Api testing Important Questions.

01/Mar/2023

→ Api - Communicate between two systems, api is used.

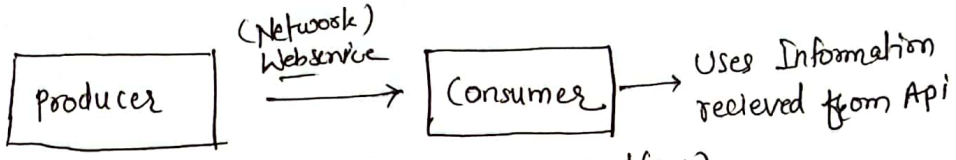
03/Mar/2023.

twitter Api, Google Api.

→ Main difference bet Api and WebWService.

all Webservices are Api's

but all Api's are not webServices.



Simple Ex. method (a+b)
return a+b;

int c = method (a+b);

Api is implimented using Webservice right to access the Api. some URL or some service is needed to access the Api.

Wbe service only useds SOAP, REST, XML-RPC call

→ What are limitations of API usage.

Usage limitations involves Number of hits alowed per day/hr. Service rendered is charging accordingly.

→ Common Api Architechtural ~~Syt~~ style.

- → HTTP for client-server communication
- → XML/JSON as formatting language to cumminate data between two systems.
- → Simple URI as address
- → Stateless Communication.

Who can use Web Api →

Any client device which can consume / support HTTP verbs such as GET, PUT, POST, DELETE etc.

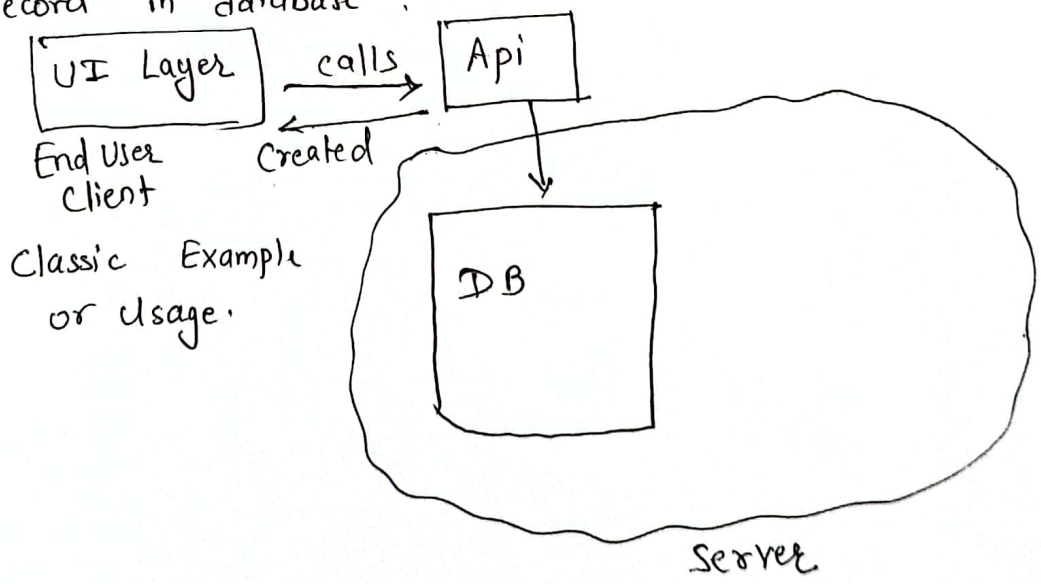
Api donot require any specific configuration to consume. Any device like android phone, Windows Machine, tablet can consume / support

Api testing Important aspects.

1. Functionality (Functional test)
2. Reliability (How it performs during outage situation).
3. Performance test (Load test)
4. Security test (Authentication)
5. Database testing vs Api Response.

→ naveenautomationlabs.com/opencart/ Register. ↓

When you hit a button like 'Register' on a Registration Page in background api call is made to create or add user record in database.



Classic Example or Usage.

In chrome Dev tools → (Right click Inspect on Web page to open)
Click Network tab → Click Fetch XHR.
Refresh Webpage → you will get list of Api calls made.
We can see each api → URL header, payload, response

Advantages of API Testing

Early test of Core Functionality :- Even without UI is available or developed we can catch bugs in Api testing Itself.

Time Effective :- Api testing is much faster than GUI testing. Requires lesser quote. Cost saving for projects.

Language Independent :- transfer modes are completely language Independent. XML/JSON can be consumed for validation using any programing language for test Automation.

Easy Integration with GUI.

What are common protocals used for Api testing.

REST, HTTP, SOAP (these are most popular).

Principles of Api test Design.

five most important principles of API test design are

① setup : Create objects, start services, initialize data
② Execution: Steps to apply API or scenario, Including logging
③ Verification: Oracles to ~~ex~~ evaluate result of execution.
④ Reporting : Pass, ~~bg~~ failed status for TCs
⑤ Clean-up:

What are common Api testing types?

① Validation testing - data is correct
② Functional testing - Whether data is retrieved functionality correct
③ UI testing - Integration of Api with UI
④ Load testing - performance test
⑤ Run time Error detection - Negative test, Server down, 404, resource Not found, proper Error handling

⑥ Security testing - verification of authentications.

What must be checked while performing Api testing

① Accuracy of data
② Schema Validation → Structure of JSON, XML in response.
③ Status code → HTTP status code table (validation of status codes).
④ Authorization Checks → Only acceptable/valid users or auth allowed to access.
⑤ Response time →
⑥ NFT using Jmeter (performance, ~~tod~~ load testing).

Video #2.

08-Mar-2023.

Pavan Sir Api testing Course – Video #1

Basic Needs
TestNG – Runner (Eclipse Plugin)
Maven – Build (Eclipse plugin)
RestAssured - java jars.
Eclipse -

dependencies Needed - in Maven pom.xml
io.restAssured
JSON-path
gson →
json →
Scribejava-apis. → for Fake Data Generator
jason-schemav validator
xml-schema_validator

Http main Requests.

→

RestAssured by default supports BDD style of writing test cases,
gerkin - keywords.

given() → Content type, authentication, query parameters, path parameters etc
when() → Request type, request urls Get, Post, Put, Delete etc.
then() → Validations in then.

website reqres.in (Sample Api's to practice).

Sample first program

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

three static important packages of restAssured.

```
@Test
public void getUsers() {
```

Sample program. (GET) Request

```
@Test
void getUsers(){
    given()          // Given is empty here since we didn't have
                        any pre-requisites in api.
    .when()
      .get("https://reqres.in/api/users2page=2")
    .then()
      .statusCode(200)  // validate status code is 200.
      .body("page", equalTo(2)) // validate body's attribute.
      .log().all();  // prints entire response on console.
}
```

static methods of RestAssured

.log().all() prints your entire response on console (regardless of output format JSON, XML).

POST - Request (Create New User).

For post Request we need Body to be created. It is an important concept.

regres.in (for same api's)

Some important points learn't here are -

① If you want to pass request body as JSON & in form of HashMap which is not recommended way. We need to have GSON in the dependencies.

② when.post() returns Response.

- jsonPath() can be used to validate data in Response body() is also available.

When a post request is executed successfully - 201 is returned.

Sample code -

```
@Test
public createUser(){
  HashMap<String, String> data = new HashMap<String, String>();
                              data.put("name", "ritesh");
                              data.put("job", "trainer");

  given().
  .contentType("application/json")
  .body(data)
  .when()
  .post("https://reqres.in/api/users")
  .then()
  .assertThat().statusCode(201); //
}
```

PUT Request (update existing data).

→ PUT HTTP Request is used to update existing resource.

→ Response returns status code as 200 if successful.

Sample code spiper

```
@Test
 public void updateUser(){
HashMap<String,String> data = new HashMap<String,String>();
    data.put("name", "nitesh");
    data.put("job", "updatedJob"); // field updated.

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

  .when()
  .put("https://reqres.in/api/users/102") // here 102 is hardcoded id we want to update.

  .then()
  .assertThat().statusCode(200)
  .log().all(); //used to print entire response on console.

 }
```

Note:- This is very basic level program. It is not recommended to hardcode data or id in request Url.

DELETE Request.

→ Delete request is used to delete user/resource at server.
→ Returns 204 as status code if successfully deleted

Sample :-

```
public void deleteUser(){

    given()
    .when()
    .delete("https://reques.in/api/users/"+id); //id is your
                                                  id u want to
    .then()                                       delete.
    .statusCode(204);

}
```

Ideally how a delete Request should be tested is using Sequence

e.g Create User → 201 - successfull creation.
delete User → 204 - successfull deletion
getUser → Should return ~~20~~ 404

Sample program available in github
https://github.com/riteshGM/APIRestAssuredLearning repo.
Program Name :- SampleDeleteRequest.java.

Key Notes: ①

In order to get a specific attribute's value from response below code is used.

```
Response res
=
.extract().response();
```

In then() static sequence.

```
int id = res.jsonPath().getInt("id");
```

---

② To print entire response on console.

`.log().all();` is used in then() static sequence.

③ To verify if VV.IMP.

⇒ Converting HashMap to JSON.

```
Map obj = new HashMap();
obj.put("name", "Ritesh");
obj.put("id", "1234");
String jsonText = JSONValue.toJSONString(obj);
```

o/p: {"name":"Ritesh","id":"1234"}

⇒ Creating JSONArray e.g. ["name","Ritesh",12,6000.0]

```
JSONArray arr = new JSONArray();
arr.add("name");
arr.add("Ritesh");
arr.add(new Integer(27));
arr.add(new Double(6000.0));
String JsonString = JSONValue.toJSONString(arr);
```

o/p: ["name","Ritesh",12,6000.0]

JSONArray.

Paran Sir's Session #2
Api testing RestAssured.

How Many Ways we can create Request Body

1) HashMap
2) Org.json library
3) Using POJO class (plain Java object class) (//Most famous way).
4) Using an External File. (JSON File)

#1 Using HashMap

Sample JSON :-

```
{
  "students": [
    {
      "id" : 1,
      "name" : "John",
      "location" : "india"
      "phone" : "12345789",
      "courses": [
        "Java",
        "Selenium"
      ]
    },
    {
      "id" = 2,
      "name" : "Kim"
      "location" : "US"
      "phone": "91756667",
      "courses" : [
        "Python",
        "Appium"
      ] //
    }
  ] //Main Array
}
```

We will see how to add a student in sample JSON.

```
@Test
public void postUsingHashMap(){
    HashMap data = new HashMap();
    data.put("name", "Scott");
    data.put("location", "France");
    data.put("phone", "123456");
    String courseArr[] = {"C", "C++"};
    data.put("courses", courseArr);

    given()
        .contentType("application/json")
        .body(data)
    .when()
        .post("http://localhost:3000/students");
                    //some local host
    .then()
        .statusCode(201)
        .body("name", equalTo("Scott"))
        .body("location", equalTo("France"))
        .body("phone", equalTo("123456"))
        .body("courses[0]", equalTo("C"))
        .body("courses[1]", equalTo("C++"));
}
```

How you validate one of the way

Now HashMap is generally not preferred to be used since it is hardcording of data. In actual project the test data body is bulk.

We will check next option.

#2 Using org.json

We need to add org.json library in class path

Org.json is all same like HashMap, in other words we need to add data into JSONObject instance same way we do for HashMap using put().

Sample code:-

```
@Test
public void testPostUsingJSONObject(){
JSONObject data = new JSONObject(); //org.json maven dependency needed.
    data.put("name", "Scott");
    data.put("location", "France");
    data.put("phone", "123456");
    String coursesArr[] = {"C", "C++"};
    data.put("courses", coursesArr);

    given()
    .contentType("application/json")
    .body(data.toString)    // Vimp please note toString() is used in this case when we supply body.
    .when()
    .post("http://localhost:3000/students")
    .then()
     .statusCode(201)
     .body("name", equalTo("Scott");
                :
     .log().all();
}
```

We can also use jsonpath() to retrive value of any attribute

Sample code implement and available in github

riteshGM

JSONBodyCreation.java program

P.T.O.

#3 Using Pojo class. (Plain old java object). (Session#2 38.50)

Here encapsulation concept of Core Java is used.

Purpose of Pojo is simply like a wrapper class for your data.

You create your Pojo class and set values of each attribute of your pay-load, for request Body

Sample Program:-

Create a Pojo class first:- write program.

```
public class Products_Pojo {
    String title;
    String description;
    int price;
    double discountPercentage;
    int rating;
    int stock;
    String brand;
    String category;
    String thumbnail;
    String images[];   → should be same as attribut Name in JSON.
    //Create getter and setter for each class attribute,
    public String getTitle(){
        return title;
    }
    public void setTitle(String title){
        this.title = title;
    }
    .
    .
    .
    public String[] getImagesArr(){
        return images;
    }
```

P.T.O

```
    public void setImagesArr(String[] imagesArr) {
        this.imagesArr = imagesArr;
    }
}
```

Program which uses this pojo class to create request Body.

```
public class JsonBodyCreation_Pojo {
 @Test
 public void requestBodyUsingPojo() {
    Products_Pojo data = new Products_Pojo();
    data.setTitle("Redmi Note 12 Pro");
    data.setDescription("Redmi Note 12 Pro: Latest Phone");
   data.setPrice(1658);
   data.setDiscountPercentage(15.23);
   data.setRating(4);
   data.setStock(1000);
   data.setBrand("Xiaomi");
   data.setCategory("smartphones");
   data.setThumbnail("https://i.dummyjson.com/data/products/1/
                      thumbnail.jpg");
   String imagesArr[] = {"https://i.dummyjson.com/data/products/1/1.jpg",
          "..... 2.jpg",
          "..... 3.jpg",
          "..... 4.jpg"};
   data.setImagesArr(imagesArr);
```

P.T.O.

```
Response res =
    given()
     .contentType("application/json")
     .body(data)
    .when()
     .post("https://dummyjson.com/products/add")
    .then()
     .extract().response();

assertEquals(res.jsonPath().get("title"), "Redmi Note 12 Pro");
    S.O.p("*** Response Body Printed ***");
    S.O.p(res.asPrettyString());
  }
}
```

4 / Using External JSON File. (Request Body). 51:00 (Session #2).

i) FileReader (java.io) → This is java.io class which allows to read a file.

```
File fileReader = new File (".\\body.json");
FileReader fr = new FileReader (fileReader);
```

ii) JSONTokener (org.json package).

```
JSONTokener jt = new JSONTokener(fr);
```

iii) JSONObject (org.json package).

```
JSONObject data = new JSONObject (jt);
```

↓

Further rest story is same.

Here we read the File as json file wherein Body data is stored.

Above three steps will create JSON Body for Request method.

This approach is pretty straight forward.
We need to have required JSON Body in an external JSON file,

Code Snippheet

```
@Test
public void requestBodyUsingExternalJSONFile() {
    File f = new File (".\\src\\test\\resource\\external.json");
    FileReader fr = new FileReader (f);
    JSONTokener jt = new JSONTokener(fr);
    JSONObject data = new JSONObject (jt);
    S.O.P (data.toString()); // just print to set content is loaded.

    Response res =
        given()
        .contentType("application/json")
        .body (data.toString())
        .when()
        .post ("https://dummyjson.com/products/add")&
        .then()
        .extract().response();
```

```
assertEquals(res.jsonPath().get("title"), "iphone 9");
S.O.P(res.asPrettyString());
}
```

Extra Notes :-

point to note we used sample approach as if used in #2 creation of Request body using JSONObject (org.json).
Only difference here is that we loaded external JSON file using FileReader and then tokenized it using JSONTokener which was supplied to JSONObject creation.
Once JSONObject is created then we can simply used it to post request Body.

Path and Query Parameters.

Session #3.

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

domain — path — query parameter

Using pathParam() we can specify path parameters.

e.g. pathParam("mypath", "users");

"mypath" — name you define; "users" — value

Using queryParam() we can specify query parameters.

e.g.

queryParam("page", 2)

Note:- query parameter once defined it automatically gets embedded in your request in when() section. since in given we have already mentioned it.

Sample Code Snipher

```
@Test
public void testPathandQueryParameteres(){
    Response res =
        given()
        .pathParam("my-path", "users");
        .queryParam("page", "2")
      .when()
      .get("https://reqres.in/api/{my-path}") // please note query
      .then()                                  parameter is embedded automatically
      .extract().response();
    assertEquals(res.jsonPath().get("page", 2);
      S.O.p(res.asPrettyString());
}
```

Cookies and headers Session #3 17:27.

In this section we work on how cookies and headers are handled

```
@Test
public void cookieSampleTest(){
  Response res =
      given()
      .when()
       .get("https://www.google.com")
       .then()
       .extract().response();
  S.O.P("Cookie value is " + res.getCookie("AEC"));
   Sop(res.asPrettyString());
}
```

This way we can retrieve cookie value using attribute Name.

If we want to get all cookies at once
we can use getCookies() → returns Map<String, String>
(Cookie name, value pairs.)

```
Map<String, String> cookiesMap = resp.getCookies();
```

Then Loop through HashMap.

Sniphet.

```
Map<String, String> cookies = res.getCookies();
for(String eachkey : cookies.keySet()){
    S.O.P("Value for key : " + eachkey);
    S.O.P(" cookies.get(eachkey));
}
```

Headers.

Similar to cookies, header can also be retrieved from a response. 57.08

Way Headers is designed

res.getHeader("header-name") returns String value

res.getHeaders() returns Headers → with holds Header

This class contains
- .getName()
- .getValue().

Sample Code Snipheit

```
@Test
public void headerSampleTest(){
    Response res =
            given()
            .get("https://www.google.com")

            .then()
            .header("server", "gws") //we use header() in then
                                     section to validate name-value pair
                                     matched
            .extract().response();

//Retrieving Single Name-value pair.
 s.o.p("Header value is " + res.getHeader("Content-Type"));

// Retrieve all headers at once
   Headers headers = res.getHeaders();
      for(Header eachHeader : Headers){
      s.o.p("Name of Header " + eachHeader.getName());
      s.o.p("Value of Header " + eachHeader.getValue());
      }
 s.o.p("Total headers found " + headers.size());
```

If we want to print headers. we can use log().headers() method which will print header on Console.

Topic log() method Session#3 59:17

Used under then() section, prints output of response on console.

log().body() → print only response body

log().cookies() → print only cookies.

log().headers() → print only header.

log().all() → print entire response (prints everything).

RestAssured : Validating Response Session#4

JSON path finder is used to find Json path.

First approach of Validating Response

Validating using default methods in then() method section.

```
@Test
public void getUsers(){
    given()
    .when()
    .get("https://reqres.in/api/users?page=2")
    .then()
    .statusCode(200)
    .header("Content-Type", "application/json; charset=utf-8")
    .header("Connection", "keep-alive")
    .body("total_pages", equalTo(2))
    .body("page", equalTo(2))
    .body("data[1].id", equalTo(8))
    .body("data[1].email", equalTo("lindsay.ferguson@reqres.in"))
    .body("support.url", equalTo("https://reqres.in/#support-heading"))
    .log().body();
}
```

This approach is useful when your Response is relatively simple and small or manageable.

This is easiest approach.

Refer to sample code JSONResponseValidation.java in ApiRestAssuredLearning in my github.

Approach #2 Session #4 19:00

Capture your response in a variable and Assert them.

ResponseOptions class's static methods are used.

We have several methods available to retrieve values from a Response.

Cookie (String cookie_name) returns value in string

getCookie(String cookie_name) returns value in string

header (String header_name) returns value in string

getHeader(String name) returns value in String

body() returns Response Body as ResponseBody.

getBody() returns Response Body as ResponseBody

~~status()~~ returns status code as int value.
statusCode()

getStatusCode() returns status code as int value.

asPrettyString() used to get printable fomatt of Response

contentType() returns header Content-type as String

getContentType() —||—

getSessionId() returns sessionID as String.

getTime() returns response time in millisecs of long type
-1 is response time could not be measured

sessionId() returns sessionID as String

jsonPath() → Used to retrieve particular attribute. It is like Xpath in Selenium.

P.T.O sample program mentioned.

```
@Test
public void responseBodyValidation_UsingResponseInstance(){
    Response res = get("https://reqres.in/api/users?page=2");
//Note: it is note necessary to always write your test in given,
when, then format. We can also directly use static methods
of RestAssured library

Assert.assertEquals(res.header("Content-Type"), "application/json;
                                                charset=utf-8");
Assert.assertEquals(res.jsonPath("data").getString("data[1].id").
                    toString(), "8");
Assert.assertEquals(res.jsonPath().getString("data[1].email"),
                    "lindsay.ferguson@reqres.in");
}
```

Whole point here is you get Response and validate it seperately, line by line.

This approach is generally preferable in project frameworks.

-x-

28.32.

Some more advance validations (Session #4 28:39)

Approach #3 Using JSONObject to read JSON

Mainly used to search through json and useful when order of your attribute is dynamic.

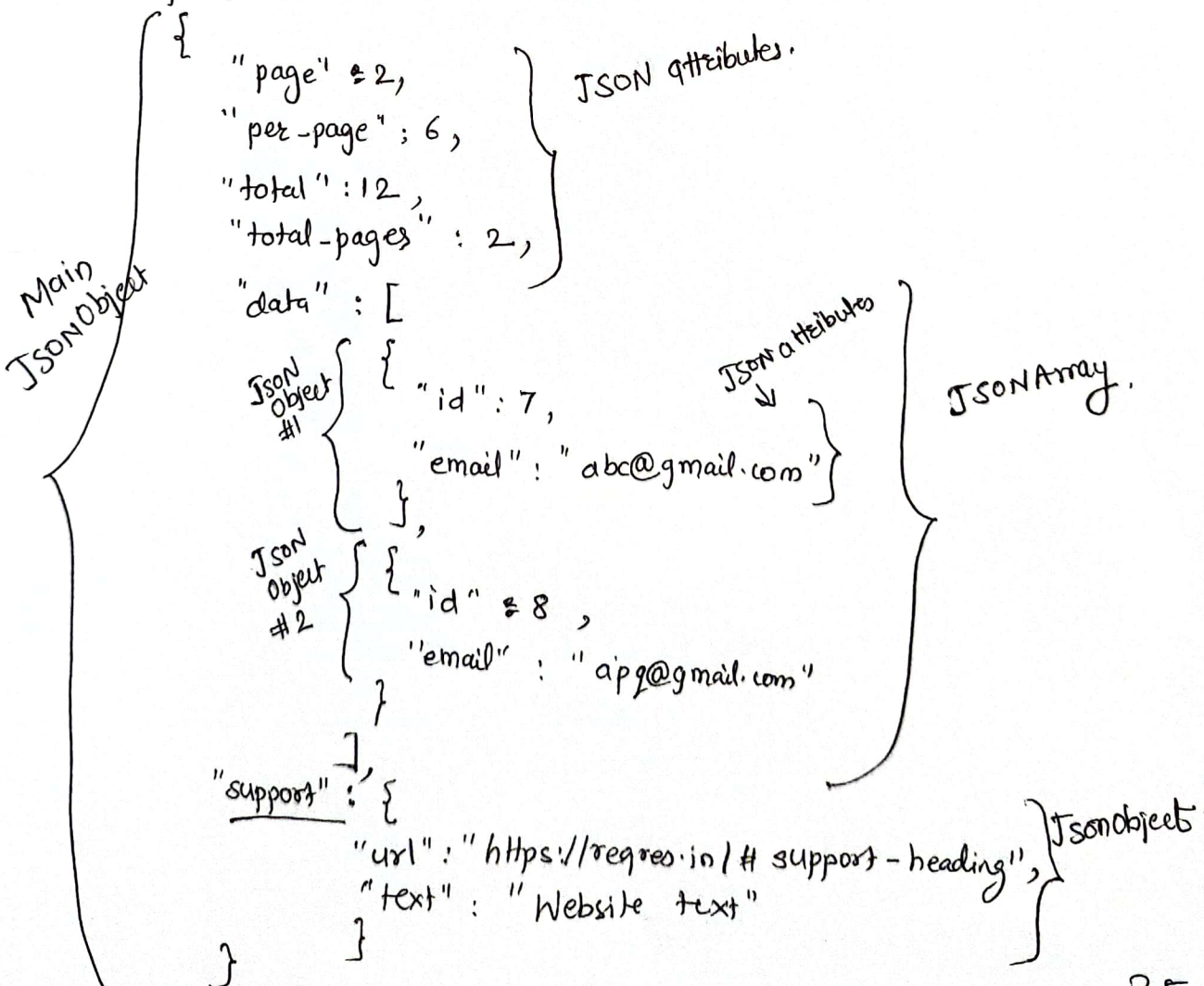
Also suppose you need to loop through attributes for a JSONObject in that case as well JSONObject approach is useful.

Classic example like you have a ~~sto~~ store api which has price for each book and we want to validate whether sum of all bookprice is equal to certain value or not we can do such ~~rad~~ validation using JSONObj.

Code Snip:- To understand usage of JSONObject approach we need to understand JSON structure first.

@Test

Sample JSON.



Suppose we need to check if id=10 is present in JSON Response.

Code snippet.

```
@Test
    public void jsonValidation_UsingJsonObj(){
     Response res = given()
                    .contentType(ContentType.JSON)
                  .when()
                    .get("https://reqres.in/api/users?page=2")
                  .then()
                  .log().body().extract().response();
                  // print body.   // get response

JSONObject jo = new JSONObject(res.asString());
boolean found = false;
for(int i=0 ; i< jo.getJSONArray("data").length(); i++)
    {
String id = jo.getJSONArray("data").getJSONObject(i).get("id").toString();
     if(id.equals("10"))
          {
            found = true;
            break;
          }
    }
Assert.assertEquals(found,true);
}
```

Suppose we need to access a URL under support.

```
@Test
public void jsonValidation_Verifysupport(){
    Respones res =
                given()
                .contentType(ContentType.JSON)**
               .when()
               .get("https://reqres.in/api/users2page=2")
               .then()
               .log().body().extract().response();
JSONObject jo = new JSONObject(res.asString());**
String URL = jo.getJSONObject("support").get("url").toString();
Assert.assertEquals(url, "https://reqres.in/#support-heading");
```

Session #4 Completes here

Session #5

Parsing XML Response

Similar to JSON we have different approaches to traverse through / validate XML Response

Approach #1 Without Using Response object() - directly validating in then() section.

Sample XML

```
<TravelerResponseInformation>                          → Root Node.
<TravellerinformationResponse xmlns:xsd="http ..... >
  <page>1</page>                                       → child-nodes.
  <per-page>10</per-page>
  <total_pages>1326</total_pages>
  <travelers>
    <Travelerinformation>                              ⇒ Inner child Nodes
    <id>11133</id>
    <name>Developer</name>
    <email>Developer12@gmail.com</email>
    <address>USA</address>
    <createdat>0001-01-01T00:00:00</createdat>
    </Travelerinformation>

    <Travelerinformation>                              Inner child Nodes.
     <id>11134</id>
     <name>AS</name>
     <email>qweqw@gmail.com</email>
     <address>USA</address>
     <createdat>0001-01-01T00:00:00</createdat>
     </Travelerinformation>

</travelers>
</TravellerinformationResponse>
```

Working with XML is relatively simple and straight forward as compared to JSON.
The information and locators are easy to Right.
The locators are called as xpath.

Code snipher (using approach #1)

```
@Test
public void xmlValidation(){

    given()
     .when()
     .get("http://restapi.adequateshop.com/api/Traveler?page=1")

     .then()
     .statusCode(200)
     .header("content-type", "application/xml; charset=utf-8")
     .body("TravellerinformationResponse.page", equalTo("1"))
     .body("TravellerinformationResponse.travelers.Travelerinformation[0].
            name", equalTo("Developer"));

}
```

Using methods in then() section

Approach #2
Capturing Response in Response class instance type and then using methods of Response class to retrieve values.

```
@Test
public void xmlValidation_UsingResponseObj(){
    Response res = given()
                   .when()
                   .get("http://restapi.adequateshop.com/api/Traveler?
                                                    page=1")
                   .then().extract().response();
Assert.assertEquals(res.statusCode(), 200);
Assert.assertEquals(res.header("content-type"), "application/xml;
Assert.assertEquals(res.xmlPath().get("Travelerinformation.page").toString(),"1");
Assert.assertEquals(res.xmlPath().get("TravelerinformationResponse.travelers.
                    Travelerinformation[0].name"), "Developer");
}
```

Approach #3 Using XmlPath class to parse xml Body.

Key important point here any node can be simply access using either get() get("locator") or getList("locator").

↓ returns String single value.

↓ Returns values in List<String>

~~Xml~~ XmlPath is available in io.restassured.path.xml library.

Code Snipheti-

```
@Test
public void xmlValidation_UsingXmlPathObj(){
Response res =
        given()
        .when()
        .get("http://restapi.adequateshop.com/api/Traveler?pages=1").
        .then().extract().response();

XmlPath xmlObj = new XmlPath(res.asString());

// Total Number of travellers returned in a page
int totalTravellers = xmlObj.getList("TravelerinformationResponse.
                travelers.Travelerinfomation").size();
S.O.p("Total travelers found as :" + totalTravellers);
Assert.assertEquals(totalTravellers, 10);
// Verify if traveller named "Developer" is present in page
   response.
Assert.assertEquals(xmlObj.getList("TravellerinfomationResponse.travelers.
        Travelinformation.name").contains("Developer"), true);
```

this is method of List returns Boolean true if found in List.

} whichever attribute needs to be accessed can be retrived in a List<String> or String variable and then assert it.

Session #5 55:15.

File Upload and download Files.

File upload :-

It mainly needs form data information which is passed in Body.

However to do this using RestAssured there is a typical way.

Sample Code Snipheet

```
@Test
void singleFileUpload () {
File myfile = new File ("C:\\Automation Practice\\Test1.txt");

given()
    .multiPart("file", myfile) // Using this method we supply file to be uploaded.
    .contentType ("multipart/form-data")
                // We are specifying form data is being sent.

.when()
  .post("http://localhost:8080/uploadFile")

.then()
  .statusCode (200)
  .body("fileName", equalTo ("Test1.txt"))
  .log().all();
}
```

In session sir wed local server hosted program which was serving file upload api locally.

Sample Response P.F.O.

Sample Response of Post Request

```
HTTP/1.1 200
Content-Type : application/json
Transfer-Encoding : chunked
Date : Fri, 05 Aug 2023 02:19:57 GMT

{
  "fileName" : "Test1.txt",
  "fileDownloadUri" : "http://localhost:8080/downloadFile/Test1.txt",
  "fileType" : "text/plain",
  "size" : 70
}
```

Uploading Multiple Files.

```
@Test
void multipleFilesUpload(){
File myFile1 = new File("C:\\AutomationPractice\\Test1.txt");
File myFile2 = new File("C:\\AutomationPractice\\Test2.txt");

given()
  .multiPart("files", myFile1);  //for multifiles we use key as files.
  .multiPart("files", myFile2);
  .contentType("multipart/form-data")
.when()
  .post("http://localhost:8080/uploadMultipleFiles")
.then()
  .statusCode(200)
  .body("[0].fileName", equalTo("Test1.txt"))
  .body("[1].fileName", equalTo("Test2.txt"))
    .log().all();
}
```

Sample Response

```
[
    {
        "fileName": "Test1.txt",
        "fileDownloadUri": "http://localhost:8080/downloadFile/
                                                Test1.txt",
        "fileType": "text/plain",
        "size": 70
    },
    {
        "fileName": "Test2.txt"
        "fileDownloadUri": "http://localhost:8080/downloadFile/
                                                Test2.txt",
        "fileType", "text/plain",
        "size", 70
    }
]
```

1:25:16.

Session #5 1:20:00

File download :-

```
@Test
 public void fileDownload(){
   {
     given()
      .when()
       .get("http://localhost:8080/downloadFile/Test1.txt").
      .then()
         .statusCode(200)
         .log().all();
   }
```

Downloads file to specific location as defined.

JSON Schema Validation

We use JSONSchema Validator which comes from
io.restassured.module.jsv.JsonSchemaValidator;

JSON schema mainly contains the data-type of structure format specification.

Using JSON schema we can validate
- → Data present is JSON Response Body is a per expected data type or not.
- → Whether the attribute is an array or simple attribute-value as mentioned in JSON schema.
- → In simple words using JSON schema we can validate Response beyond data checks.

We can generate JSON schema for any available JSON Body using https://jsonformatter.org/json-to-jsonschema (Converter online)

place your schema file under src/test/resources
e.g. UsersJsonSchema.json

Sample Code:-

```
@Test
public void schemaValidation(){
   given()
   .when()
   .get("https://reqres.in/api/users?page=2")
   .then()
   .log().body()
   .assertThat().body(JsonSchemaValidator.matchesJsonSchemaInClasspath
                      ("UsersJsonSchema.json"));
      // JsonSchemaValidator will automatically look for schema
   }     file in classpath. No need to give absolute location of file.
```

XML Schema Validation Session #6 23:51

In XML files schema is in .xsd type.
Similar to JSON we can use any online tool to get XML Schema using your XML Body.

RestAssuredMatchers is used from
io.restassured.matcher.RestAssuredMathers.*;

```
@Test
public void XMLSchemaValidation {
{
given()
.when()
.get("http://restapi.adequateshop.com/api/Traveler")
.then()
.assertThat().body(RestAssuredMatchers.matchesXsdInClasspath
("traveler.xsd"));
// Need not to be absolute path, program
automatically seraches in Classpath.
}
```

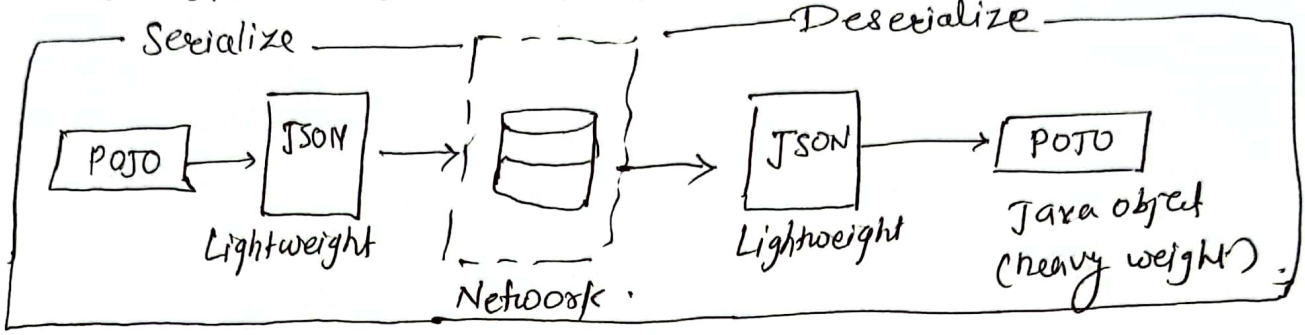
For online conversion of XML to XSD Schema we used
www.liquid-technologies.com/onile-xml-to-xsd-converter.

Serialization and De-serialization Session#6 40:57

Body (JSON) ----> Request ----> Response (JSON)

Serialization POJO ----> JSON
We send.

De-serilization JSON ----> POJO.



In simple words No matter how you create/pass-on Body (hashMap, JSONObject, POJO) it is automatically serialized to JSON and sent to server. This is taken care by RestAssured.

and vice-versa.

POJO to JSON conversion (serialization) example

Code Sample:

```
@Test
void convertPojo2Json() throws JsonProcessingException {
  Student stupojo = new Student() ; //pojo class object
   stupojo.setName("scott");
   stupojo.setLocation("France");
   stupojo.setPhone("123456");
   String courseArr [] = {"C", "C++"};
   stupojo.setCourses(courseArr);

  // convert Pojo (Java Object) -> Json object
  ObjectMapper objMapper = new ObjectMapper() ;
          // com.fasterxml.jackson.databind.ObjectMapper;
```

```
String jsondata = objMapper.writeWithDefaultPrettyPrinter().
                  writeValueAsString(stupojo);
S.o.p(jsondata);
```

—x—

Now process of converting JSON to Pojo is called De-serialization

Sample Code.

```
@Test
void convertJson2Pojo() throws JsonProcessingException {
  String jsondata = "{\r\n"
      + " \"name\" : \"Scott\", \r\n"
      + " \"location\" : \"France\", \r\n"
      + " \"phone\" : \"123456\", \r\n"
      + " \"courses\" : [ \"C\",\"C++\" ] \r\n"
      + "}";
  // convert json data to Pojo.
  ObjectMapper objMapper = new ObjectMapper();
  Student objstu = objMapper.readValue(jsondata, Student.class);
                                        ↑ which json   ↑ what type of Pojo.
  s.o.p("Name :"+ objstu.getName());
}
```

—x—

Session #6 ends here.

Type of Authorizations

Session #7

Authentication - Use credentials are valid or not

Authorization - Whether a valid use is having access permissions or not.

⇒ Authentications are pre-requisites

There are several different types of Authentications in API's.

#1 Basic Authentication

Sample code

```
@Test
public void testBasicAuthentication(){

    given()
      .auth().basic("postman", "password")

    .when
    .get("https://postman-echo.com/basic-auth")

    .then()
    .status(code(200)
     .body("authenticated", equalTo(true))
     .log().all();
}
```

9.11

→ Basic, digestive and preemptive authentications are same in nature whoever internally algorithm is slightly different.

→ Code snipet above will be same for digestive and preemptive authentication. Only the method in given() section will change.

#2 Digest

⇒

```
given()
  .auth().digest("postman", "password")
```

#3 preemptive

⇒

```
given()
  .auth().preemptive().basic("postman", "password")
```

In pre-emptive auth basic is internally used. The design is in such a way extra layer is added above basic auth.

#4 Bearer token Authentication

Bearer Authorization works using token system. Like we can create token from github application and use it for authentication for any further requests / access.

Bearer Auth information is sent via headers in api.

Key = "Authorization"
Value = "Bearer" + string-token

Sample Code Sniphet

```
@Test (priority=4)
void testBearerToken_Authentication(){
 String bearerToken = "ghp_24PH----------KP";

 given()
   .headers("Authorization", "Bearer " + bearerToken)

 .when()
   .get("https://api.github.com/user/repos")

 .then()
   .statusCode(200)
   .log().all();
}
```

#5 OAuth 1.0 and OAuth 2.0

Again way we pass auth infomation in given() section only that will change.

OAuth authentication has four main parameters which are utilized.

⇒ consumerKey, consumerSecret, accessToken, tokenSecret

All these keys are shared by api developer and same we have to use.

Code Snipet

```
@Test
public void testOAuth1Authentication(){
  given()
    .auth().oauth("consumerKey","consumerSecret","accessToken",
                  "tokenSecret")
  .when()
    .get("URL")
  .then()
    .statusCode(200)
    .log().all();
}
```

However many of Api's are upgraded to OAuth2 authentication

OAuth2 authentication

There are certain steps to generate token in this case. Multiple requests are sent and at run-time token is generated.

```
@Test
public void testOAuth2Authentication(){
    given()
     .auth().oauth2("ghp_24phOI.....KP")
  .when()                          //Token ye key.
     .get("https://api.github.com/user/repos")
  .then()
    .statusCode(200)
    .log().all();
}
```

#6 Api Key authentication

In this case user is requested to generate api key on the website of Api. This way user have to use google / microsoft account to generate Api key and in all future subsequent request that Api key can be used to retrieve information from Api Requests.

Depending on the design of API we have to pass key in Request either (1) in header .header()

(2) in QueryParamenter in Given() section.

.queryParam()

```
@Test
public void testApikeyAuthentication() {
    given()
      .queryParam("appid", "f29c.....2c") //appid is key
                                  value
    .when()
      .get("api.openweathermap.org/data/2.5/forecast/daily?
          q=Delhi&units=metric&cnt=7")
    .then()
      .statusCode(200)
      .log().all();
}
```

Faker Library

Session#7 50:00

In testing world many of the times we need to generate or pass on dummy data inOrder to fill up different fields on WebUI or fill-up attributes in Api. If we hard-code the data it would not be wise to test same script again and again with exactly same dummy data.

To solve this problem we use faker Library which generates dummy data on the go randomly.

Just search over google Java Faker Github

maven dependency

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

Sample Usage Code Snipt

```
@Test
public void
```

Classic use case of faker Library - Consider you have a POJO which needs data to feed in and create an instance which we can pass on to create Request Body

We can get Random data passed on using Faker Library

```
public class DummyProductData Generator{
    public static Products_Pojo generate_Data(){
        Faker dummy = new Faker();
        Products_Pojo data = new Products_Pojo();
        data.setTitle(dummy.book().title());
        data.setPrice(dummy.number().randomDigit());
        data.setBrand(dummy.book().author());
        data.setCategory(dummy.book().genre());
        return data;
    }
}
```

Usage of dummyDataGenerator

```
@Test
public void requestBodyUsingPojo() {
  Products_Pojo data = DummyProductDataGenerator.generateData();
  Response res =
        given()
        .contentType("application/json")
        .body(data)
        .when()
        .post("https://dummyjson.com/products/add")
        .then()
        .extract().response();
   assertEquals(res.jsonPath().get("title"), data.getTitle());
}
```

In above code we utilize faker library to set Products_Pojo. This way QA does have to worry about changing input data in each Run.

Session#7 ends

Topic: Api chaining

Session # 8

→ Manage Order of your tests
→ Manage variables which we want to record and use in subsequent Api Requests.

Create User → Get User → Update User → delete User
This is the flow - classic example of Api chaining usage

Create User Program

```
@Test
void test_createUser (ITestContext context) {
    Faker faker = new Faker();
    JSONObject data = new JSONObject();
    data.put("name", faker.name().fullName());
    data.put("gender", "Male");
    data.put("email", faker.internet().emailAddress());
    data.put("status", "inactive");

String bearerToken = "c35e10e7 ..... 7b06" };
                                  ↳ Long token string
int id = given()
          .headers ("Authorization", "Bearer " +bearerToken)
          .contentType("application/json")
          .body(data.toString())
        .when()
          .post("https://gorest.co.in/public/v2/users")
          .jsonPath().getInt("id");
    S.O.p("Generated id is : " +id);
    context.setAttribute("user_id", id);
                                  //This once set will be available to other @Test during execution.
}
```

key_name: "user_id"
value: id

Note this only works if you run tests via testNG.xml

Get User Program

```
@Test
void test_getUser(ITestContext context) {
int id = (Integer) context.getAttribute("user_id");
                //(This is utilized which was set at
                    createUser Program previously).
String bearerToken = "c35e10e7.....b06";
    given()
      .headers("Authorization", "Bearer " + bearerToken)
      .pathParam("id", id)
   .when()
      .get("https://gorest.co.in/public/v2/users/{id}")
   .then()
      .statusCode(200)
      .log().all();
}
```

Update User Program

```
@Test
public void test_updateUser(ITestContext context){
   Faker faker = new Faker();
   JSONObject data = new JSONObject();
   data.put("name", faker.name().fullName());
   data.put("gender", "Male");
   data.put("email", faker.internet().emailAddress());
   data.put("status", "active");

String bearerToken = "c35e10e7.....b06";
int id = (Integer) context.getAttribute("user_id")
                    ^                  → Returns Object type.
   given()
      .headers("Authorization", "Bearer " + bearerToken)
      .contentType("application/json")
      .body(data.toString())
      .pathParam("id", id)
```

P.T.O.

```
.when()
    put("https://gorest.co.in/public/v2/users/{id}")
.then()   (//you can use then section if want to
                validate anything)
    s.o.p("Generated id is : " +id);
}
```

Now we can try deleting the user we created:

Delete User Program

```
@Test
pubic void test_deleteUser(ITestContext context){
String bearerToken = "c35e10e74 ..... b06";
int id = (Integer) context.getAttribute("user_id");

  given()
     .headers("Authorization", "Bearer " + bearerToken)
     .pathParam("id", id)
  .when()
     .delete("https://gorest.co.in/public/v2/users/{id}")
  .then()
     .statusCode(204)
     .log().all();
}
```

Path Parameter

Programs structure.

package
- CreateUser.java
- DeleteUser.java
- GetUser.java
- UpdateUser.java

Generate XML (testNG) using Right Click on package → TestNG → Convert to testNG.

It will give testNG.xml file as below (save in same package).

testNG.xml file will look as

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite">
  <test thread-count="5" name="test">
    <classes>
      <class name="day8.CreateUser"/>
      <class name="day8.GetUser"/>
      <class name="day8.UpdateUser"/>
      <class name="day8.DeleteUser"/>
    </classes>
  </test>
</suite>
```

Sequence mentioned here in same sequence Test will Run.

Make a note ~~ContextTes~~ ITestContext will only work if all your test are gathered into single <test> in your testNG.xml.

In other words scope of ITestContext is limited to <test> tag only.

If we arrange ~~<classes>~~ Each class individually in seperate <test> tag ITestContext won't work.

e.g.

```
<test name="Test1">
  <classes>
    <class name="day8.CreateUser"/>
  </classes>
</test>
<test name="Test2">
  <classes>
    <class name="day8.GetUser"/>
  </classes>
</test>
```

ITestContext scope

ITestContext Refreshed.

ITestContext won't work In this Case.

If we want to access context variable at entire suite level we need to change how we set context variable and how we retreive context variable.

In CreateUser.java

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

In other programs where we use this attribute

In getUser.java

```
context.getSuite().getAttribute("user_id");
```

Same in (updateUser.java and deleteUser.java)

This way we will cover both suite level and single <test> ~~te~~ level.

Since Suite is higher level scope in TestNG. preferably use context.getSuite().setAttribute(key, value) in your framework.

—x—

Session #8 ends here

Course Ends as well.