What is query and path parameters?

Query params:filter results based on some condition

e.g. //http://localhost:8080/student/list?programme=law&limit=3

//http://localhost:8080/student/list 🡪gives all student info

//http://localhost:8080/student/list?programme=law&limit=3 🡪gives students with program=law and returns first 3 results alone .

Path params: identify particular resource e.g.based on id

Get call examples:

@Test

**public** **void** getallstudentinformation() {

Response res = *given*().when().get("/list");

System.***out***.println("Responsebody is:" + res.body().asString());

System.***out***.println("response is:" + res.asString());

System.out.println("Responsebody in a formatted way is:" +

res.body().prettyPrint());

// validate the status code

*given*().when().get("/list").then().statusCode(200);

}

@Test

**public** **void** getspecificstudentinfo() {

Response res1 = *given*().when().get("/1");

System.***out***.println("response for student id 1:" + res1.asString());

}

@Test

**public** **void** getstudentinfowithparam() {

// parameter shd be passed in given()

Response res = *given*()

.param("programme","law")

.param("limit"=3)

.when().get("/list");

System.***out***.println("response for law:" + res.prettyPeek());

}

Notes:

Post call:

🡪creating a new user info

Step1:create user/student POJO in a separate class in src/main folder

Initialize member variables

Create a parametrized constructor

Define getter and setter

And now POJO creation is complete

Step 2: add all the courses in array list ,since courses parameter is an array

ArrayList courses=**new** ArrayList();

courses.add("java");

courses.add("c++");

courses.add("selenium");

Step3: Create obj.ref variable to the POJO and pass all the parameters

It can be done by two ways,

Method1:either create a parameterized constructor in POJO and pass the parameters in test class.

student stu=**new** student("jansi","Lakshmanan","jansi.bt@gmail.com","software testing",courses);

method 2: creating obj ref var and using set method as below,

student stu=**new** student();

stu.setCourses(courses);

stu.setEmail("jansi@gmail.com");

stu.setFirstName("jansi");

stu.setLastName("lakshmanan");

stu.setProgramme("testing");

here,we need not create any constructor in POJO

step4:

Now,write post call

*given*()

.contentType(ContentType.***JSON***)

.when()

.body(stu)//stu is the obj ref var for student (POJO) class

.post()

.then()

.statusCode(201);

Errors:

Cannot serialize object because no JSON serializer found in classpath. Please put either Jackson or Gson in the classpath.

Solution: add Jackson databind maven dependency

Why is Jackson dependency added?

It is used for data binding i.e. converting JSON to and from POJO(Plain old java object )

Put call :

This is similar to post call,except the below changes:

1.change the method name to put() and pass the unique id in the put call()

2.update the values of the field.

Note:

We have to set values for all the variables again including the values which r not req to update.

ArrayList courses=**new** ArrayList();

courses.add("java");

courses.add("c++");

courses.add("selenium");

student stu=**new** student();

stu.setCourses(courses);

stu.setEmail("jansi@gmail.com");

stu.setFirstName("theju");

stu.setLastName("rajagopal");

stu.setProgramme("testing");

*given*()

.contentType(ContentType.***JSON***)

.when()

.body(stu)

.put("/101")

.then()

.statusCode(200)

Here,values updated are firstname and lastname alone,but all the values r set again

And id of the record is 101.

Patch() request:

Similar to put,except that it is used for Updating single field of the particular user with patch () request.

**public** **void** patch()

{

ArrayList courses=**new** ArrayList();

courses.add("java");

courses.add("c++");

courses.add("selenium web driver");

student stu=**new** student();

stu.setCourses(courses);

stu.setEmail("jansi.bt@gmail.com");

stu.setFirstName("theju");

stu.setLastName("rajagopal");

stu.setProgramme("testing");

*given*()

.contentType(ContentType.***JSON***)

.when()

.body(stu)

.patch("/101")

.then()

.statusCode(200);

Delete() request:

Since,we are just deleting,we need not pass any parameters /need not add content-type or need not add any body in the when(),

Just pass the unique id in the delete call

@test

Public void delete()

{

Given()

.when()

.delete(“/101”)

.then()

.statuscode(204)

//for delete req,status code is 204

Reconstructing the code:

Here,we are initializing the base URI,port,base path in the @BeforeMethod in each class,instead we can put these code in a base class.

In src\main\java\com.student.base\TestBase.java🡪add

**public** **class** TestBase {

@BeforeMethod

**public** **void** init() {

RestAssured.*baseURI* = "http://localhost";

RestAssured.*port* = 8080;

RestAssured.*basePath* = "/student";

}

}

And then extend baseclass in all other test class.

Note:don’t forget to add @BeforeMethod annotation here.only then new request will be sent.

Errorrs:

1.RestAssured cannot be resolved

when we create a base class in src/main🡪we get an error in the import

Reason:

In POM.xml when we add dependency for Rest Assured,there may be a line for scope which is limited only to src/test

<dependency>

<groupId>com.jayway.restassured</groupId>

<artifactId>rest-assured</artifactId>

<version>2.9.0</version>

<scope>test</scope></dependency>

i.e.scope of RestAssured is extended only to src/test package and we are not able to import it in src/main

Solution:

remove the tag of scope

<scope>test</scope></

Logging Request information:

It is nothing but printing the info requested in the console.

Note:

\*.log() method is used to print out the info in the console and it is available within given() method

.headers()🡪to print req heders

.param🡪to print out req parameters.

.body() 🡪to print out req.body

.all🡪to print out all request params.

. .ifValidationFails()🡪to print logs only if test case validation fails

@Test

**public** **void** test1() {

// to print out all the request headers

System.***out***.println("\*\*\*\*Printing request headers\*\*\*\*\*");

*given*().log().headers().when().get("/1").then().statusCode(200);

}

@Test

**public** **void** test2() {

// to print out all the request parameters

System.***out***.println("\*\*\*\*Printing request parameters\*\*\*\*\*");

*given*().param("programme", "Computer Science").param("limit", 5).log().params().when().get("/list").then()

.statusCode(200);

@Test

**public** **void** test3() {

// to print out all the request body in post/put/patch call

System.***out***.println("\*\*\*\*Printing request body\*\*\*\*\*");

List<String> lst = **new** ArrayList();

lst.add("Calculus");

lst.add("software Testing");

student stu = **new** student();

stu.setCourses(lst);

stu.setEmail("jansilakshman@gmail.com");

stu.setFirstName("Jansi L");

stu.setLastName("Lakshmanan");

stu.setProgramme(" Software Testing");

*given*().contentType(ContentType.***JSON***).body(stu).log().body().when().post().then().statusCode(201);

@Test

**public** **void** test5() {

// to print out all the request info only if test fails

System.***out***.println("\*\*\*\*Printing failure test case details alone \*\*\*\*\*");

List<String> lst = **new** ArrayList();

lst.add("Calculus");

lst.add("software Testing");

student stu = **new** student();

stu.setCourses(lst);

stu.setEmail("jans@gmail.com");

stu.setFirstName("Jansi L");

stu.setLastName("Lakshmanan");

stu.setProgramme(" Software Testing");

*given*().contentType(ContentType.***JSON***).body(stu).log().ifValidationFails().when().post().then().statusCode(201);

}

note:

\*\*\*🡪.statusCode(201) is used to validate if new user is created successfully or not.is test is passed,then user is created,if failed with assertion error,then user not created.

Error:

java.lang.AssertionError: 1 expectation failed.

Expected status code <201> doesn't match actual status code <500>.

Reason: same user details are given again,so new user with diff id is not created.

Solution:change the unique field and then pass the info,here,unique field is emailed

Logging Response info :

Here,we are logging response information that we get in getcall()

// to Print out response info

@Test

**public** **void** test1() {

System.***out***.println("\*\*\*Print out response headers\*\*\*");

*given*().param("programme", " Software Testing").param("limit", 2).when().get("/list").then().log().headers()

.statusCode(200);

}

@Test

**public** **void** test2() {

// to Print out response info

System.***out***.println("\*\*\*Print out response status line\*\*\*");

*given*().param("programme", " Software Testing").param("limit", 2).when().get("/list").then().log().status()

.statusCode(200);

}

@Test

**public** **void** test3() {

// to Print out response info

System.***out***.println("\*\*\*Print out response body\*\*\*");

*given*().param("programme", " Software Testing").param("limit", 2).when().get("/list").then().log().body()

.statusCode(200);

}

@Test

**public** **void** test4() {

// to Print out response info

System.***out***.println("\*\*\*Print out response info only if there is an error\*\*\*");

*given*().param("programme", " Software Testing").param("limit", -2).when().get("/list").then().log().ifError();

}

Note:

Since,we get response in then() method,here .log() is given after .then() to capture the response info.

.then().log().header()🡪to print response headers

.then().log().body() 🡪to print response body

.then().log(),statusline()🡪to print out response status line.

.then().log().iferror()🡪print response info only if test case fails