courses", courseArr);

                        given()
                          .contentType("application/json")
                        .body(data.toString())
                      .when()
                        .post("http://localhost:3000/students")
                      .then()
                        .statusCode(201)
                        .body("name", equalTo("Scott"))
                        .body("location", equalTo("France"))
                        .body("phone", equalTo("1234567890"))
                        .body("courses[0]", equalTo("C"))
                        .body("courses[1]", equalTo("C++"))
                        //verify header
                        .header("Content-Type", "application/json") //;charset=utf-8")
                        //print entire response body
                        .log().all();


              }
*/


              //3) Post request body creating using POJO class
       /*
                      //create method
                      @Test(priority=1)
                      void testPostusingPOJO() {
                              Pojo_PostRequest data = new Pojo_PostRequest();
                              //Pojo_PostRequest data = new Pojo_PostRequest();



                              data.setName("Scott");
                              data.setLocation("France");
                              data.setPhone("1234567890");
```

```
				String courseArr[] = {"C", "C++"};

				data.setCourses(courseArr);

				given()
				  .contentType("application/json")
				  .body(data)
				.when()
				  .post("http://localhost:3000/students")
				.then()
				  .statusCode(201)
				  .body("name", equalTo("Scott"))
				  .body("location", equalTo("France"))
				  .body("phone", equalTo("1234567890"))
				  .body("courses[0]", equalTo("C"))
				  .body("courses[1]", equalTo("C++"))
				  //verify header
				  .header("Content-Type", "application/json") //;charset=utf-8")
				  //print entire response body
				  .log().all();

			}
*/

//4) Post request using external json file data

//create method
@Test(priority=1)
void testPostusingExternalJSONFile() throws FileNotFoundException {
	File f = new File(".\\body.json");
```

```
        FileReader fr = new FileReader(f);

        JSONTokener jt = new JSONTokener(fr);

        JSONObject data = new JSONObject(jt);

        given()
          .contentType("application/json")
          .body(data.toString())
        .when()
          .post("http://localhost:3000/students")
        .then()
          .statusCode(201)
          .body("name", equalTo("Scott"))
          .body("location", equalTo("France"))
          .body("phone", equalTo("1234567890"))
          .body("courses[0]", equalTo("C"))
          .body("courses[1]", equalTo("C++"))
          //verify header
          .header("Content-Type", "application/json") //;charset=utf-8")
          //print entire response body
          .log().all();

    }

    //Deleting student record
    @Test(priority=2)
    void testDelete() {
```

```
                given()
                .when()
                   .delete("http://localhost:3000/students/2b50")
                .then()
                   .statusCode(200);
        }


}
```

Cookies & Headers | Query & Path Parameters

1) Path & Query Parameters

   https://reqres.in/api/users?page=2

   https://reqres.in/api/users?page=2&id=5

   here, ‘user’ = path parameter; ‘page’ & ‘id’ = query parameter.

   Anything after ‘?’ is query parameter.

2) Cookies & Headers
   Cookies:

```
package day3CookiesHeadersQueryPathParameters;


import org.testng.annotations.Test;

import io.restassured.response.Response;

import static io.restassured.RestAssured.*;
import static io.restassured.matcher.RestAssuredMatchers.*;
import static org.hamcrest.Matchers.*;

import java.util.HashMap;
import java.util.Map;


public class CookiesDemo {

      @Test(priority=1)

      void testCookies() {
            given()
            .when()
               .get("https://www.google.com")
            .then()
                .cookie("AEC",
"AVYB7cq7fYZfxDlK1mGmZR2OojvMI79YlIzN0ZMkJHQ7XMolrXm4WN6IWQ")
               .log().all();
```

```
	}

	@Test(priority=2)

	void getCookiesInfo() {
		Response res = given()
		.when()
		   .get("https://www.google.com");

		//get single cookie info
		String cookie_value=res.getCookie("AEC");
		System.out.println("Value of cookie is ====>"+cookie_value);

		//get all cookies info
		Map<String,String> cookies_values = res.getCookies();

		//System.out.println(cookies_values.keySet());

		for(String k:cookies_values.keySet()) {

			String cookie_valuee = res.getCookie(k);
			System.out.println(k+" "+cookie_valuee);

		}
	}

}
```

Headers:
Header name: value

Headers:
Header name: value
Header name: value
Header name: value

Ex: `package day3CookiesHeadersQueryPathParameters;`

```
import org.testng.annotations.Test;

import io.restassured.http.Header;
import io.restassured.http.Headers;
import io.restassured.response.Response;

import static io.restassured.RestAssured.*;
import static io.restassured.matcher.RestAssuredMatchers.*;
import static org.hamcrest.Matchers.*;

import java.util.HashMap;

public class HeadersDemo {

	@Test(priority=1)

	void testHeaders() {
		given()
		.when()
		   .get("https://www.google.com")
		.then()
		   .header("Content-Type", "text/html; charset=ISO-8859-1")
		   .and()
		   .header("Content-Encoding", "gzip")
		   .and()
		   .header("Server", "gws")
		   .log().headers(); //print only headers in the response
	}

	@Test(priority=2)

	void getHeaders() {
		//Response res = (Response) given()
		Response res = given()
					    .when()
					      .get("https://www.google.com/");

					      //get single header info
					      String headervalue = res.getHeader("Content-
Type");
				      System.out.println("The value of Content-type header
is: "+headervalue);


				      //get all headers info
				      Headers myheaders = res.getHeaders();

				      for(Header hd:myheaders) {
					      System.out.println(hd.getName()+ "
"+hd.getValue());

				      }
	}

}
```

Looging:

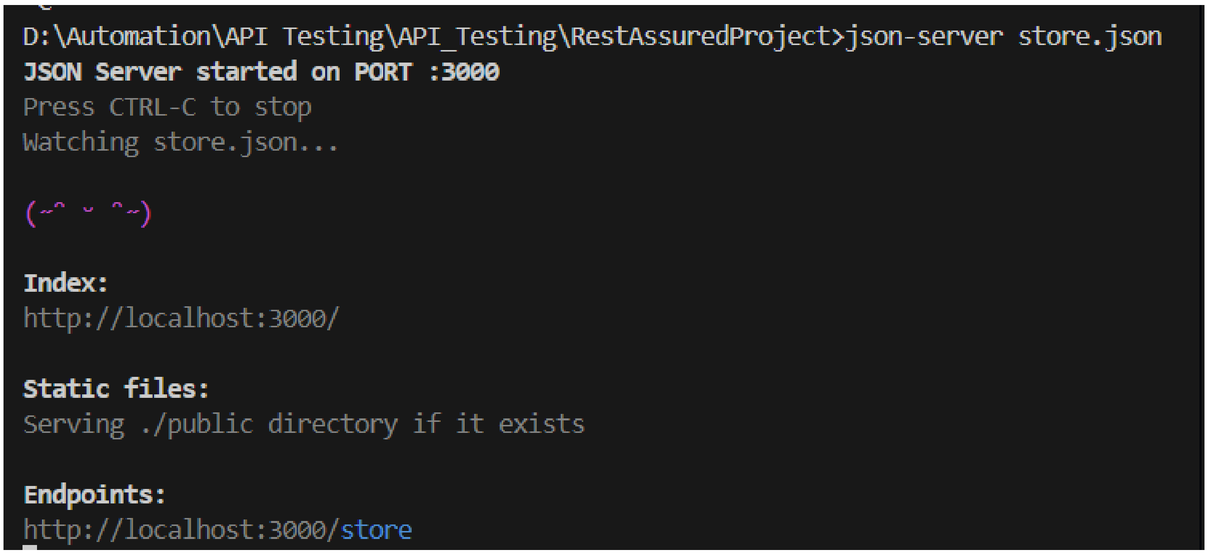
```
package day3CookiesHeadersQueryPathParameters;


import static io.restassured.RestAssured.*;
import static io.restassured.matcher.RestAssuredMatchers.*;
import static org.hamcrest.Matchers.*;

import org.testng.annotations.Test;


public class LoggingDemo {

	@Test(priority=1)

	void testLogs() {
		given()
		.when()
		  .get("https://reqres.in/api/users?page=2")
```

```
        .then()
          //.log().body()
          //.log().cookies() //to print only cookies from the response
          //.log().headers() //to print only headers in response
          .log().all(); //to print all the logs in response
    }

}
```

Parsing Response Body | JSONObject

Parsing: means we can traverse through json response to get required field.

Run store.json file locally on json server.



```
D:\Automation\API Testing\API_Testing\RestAssuredProject>json-server store.json
JSON Server started on PORT :3000
Press CTRL-C to stop
Watching store.json...

(˶ᵔ ᵕ ᵔ˶)

Index:
http://localhost:3000/

Static files:
Serving ./public directory if it exists

Endpoints:
http://localhost:3000/store
```

Ex:

```
package day4ParsingResponseBodyJSONObject;


import static io.restassured.RestAssured.given;

import org.json.JSONObject;
import org.testng.Assert;
import org.testng.annotations.Test;

import io.restassured.http.ContentType;
import io.restassured.response.Response;


public class ParsingJSONResponseData {

    @Test(priority=1)

    void testJSONResponse() {

        //Approach 1
    /*
        given()
          .contentType("ContentType.JSON")
        .when()
          .get("http://localhost:3000/store")
```

```
            .then()
              .statusCode(200)
              .header("Content-Type", "application/json") //; charset=utf-
8")
              //find JSON Path from https://jsonpathfinder.com/
              .body("book[3].title", equalTo("The Lord of the Rings"))
              ;
      */
            //Approach 2 => Capture entire response in one variable

            Response res = given()
              //.contentType("ContentType.JSON")
                        .contentType(ContentType.JSON)
            .when()
              .get("http://localhost:3000/store");

            //res.getStatusCode()
            Assert.assertEquals(res.getStatusCode(), 200) ; //validation 1
            Assert.assertEquals(res.header("Content-Type"),
"application/json");

            String bookname =
res.jsonPath().get("book[3].title").toString();
            Assert.assertEquals(bookname, "The Lord of the Rings");

      }


      @Test(priority=2)

      void testJSONResponsebodyData() {


            Response res =
            given()
              .contentType(ContentType.JSON)
            .when()
              .get("http://localhost:3000/store");
      /*
            //res.getStatusCode()
            Assert.assertEquals(res.getStatusCode(), 200) ; //validation 1
            Assert.assertEquals(res.header("Content-Type"),
"application/json");

            String bookname =
res.jsonPath().get("book[3].title").toString();
            Assert.assertEquals(bookname, "The Lord of the Rings");
   */

            //JSONObject class
            JSONObject jo = new JSONObject(res.toString()); //converting
response to json object type
      /*
            for(int i=0; i<jo.getJSONArray("book").length(); i++) {
                  String bookTitle =
jo.getJSONArray("book").getJSONObject(i).get("title").toString();
                  System.out.println(bookTitle);
      */

            //search for title of book in json - validation 1
            boolean status=false;
            for(int i=0; i<jo.getJSONArray("book").length(); i++) {
                String bookTitle =
jo.getJSONArray("book").getJSONObject(i).get("title").toString();
                if(bookTitle.equals("The Lord of the Rings"))
                {
                    status=true;
                    break;
                }


            }

            Assert.assertEquals(status, true);
        /*
            if(status==false) {
                //not found

            }
        */


            //validating total price of books - validation 2
            double totalprice=0;
            for(int i=0; i<jo.getJSONArray("book").length(); i++) {
                String price =
jo.getJSONArray("book").getJSONObject(i).get("price").toString()   ;

                totalprice = totalprice + Double.parseDouble(price);


            }

            System.out.println("total price of book is:"+totalprice);
            Assert.assertEquals(totalprice, 53.92);
        }

}
```

Parsing XML Response Body | File Upload & Download API's

Parsing XML

Ex:
```
package day5ParsingXMLResponseBodyFileUploadDownloadAPI;


import org.testng.Assert;
import org.testng.annotations.Test;

import io.restassured.path.xml.XmlPath;
import io.restassured.response.Response;

import static io.restassured.RestAssured.*;
import static io.restassured.matcher.RestAssuredMatchers.*;
import static org.hamcrest.Matchers.*;

import java.util.List;

public class ParsingXMLResponse {
      @Test(priority=1)
      void testXMLResponse() {
            //Approach 1
      /*
            given()
            .when()
               .get("http://restapi.adequateshop.com/api/Traveler?page=1")
            .then()
               .statusCode(200)
             // .header("Content-Type", "text/html; charset=us-ascii")
               .header("Content-Type", "application/xml")
               .body("TravelerinformationResponse.page", equalTo("1"))

.body("TravelerinformationResponse.travelers.Travelerinformation[0].name",
equalTo("Vijay Bharat Reddy"))

               ;
      */

            //Aproach 2

            Response res = given()
                        .when()

.get("http://restapi.adequateshop.com/api/Traveler?page=1");
            Assert.assertEquals(res.getStatusCode(), 200 );
            Assert.assertEquals(res.header("Content-Type"),
"application/xml");
            String pageNo =
res.xmlPath().get("TravelerinformationResponse.page").toString();
            Assert.assertEquals(pageNo, "1");

            String travelName =
res.xmlPath().getString("TravelerinformationResponse.travelers.Travelerinfo
rmation[0].name").toString();
            Assert.assertEquals(travelName, "Vijay Bharat Reddy");
      }

      @Test(priority=2)

      void testXMLResponseBody() {


            Response res = given()
                        .when()

.get("http://restapi.adequateshop.com/api/Traveler?page=1");
            XmlPath xmlobj = new XmlPath(res.asString());


            //Verify total number of travelers
            List<String> travelers =
xmlobj.getList("TravelerinformationResponse.travelers.Travelerinformation")
;
            Assert.assertEquals(travelers.size(), 10);

            //Verify traveler name is present in response
            List<String> traveler_names =
xmlobj.getList("TravelerinformationResponse.travellers.Travelerinformation.
name");
```

```
		boolean status=false;
		for(String travelername:traveler_names) {
			//System.out.println(travelername);
			if(travelername.equals("Vijay Bharat Reddy")) {
				status=true;
				break;
			}
		}


		Assert.assertEquals(status, true);

		;
	}


}
```

File Upload:

Create file-upload-RestAPI.jar file.

Run it locally using command “java -jar file-upload-RestAPI.jar”

```
package day5ParsingXMLResponseBodyFileUploadDownloadAPI;


import static io.restassured.RestAssured.*;
import static io.restassured.matcher.RestAssuredMatchers.*;
import static org.hamcrest.Matchers.*;

import java.io.File;

import org.testng.annotations.Test;


public class FileUploadAndDownload {

	@Test(priority=1)

	void singleFileUpload() {
		 File myfile = new File("D:\\Automation\\API
Testing\\API_Testing\\RestAssuredProject\\RestAssuredTraining\\Text1.txt");

		given()
		  .multiPart("file",myfile)
		  .contentType("multipart/form-data")
		.when()
		   .post("https://localhost:8080/uploadFile")
		.then()
		   .statusCode(200)
		   .body("fileName", equalTo("Test1.txt"))
		   .log().all();

	}
```

```java
	@Test(priority=3)

	void multipleFileUpload() {
		File myfile1 = new File("D:\\Automation\\API 
Testing\\API_Testing\\RestAssuredProject\\RestAssuredTraining\\Text1.txt");
		File myfile2 = new File("D:\\Automation\\API 
Testing\\API_Testing\\RestAssuredProject\\RestAssuredTraining\\Text2.txt");

		given()
		  .multiPart("files",myfile1)
		  .multiPart("files",myfile2)
		  .contentType("multipart/form-data")
		.when()
		   .post("https://localhost:8080/uploadMultipleFiles")
		.then()
		   .statusCode(200)
		   .body("[0].fileName", equalTo("Test1.txt"))
		   .body("[1].fileName", equalTo("Test2.txt"))
		   .log().all();

	}


	//Approach - using single array (won't work for all kinds of api)
	@Test(priority=4)

	void multipleFileUpload2() { //(won't work for all kinds of api)
		File myfile1 = new File("D:\\Automation\\API 
Testing\\API_Testing\\RestAssuredProject\\RestAssuredTraining\\Text1.txt");
		File myfile2 = new File("D:\\Automation\\API 
Testing\\API_Testing\\RestAssuredProject\\RestAssuredTraining\\Text2.txt");



		File filearr[] = {myfile1, myfile2};

		 given()
		  .multiPart("files",filearr)
		  .contentType("multipart/form-data")
		.when()
		   .post("https://localhost:8080/uploadMultipleFiles")
		.then()
		   .statusCode(200)
		   .body("[0].fileName", equalTo("Test1.txt"))
		   .body("[1].fileName", equalTo("Test2.txt"))
		   .log().all();

	}

	// File Download
	@Test(priority=2)

	void fileDownload() {
		given()
		.when()
		  //.get("http://localhost:8080/downloadFile/Test1.txt")
		  .get("http://localhost:8080/downloadFile/Test2.txt")
		.then()
		  .statusCode(200)
		  .log().body();
```

```
      }

}
```

JSON & XML Schema validations | Serial & De-serilisation

1) Response validation: data
2) Schema Validation: type of data

Convert JSON to JSON Schema and pass it.

Run json server locally using command: ‘json-server store.json’.

Ex:

```
package day6JSONXMLSchemaValidationsSerialDEserilisation;


//import io.restassured.module.jsv.JsonSchemaValidator;

import static io.restassured.RestAssured.*;
import static io.restassured.matcher.RestAssuredMatchers.*;
import static org.hamcrest.Matchers.*;

import org.testng.annotations.Test;

//json -> jsonschema converter
// https://www.liquid-technologies.com/online-json-to-schema-converter


public class JSONSchemaValidation {

      @Test(priority=1)

      void JSONschemavalidation() {

            given()
            .when()
              .get("http://localhost:3000/store")
            .then()
//
.assertThat().body(JsonSchemaValidator.matchesJsonSchemaInClasspath("storeJ
SONSchema.json"));
             //
.assertThat().body(JsonSchemaValidator.matchesJsonSchemaInClasspath("storeJ
SONSchema.json"));
            ;
      }

}
```

XML Schema validation is not possible in Postman. Possible in RestAssured.

1) Json Response (.json) -> Json Schema (.json)
2) XML Response (.xml) -> Xml Schema (.xsd)

For XML Schema -> first convert JSON to XML format. Then convert XML format to XSD format.

https://www.site24x7.com/tools/json-to-xml.html.

https://www.liquid-technologies.com/online-xml-to-xsd-converter.

Ex:
```
package day6JSONXMLSchemaValidationsSerialDEserilisation;


import static io.restassured.RestAssured.*;
import static io.restassured.matcher.RestAssuredMatchers.*;
import static org.hamcrest.Matchers.*;

import org.testng.annotations.Test;

import io.restassured.matcher.RestAssuredMatchers;


public class xmlSchemaValidation {

	@Test

	void xmlSchemavalidation() {

		given()
		.when()
		  .get("http://restapi.adequateshop.com/api/Traveler")
		.then()

.assertThat().body(RestAssuredMatchers.matchesXsd("traveler.xsd"));
		;

	}

}
```

1) Serialization: pojo --→ json
2) De-serilization: json --→ pojo

Body(json) ---→ Request --→ Response(json)

We can create response in following forms:

1) POJO
2) Hashmap
3) JSON
4) gson

Which class is used/how to convert pojo to json format?

We need to import 'Jackson' package and ObjMapper class is used.

Ex: import com.fasterxml.jackson.core.JsonProcessingException;

import com.fasterxml.jackson.databind.ObjectMapper;

&

ObjectMapper objMapper = new ObjectMapper();

Student_POJO stupojo = objMapper.readValue(jsondata, Student_POJO.class); //convert json to pojo

Ex:
```
package day6JSONXMLSchemaValidationsSerialDEserilisation;

import org.testng.annotations.Test;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;

//Pojo --- Serilize ---> JSON Object --- de-serilize ---> Pojo

public class SerilizationDeserilization {

	//POJO -----------> JSON (Serilization)
	@Test

	void convertPojo2Json() throws JsonProcessingException {

		//created java object using pojo class
	//	Student stupojo = new Student(); //pojo


		Student_POJO stupojo=new Student_POJO();

		stupojo.setName("Scott");
		stupojo.setLocation("France");
		stupojo.setPhone("1234567890");
		String courseArr[] = {"C", "C++"};
		stupojo.setCourses(courseArr);

		//convert java object -> json object (serilization)
		ObjectMapper objMapper = new ObjectMapper();

		String jsondata = 
objMapper.writerWithDefaultPrettyPrinter().writeValueAsString(stupojo);

		System.out.println(jsondata);

	}

	//JSON -----------> POJO (De-Serilization)
```

```
            @Test

            void convertJson2Pojo() throws JsonProcessingException {

                  String jsondata = "{\r\n"
                              + "   \"name\" : \"Scott\",\r\n"
                              + "   \"location\" : \"France\",\r\n"
                              + "   \"phone\" : \"1234567890\",\r\n"
                              + "   \"courses\" : [ \"C\", \"C++\" ]\r\n"
                              + "}";
                  //convert json data ---> Pojo object

                  ObjectMapper objMapper = new ObjectMapper();

                  Student_POJO stupojo = objMapper.readValue(jsondata,
Student_POJO.class); //convert json to pojo

                  System.out.println("Name: "+stupojo.getName());
                  System.out.println("Location: "+stupojo.getLocation());
                  System.out.println("Phone: "+stupojo.getPhone());
                  System.out.println("Course 1: "+stupojo.getCourses()[0]);
                  System.out.println("Course 2: "+stupojo.getCourses()[1]);
            }

}
```

Types of Authorizations | Faker Library

Authorizations

1) Authentication: user credentials valid or not.
2) Authorization: user access permission.

Authorization is only valid for Authenticated user i.e. first privacy primary security parameter is Authentication.

Kinds of Authentication in RestAssured:

1) Basic

2) Digest

3) Preemptive

4) Bearer Token

5) oauth 1.0, 2.0

6) API Key

OAuth 1.0 Authentication:

Syntax:

```
@Test(priority=5)

      void testOAuth1Authentication() {
            given()
              .auth().oauth("consumerKey", "consumerSecret", "accessToken",
"tokenSecrate ") //this is for OAuth 1.0 authentication
            .when()
              .get("url")
            .then()
              .statusCode(200)
              .log().all();
      }
```

OAuth 2.0 Authentication:

Syntax:

```
@Test(priority=6)

      void testOAuth2Authentication() {
            given()
              .auth().oauth2("ghp_24pH....") //access token
            .when()
               .get("https://api.github.com/user/repos")
            .then()
              .statusCode(200)
              .log().all();
      }
```

Generate API Key on: https://openweathermap.org/forecast16

Faker library:

https://github.com/DiUS/java-faker

Add in pom.xml

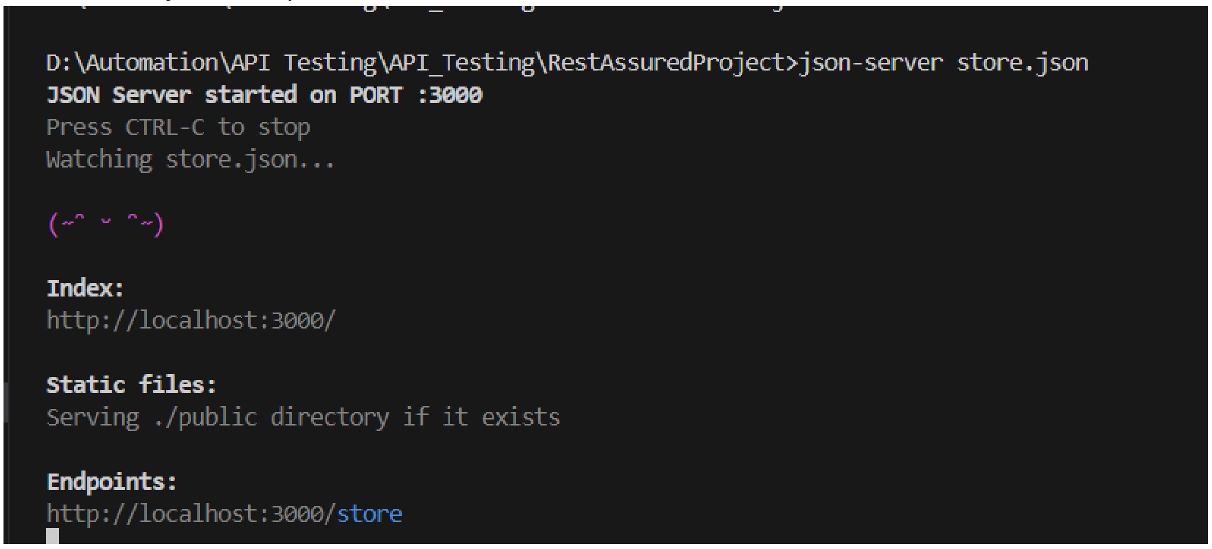
```
<dependency>
  <groupId>com.github.javafaker</groupId>
  <artifactId>javafaker</artifactId>
  <version>1.0.2</version>
</dependency>
```

Faker will generate random data.

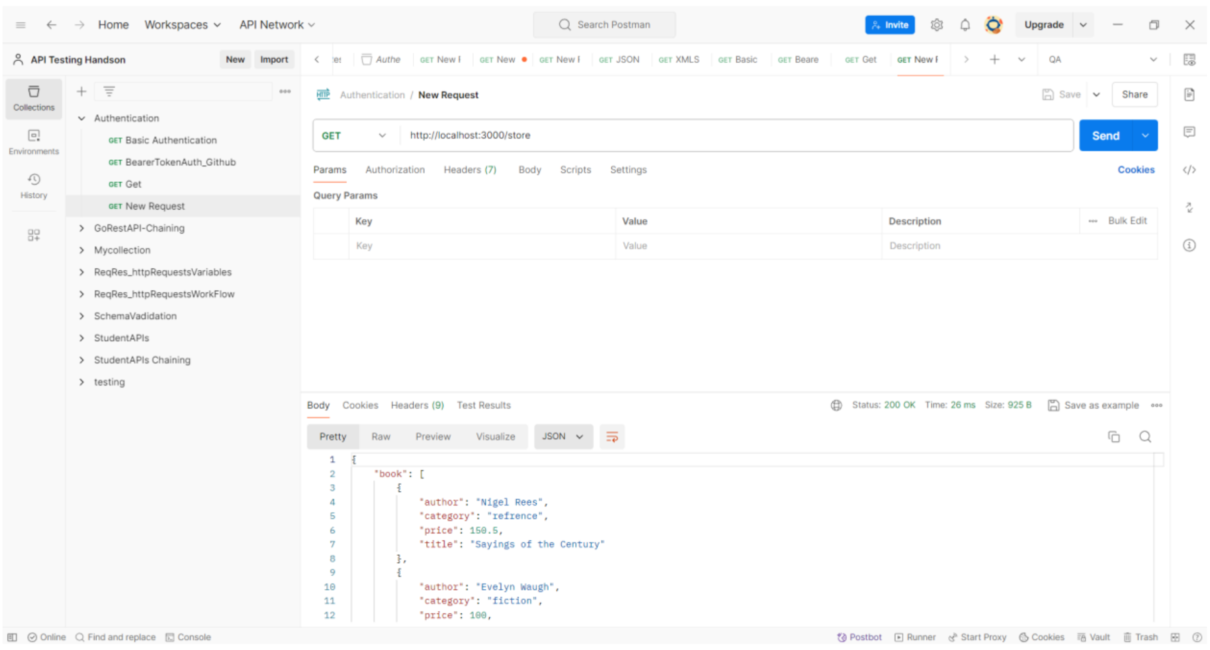
Ex:

```
package day7TypesOfAuthorizationsFakerLibrary;

import org.testng.annotations.Test;

import com.github.javafaker.Faker;


public class FakerDataGenerator {
	@Test

	void testGenerateDummyData() {
		Faker faker = new Faker();

		String fullname = faker.name().fullName();
		String firstname = faker.name().firstName();
		String lastname = faker.name().lastName();

		String username = faker.name().username();
		String password = faker.internet().password();

		String phoneno = faker.phoneNumber().cellPhone();

		String email = faker.internet().safeEmailAddress();

		System.out.println("Full Name:  "+fullname);
		System.out.println("First Name:  "+firstname);
		System.out.println("Last name:  "+lastname);
		System.out.println("Username:  "+username);
		System.out.println("Password:  "+password);
		System.out.println("Phone No.:  "+phoneno);
		System.out.println("Email:  "+email);
	}

}
```

1) JSON Object:
   {
   }
2) JSON Array:
   [
   ]
3) JSON Element: combination of json object and array
   Ex: Run store.json locally



```
D:\Automation\API Testing\API_Testing\RestAssuredProject>json-server store.json
JSON Server started on PORT :3000
Press CTRL-C to stop
Watching store.json...

(˶ᵔ ᵕ ᵔ˶)

Index:
http://localhost:3000/

Static files:
Serving ./public directory if it exists

Endpoints:
http://localhost:3000/store
```


Authentication / New Request

GET http://localhost:3000/store

Status: 200 OK Time: 26 ms Size: 925 B

```
{
    "book": [
        {
            "author": "Nigel Rees",
            "category": "refrence",
            "price": 150.5,
            "title": "Sayings of the Century"
        },
        {
            "author": "Evelyn Waugh",
            "category": "fiction",
            "price": 100,
```

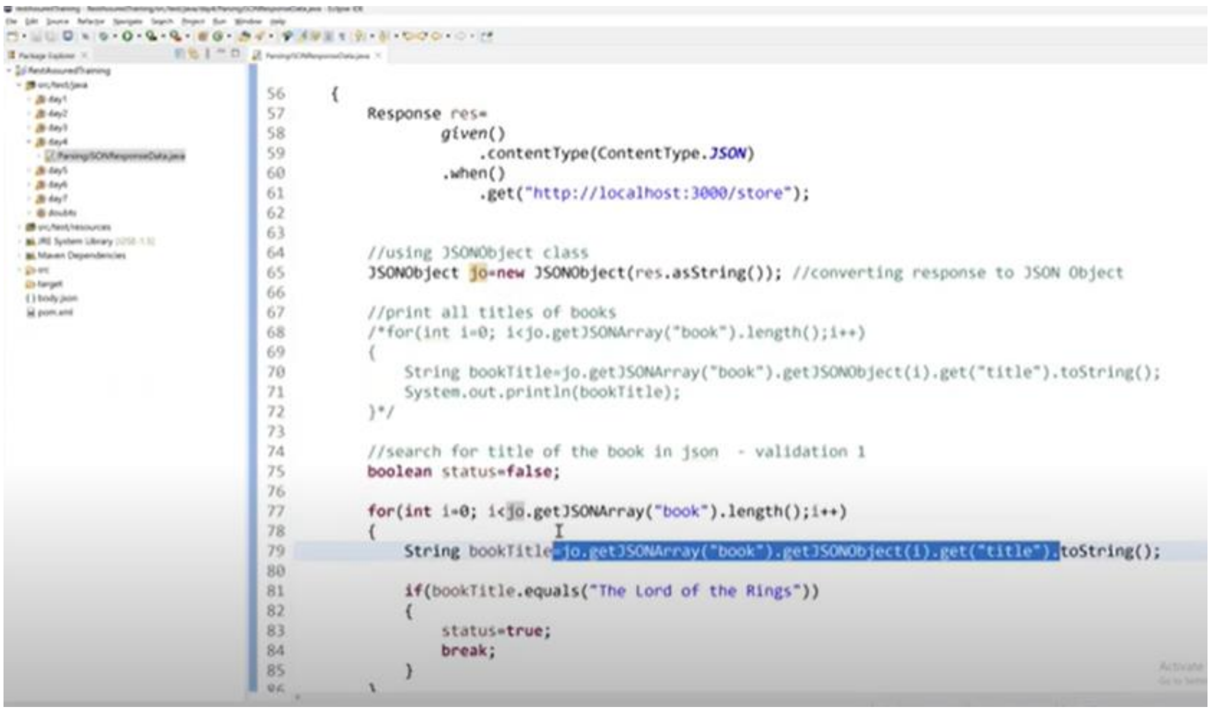
```
{
  "book": [
    {
      "author": "Nigel Rees",
      "category": "refrence",
      "price": 150.5,
      "title": "Sayings of the Century"
    },
    {
      "author": "Evelyn Waugh",
      "category": "fiction",
      "price": 100,
      "title": "Sword of Honour"
    },
    {
      "author": "Herman Melville",
      "category": "fiction",
      "isbn": "0-553-21311-3",
      "price": 75.5,
      "title": "Moby Dick"
    },
    {
      "author": "J.R.R. Tolkien",
      "category": "fiction",
      "isbn": "0-395-19395-8",
      "price": 200,
      "title": "The Lord of the Rings"
    }
  ]
}
```

JSON Object -> JSON Array -> JSON Object

JSONObject jo = new JSONObject(res.toString());

jo.getJSONArray(“book”).getJSONObject(i).get(“title”)

jo.getJsONArray(“book”).getJSONObject(3).get(“author”)



```java
	{
		Response res=
				given()
					.contentType(ContentType.JSON)
				.when()
					.get("http://localhost:3000/store");


		//using JSONObject class
		JSONObject jo=new JSONObject(res.asString()); //converting response to JSON Object

		//print all titles of books
		/*for(int i=0; i<jo.getJSONArray("book").length();i++)
		{
			String bookTitle=jo.getJSONArray("book").getJSONObject(i).get("title").toString();
			System.out.println(bookTitle);
		}*/

		//search for title of the book in json  - validation 1
		boolean status=false;

		for(int i=0; i<jo.getJSONArray("book").length();i++)
		{
			String bookTitle=jo.getJSONArray("book").getJSONObject(i).get("title").toString();

			if(bookTitle.equals("The Lord of the Rings"))
			{
				status=true;
				break;
			}
		}
```

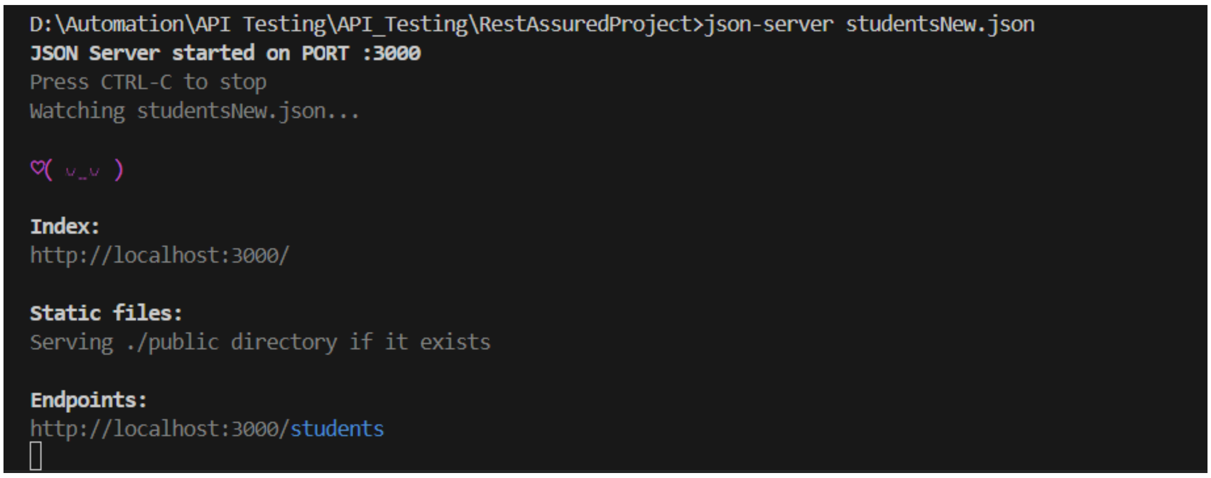
Ex: employee.json

JSON Array -> JSON Object

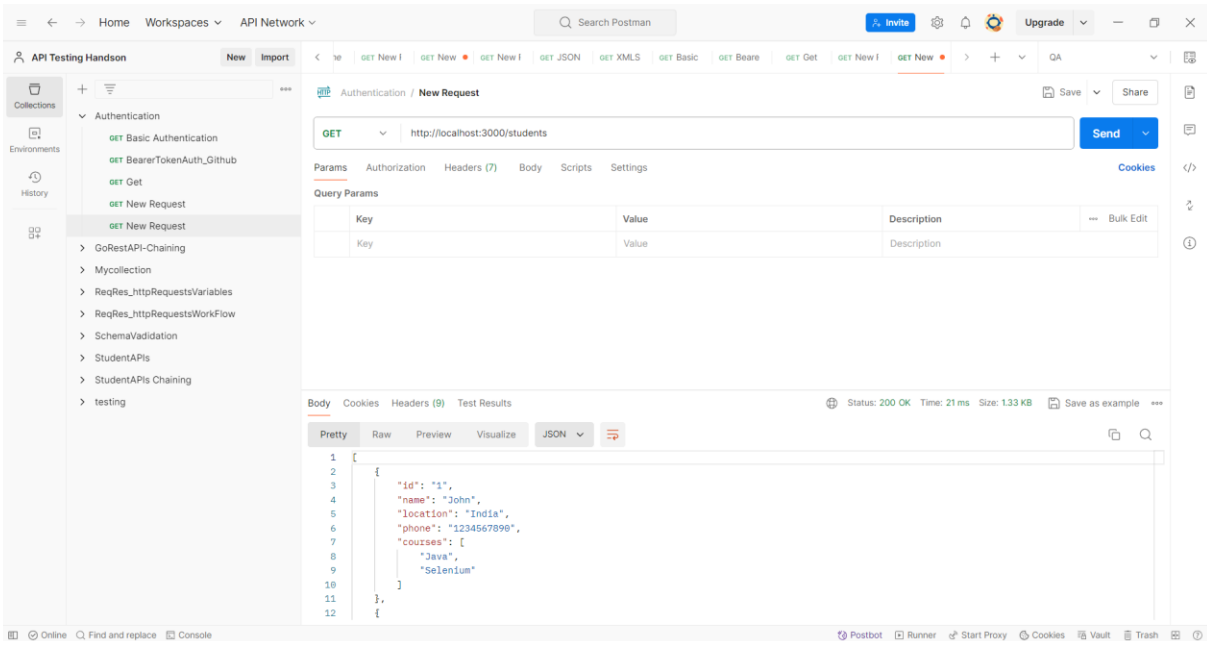
JSONArray jarr = new JSONArray(res.asString[]);

Jar.getJSONObject(2).get(“name”);

Ex: studentsNew.json



```
D:\Automation\API Testing\API_Testing\RestAssuredProject>json-server studentsNew.json
JSON Server started on PORT :3000
Press CTRL-C to stop
Watching studentsNew.json...

♡( ◡‿◡ )

Index:
http://localhost:3000/

Static files:
Serving ./public directory if it exists

Endpoints:
http://localhost:3000/students
```



GET http://localhost:3000/students

Status: 200 OK Time: 21 ms Size: 1.33 KB

```
[
    {
        "id": "1",
        "name": "John",
        "location": "India",
        "phone": "1234567890",
        "courses": [
            "Java",
            "Selenium"
        ]
    },
    {
```

```
[
  {
    "id": "1",
    "name": "John",
    "location": "India",
    "phone": "1234567890",
    "courses": [
      "Java",
      "Selenium"
    ]
  },
  {
    "id": "2",
    "name": "Kim",
    "location": "US",
    "phone": "2345678901",
    "courses": [
      "Python",
      "Appium"
    ]
  },
  {
    "id": "3",
    "name": "Smith",
    "location": "Canada",
    "phone": "3456789012",
    "courses": [
      "C#",
      "RestAPI"
    ]
  },
  {
    "id": "b4f3",
    "courses": [
      "C",
      "C++"
    ],
    "phone": "1234567890",
    "name": "Scott",
    "location": "France"
  },
  {
    "id": "b101",
        "courses": [
            "C",
            "C++"
        ],
        "phone": "1234567890",
        "name": "Scott",
        "location": "France"
    },
    {
        "id": "bed5",
        "name": "Scott",
        "location": "France",
        "phone": "1234567890",
        "courses": [
            "C",
            "C++"
        ]
    },
    {
        "id": "0cab",
        "courses": [
            "C",
            "C++"
        ],
        "phone": "1234567890",
        "name": "Scott",
        "location": "France"
    }
]
```

JSON Array -> JSON Objects (here JSON Objects contains JSON Array)

JSONArray jarr = new JSONArray(res.asString[]);

jarr.getJSONObject(0).getJSONArray(“courses”).get(1); /////to get value from i=0 array index of course=selenium.

RestAssured | API Chaining

Chaining

GoRest

API Chaining means response from one request becomes request for another request.

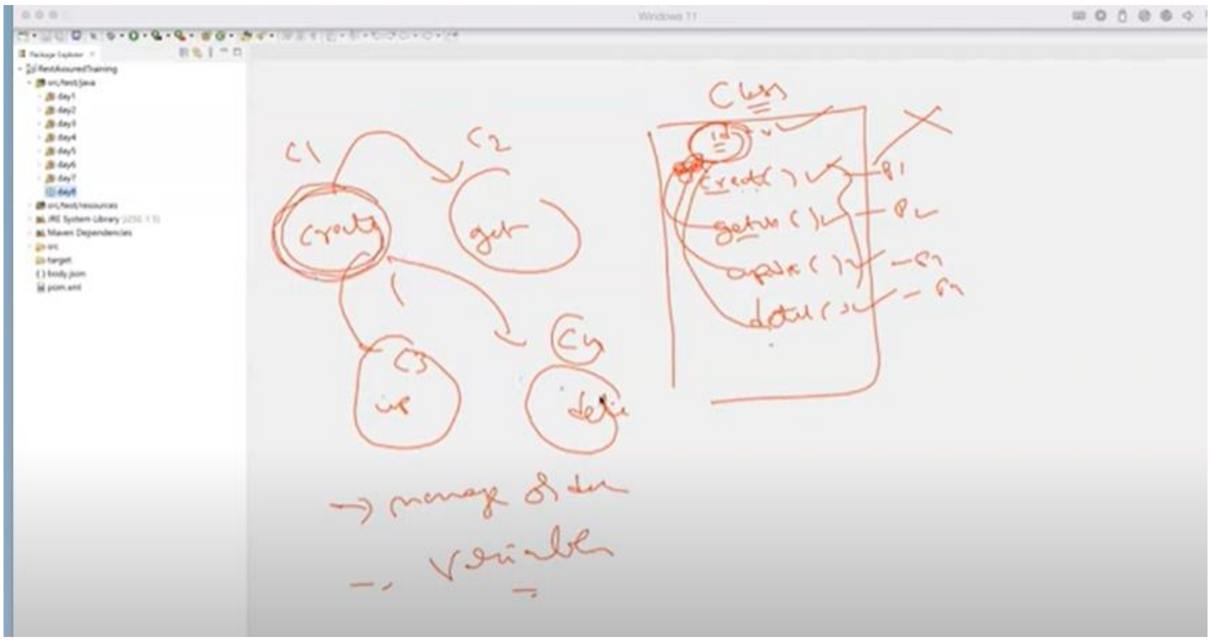
Create -> **ID**

Operations performed:

1) Get user
2) Update user
3) Delete user

Using TestNG feature using TestNG Xml file.

Ex: Go Rest API https://gorest.co.in/



Create user-> get user-> update user-> delete user

Use TestNG feature i.e. ‘**ITestContext context’**.

Ex: **void** tes_createUser(ITestContext context)

context.setAttribute("user_id", id);

void test_deleteUser(ITestContext context)

int id = (Integer) context.getAttribute("user_id");

Now create xml file from package.

Package (right click) -> TestNG -> Convert to TestNG -> store xml file in same package.

Update the request order in testng.xml file as per request priority/flow.

Now execute xml file i.e testng.xml -> run as TestNG Suite.

For Test Level: context.setAttribute();

For Suite Level: context.getSuite().setAttribute();

Ex:

context.setAttribute("user_id", id);

context.getSuite().setAttribute("user_id", id);

int id = (Integer) context.getSuite().getAttribute("user_id");

Part 1: Building API Automation Testing Framework in Rest Assured from Scratch

Framework Development

Framework – maintain all project related files.

Objective:

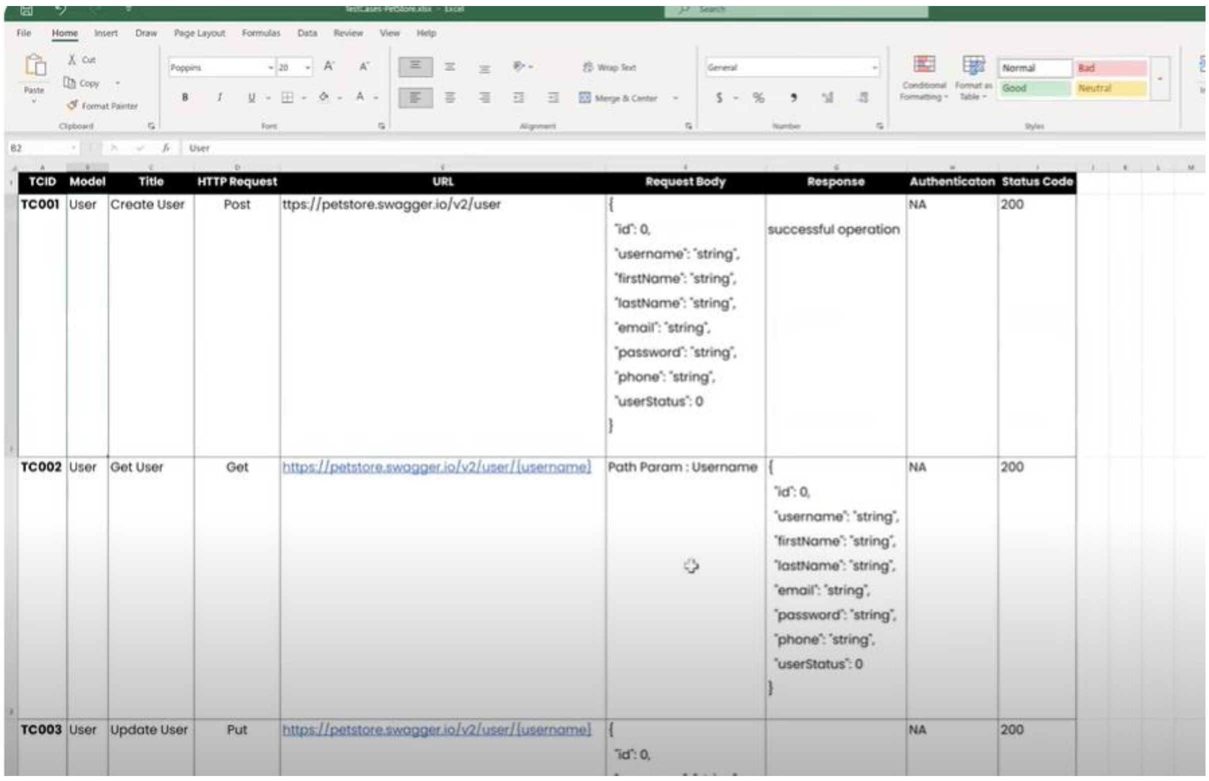
1) Re-usability
2) Maintainability
3) Readability

Hybrid Driven: Combination of 2 frameworks

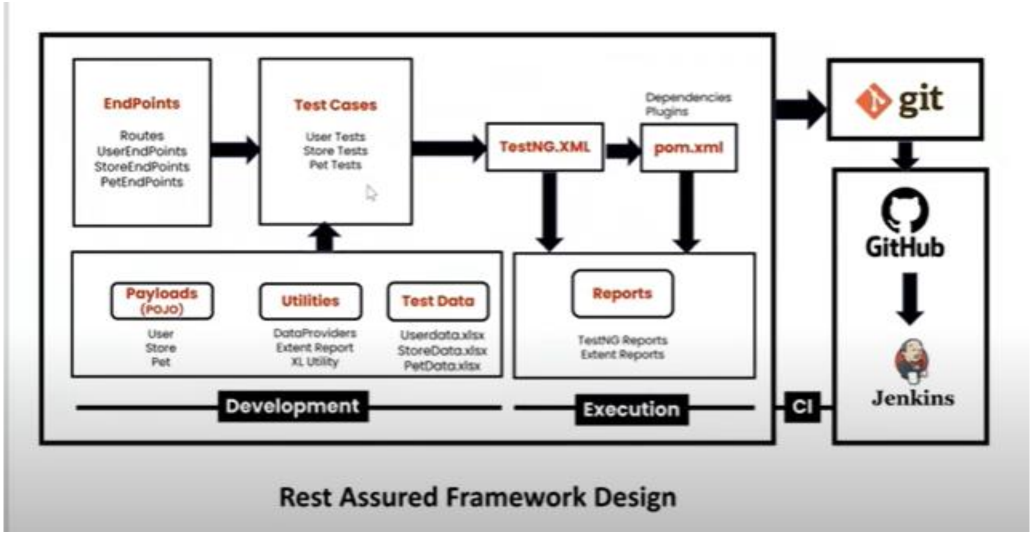
Phases:

1) Understanding requirement:
- Functional specifications (static)
- Swagger
2) Choose automation tool – Rest Assured Library
3) Design
4) Development
5) Execution + CI (Continuous Integration)

Ex: Swagger Petstore: https://petstore.swagger.io/



TCID	Model	Title	HTTP Request	URL	Request Body	Response	Authenticaton	Status Code
TC001	User	Create User	Post	ttps://petstore.swagger.io/v2/user	{ "id": 0, "username": "string", "firstName": "string", "lastName": "string", "email": "string", "password": "string", "phone": "string", "userStatus": 0 }	successful operation	NA	200
TC002	User	Get User	Get	https://petstore.swagger.io/v2/user/{username}	Path Param : Username	{ "id": 0, "username": "string", "firstName": "string", "lastName": "string", "email": "string", "password": "string", "phone": "string", "userStatus": 0 }	NA	200
TC003	User	Update User	Put	https://petstore.swagger.io/v2/user/{username}	{ "id": 0,		NA	200

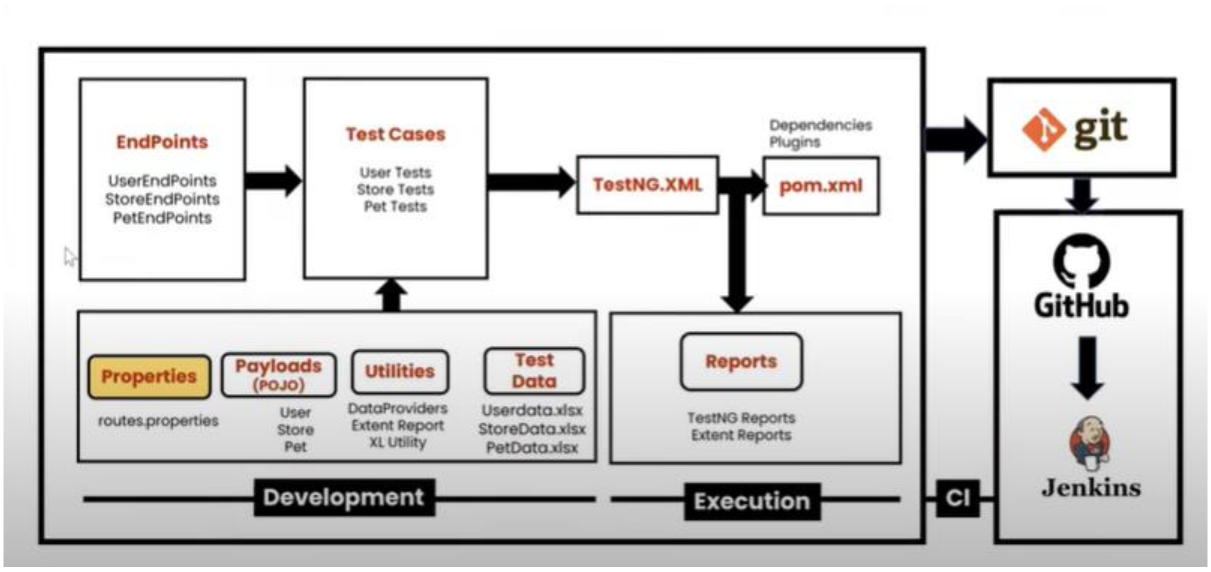


Rest Assured Framework Design

TestNG Test includes following:

```
@Test
void test()
{
//pre-requisite
//request type
//response validation
}
```

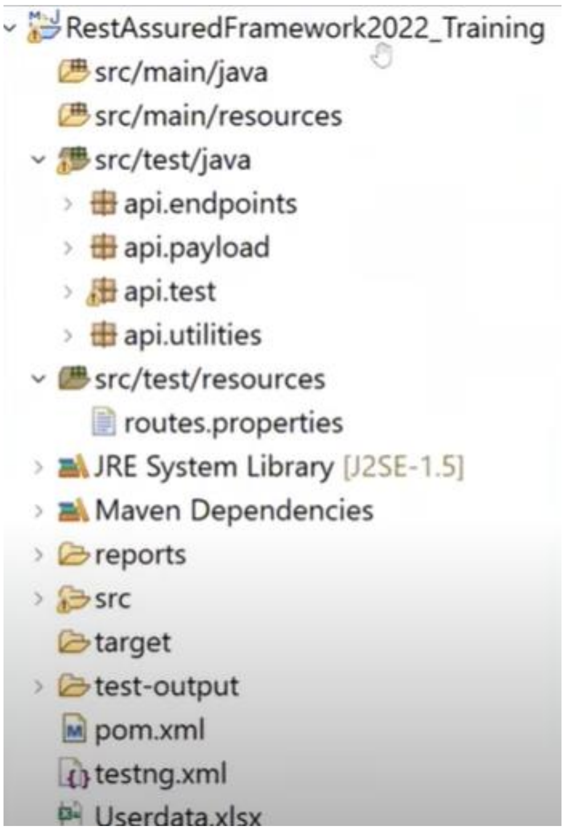
Approach 2 using properties file



Pre-requisites Steps:

1) Create Maven Project.
2) Update pom.xml with required dependencies.
3) Create folder structure.

Rest Assured Framework folder structure:



```
RestAssuredFramework2022_Training
    src/main/java
    src/main/resources
    src/test/java
        api.endpoints
        api.payload
        api.test
        api.utilities
    src/test/resources
        routes.properties
    JRE System Library [J2SE-1.5]
    Maven Dependencies
    reports
    src
    target
    test-output
    pom.xml
    testng.xml
    Userdata.xlsx
```

Designing steps:

4) Create Routes.java --→ contains URL's.
5) Create UserEndPoint.java -→ CRUD methods implementation. (Create Read Update Delete).

Part 2: Building API Automation Testing Framework in Rest Assured from Scratch

6) Create test cases.
7) Create data driven test.
   Pre-requisite: excel sheet data, ExcelUtility file, Dataproviders.

Add Apache POI Maven dependency to pom.xml:
https://mvnrepository.com/artifact/org.apache.poi/poi/5.2.3

```
<!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->

<dependency>

  <groupId>org.apache.poi</groupId>

  <artifactId>poi</artifactId>

  <version>5.2.3</version>

</dependency>
```

Add Apache POI OOXML dependency to pom.xml:
https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml/5.2.3

```
<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->

<dependency>

  <groupId>org.apache.poi</groupId>

  <artifactId>poi-ooxml</artifactId>

  <version>5.2.3</version>

</dependency>
```

8)