# REST API AUTOMATION MARCH 2022 UDEMY

# BASICS:

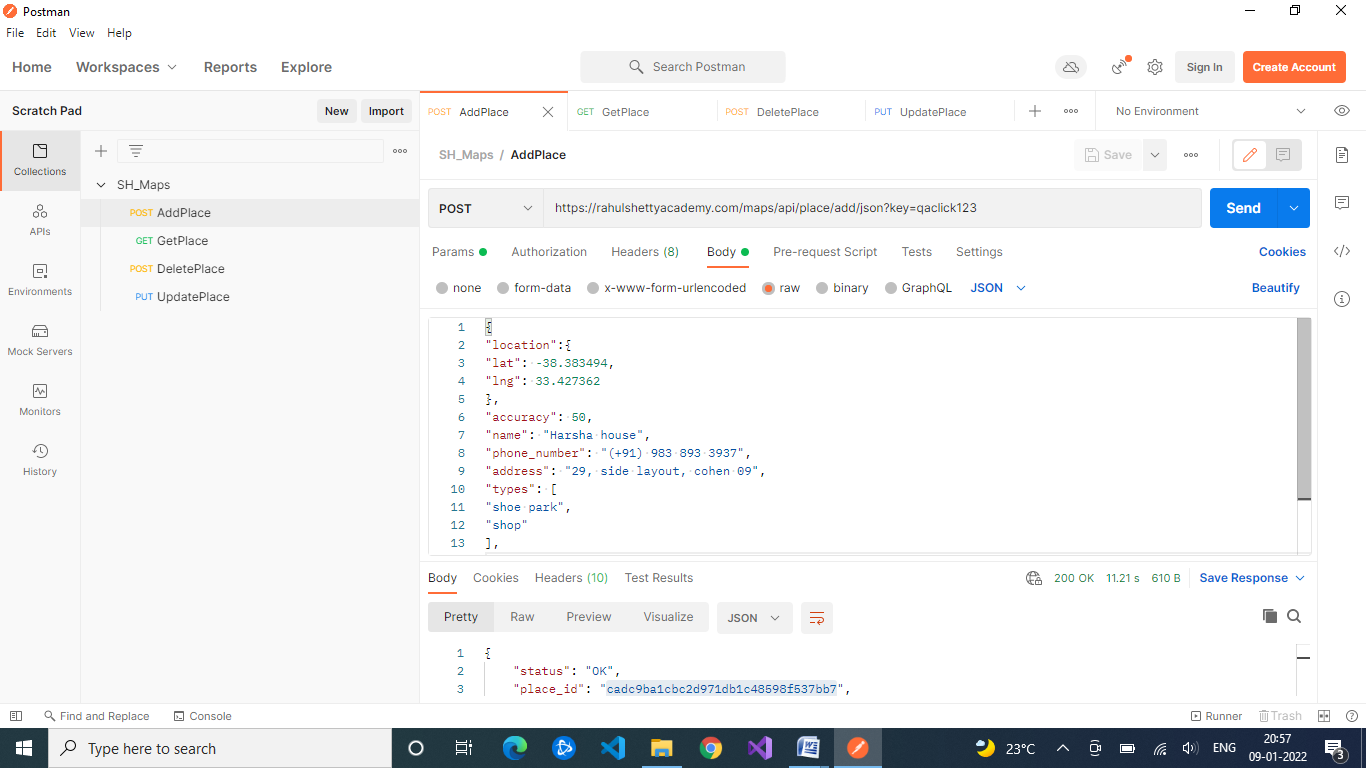
C:\RESTAPIAutomation\DemoProject

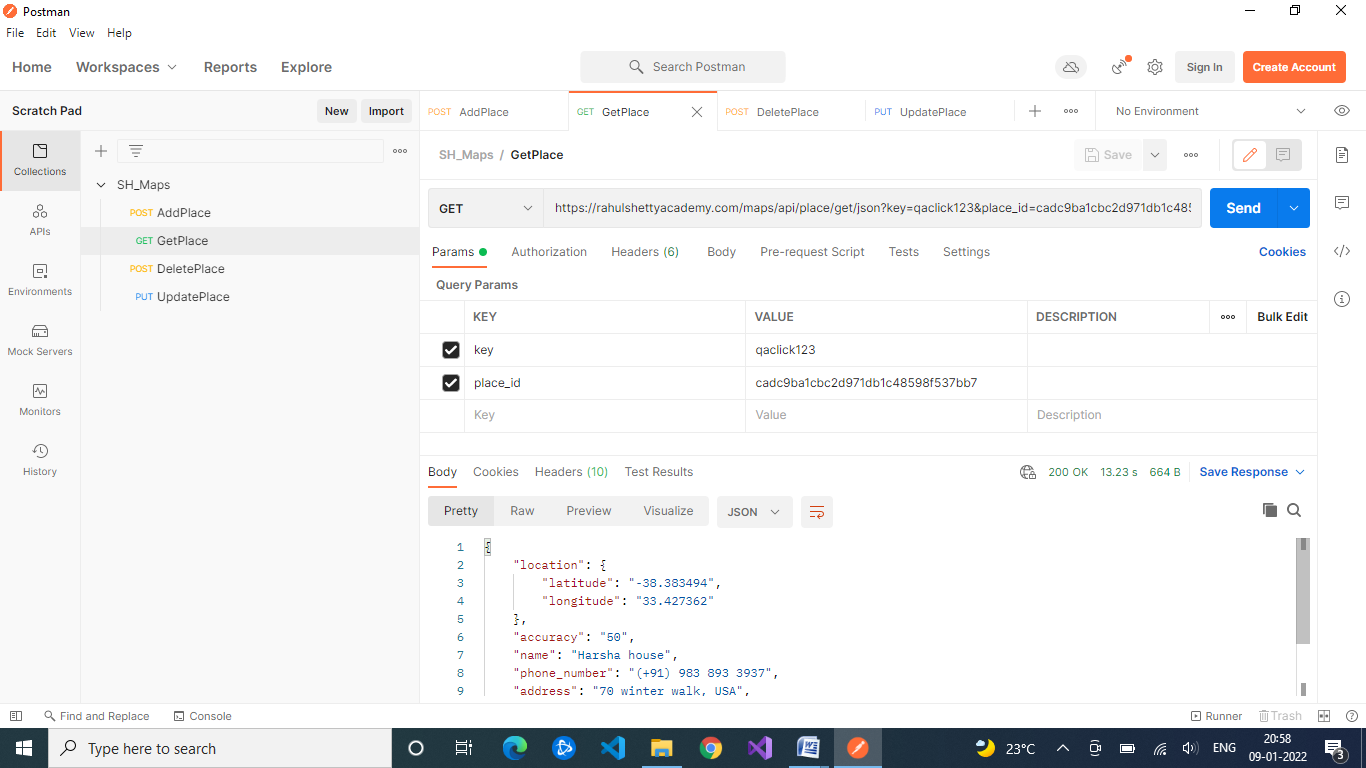
* End point: Address where API is hosted on the Server.
* HTTP methods which are commonly used to communicate with Rest API’s are
* **GET, POST, PUT, and DELETE**
* GET- The GET method is used to extract information from the given server using a given URI. While using GET request, it should only extract data and should have no other effect on the data. No Payload/Body required
* **How to send input data in GET?**  
  Ans: Using Query Parameters
* POST- A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.
* **How to send input data in POST?**  
  Ans: Using Form Parameters /Body Payload
* PUT- Replaces all current representations of the target resource with the uploaded content.
* DELETE- Removes all current representations of the target resource given by a URI.
* **Resources:  
  Resources represent API/Collection which can be accessed from the Server**
* Google.com/maps  
  google.com/search  
  google.com/images
* **Path Parameters:**  
  ***Path parameters*** are variable parts of a URL path. They are typically used to point to a specific resource within a collection, such as a user identified by ID
* <https://www.google.com/Images/1123343>  
  <https://www.google.com/docs/1123343>  
  <https://amazon.com/orders/112>
* <https://www.google.com/search?q=newyork&oq=newyork&aqs=chrome..69i57j0l7.2501j0j7&sourceid=chrome&ie=UTF-8>
* **Query Parameters:**  
  Query Parameter is used to sort/filter the resources.
* Query Parameters are identified with?””
* https://amazon.com/orders?sort\_by=2/20/2020
* **Headers/Cookies**:
* Headers represent the meta-data associated with the API request and response. In layman terms, we were sending Additional details to API to process our request.  
  Example : Authorization details
* **End Point Request URL can be constructed as below**  
  Base URL/resource/(Query/Path)Parameters

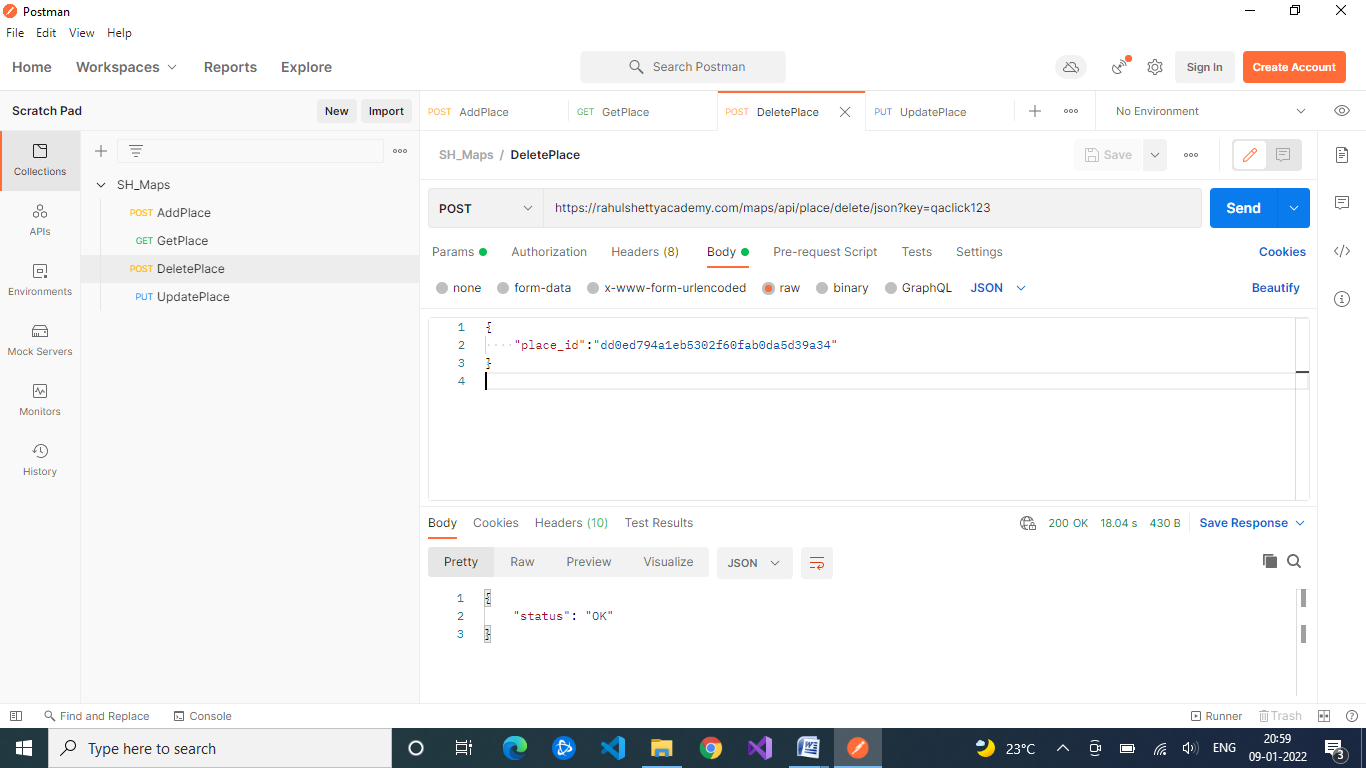
# POSTMAN: POST,GET,DELETE,PUT

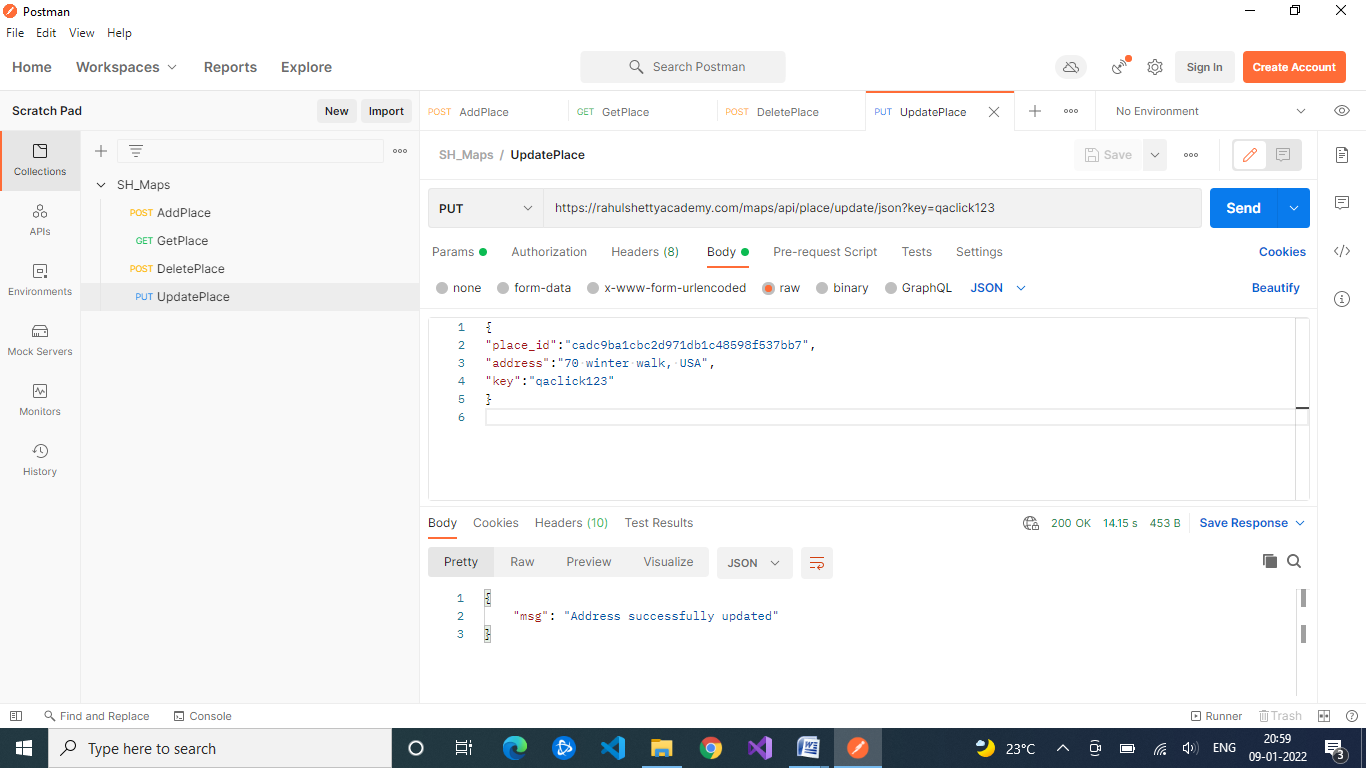
Install postman by going to official website

* Create new collection and provide name for it as SH\_Maps
* Click on view more actions … button and click on Add request
* Provide Base URL and append Resource to it in the table pass query params and in body provide payload and click on send button and observe the response









# Download Java and Eclipse

Download Java and eclipse from following paths:

Place java jdk in c:/programfiles/java and take the path inside till jdk and set env variable system variable

<https://www.eclipse.org/downloads/>

* Open eclipse with workspace C:\RESTAPIAutomation and create demoproject java project
* Right click on project and configure-> convert to maven
* Include <dependencies> tag in pom.xml and take dependencies from maven repository
* Search with restassured, testng, hamcrest keywords and take latest and add to pom.xml
* <https://mvnrepository.com/artifact/org.hamcrest/hamcrest/2.2>
* Project-> build automatically ticked

# POST Request Automation:

Right click on src folder and create a new class with name Basics.java and include following:

package restapiautomation;

import io.restassured.RestAssured;

import static io.restassured.RestAssured.\*;

public class Basics {

public static void main(String[] args) {

// validate if Add Place API is workimg as expected

//given - all input details

//when - Submit the API -resource,http method

//Then - validate the response

RestAssured.baseURI= "https://rahulshettyacademy.com";

given().log().all().queryParam("key", "qaclick123").header("Content-Type","application/json")

.body("{\r\n"

+ "\"location\":{\r\n"

+ "\"lat\": -38.383494,\r\n"

+ "\"lng\": 33.427362\r\n"

+ "},\r\n"

+ "\"accuracy\": 50,\r\n"

+ "\"name\": \"Shree house\",\r\n"

+ "\"phone\_number\": \"(+91) 983 893 3937\",\r\n"

+ "\"address\": \"29, side layout, cohen 09\",\r\n"

+ "\"types\": [\r\n"

+ "\"shoe park\",\r\n"

+ "\"shop\"\r\n"

+ "],\r\n"

+ "\"website\": \"http://google.com\",\r\n"

+ "\"language\": \"French-IN\"\r\n"

+ "}\r\n"

+ "").when().post("maps/api/place/add/json")

.then().log().all().assertThat().statusCode(200);

}

}

# SECTION 5 : POST, PUT and GET automation:

package restapiautomation;

import io.restassured.RestAssured;

import io.restassured.path.json.JsonPath;

import static io.restassured.RestAssured.\*;

import static org.hamcrest.Matchers.\*;

import org.testng.Assert;

import files.Payload;

import files.ReUsableMethods;

public class Basics {

public static void main(String[] args) {

// validate adding place then update place using place id and then get place id and assert it with actual

//given - all input details

//when - Submit the API -resource,http method

//Then - validate the response

//ADD PLACE

RestAssured.*baseURI*= "https://rahulshettyacademy.com";

String response = *given*().log().all().queryParam("key", "qaclick123").header("Content-Type","application/json")

.body(Payload.*AddPlace*()).when().post("maps/api/place/add/json")

.then().assertThat().statusCode(200).body("scope", *equalTo*("APP"))

.header("server", "Apache/2.4.18 (Ubuntu)").extract().response().asString();

System.***out***.println(response);

JsonPath js = new JsonPath(response);

String placeId = js.getString("place\_id");

System.***out***.println(placeId);

//UPDATE PLACE

String newAddress = "Winter Walk, Africa";

*given*().log().all().queryParam("key","qaclick123").header("Content-Type","application/json")

.body("{\r\n"

+ "\"place\_id\":\""+placeId+"\",\r\n"

+ "\"address\":\""+newAddress+"\",\r\n"

+ "\"key\":\"qaclick123\"\r\n"

+ "}\r\n"

+ "")

.when().put("maps/api/place/update/json")

.then().assertThat().log().all().statusCode(200).body("msg", *equalTo*("Address successfully updated"));

//GET PLACE

String getPlaceResponse = *given*().log().all().queryParam("key", "qaclick123")

.queryParam("place\_id", placeId)

.when().get("maps/api/place/get/json")

.then().assertThat().log().all().statusCode(200).extract().response().asString();

JsonPath js1 = ReUsableMethods.*rawToJson*(getPlaceResponse);

String actualAddress = js1.getString("address");

System.***out***.println(actualAddress);

Assert.*assertEquals*(actualAddress, newAddress);

}

}

Payload.java:

package files;

public class Payload {

public static String AddPlace() {

return "{\r\n"

+ "\"location\":{\r\n"

+ "\"lat\": -38.383494,\r\n"

+ "\"lng\": 33.427362\r\n"

+ "},\r\n"

+ "\"accuracy\": 50,\r\n"

+ "\"name\": \"Harsha New house\",\r\n"

+ "\"phone\_number\": \"(+91) 983 893 3937\",\r\n"

+ "\"address\": \"29, side layout, cohen 09\",\r\n"

+ "\"types\": [\r\n"

+ "\"shoe park\",\r\n"

+ "\"shop\"\r\n"

+ "],\r\n"

+ "\"website\": \"http://google.com\",\r\n"

+ "\"language\": \"French-IN\"\r\n"

+ "}";

}

}

ReUsableMethods.java:

package files;

import io.restassured.path.json.JsonPath;

public class ReUsableMethods {

public static JsonPath rawToJson(String response)

{

JsonPath js1 = new JsonPath(response);

return js1;

}

}

# SECTION 6: Nested Json

ComplexJsonParse.java:

package restapiautomation;

import files.Payload;

import io.restassured.path.json.JsonPath;

public class ComplexJsonParse {

/\*TEST CASES:

\* 1. Print No of courses returned by API

\* 2.Print Purchase Amount

\* 3. Print Title of the first course

\* 4. Print All course titles and their respective Prices

\* 5. Print no of copies sold by RPA Course

\* 6. Verify if Sum of all Course prices matches with Purchase Amount

\*/

public static void main(String[] args)

{

JsonPath js = new JsonPath(Payload.*CoursePrice*());

//1.Print No of courses returned by API

System.***out***.println("1.Print No of courses returned by API");

int coursesCount = js.getInt("courses.size()");

System.***out***.println(coursesCount);

//2.Print Purchase Amount

System.***out***.println("2.Print Purchase Amount");

int totalAmount = js.getInt("dashboard.purchaseAmount");

System.***out***.println(totalAmount);

//3.Print Title of the first course

System.***out***.println("3.Print Title of the first course");

String titleFirstCourse = js.getString("courses[0].title");

System.***out***.println(titleFirstCourse);

//4. Print All course titles and their respective Prices

System.***out***.println("4. Print All course titles and their respective Prices");

for(int i=0;i<coursesCount;i++)

{

String courseTitles = js.get("courses["+i+"].title");

System.***out***.println(courseTitles);

System.***out***.println(js.get("courses["+i+"].price").toString());

}

//5. Print no of copies sold by RPA Course

System.***out***.println("5. Print no of copies sold by RPA Course");

for(int i=0;i<coursesCount;i++)

{

String courseTitles = js.get("courses["+i+"].title");

if(courseTitles.equalsIgnoreCase("RPA"))

{

int copies = js.get("courses["+i+"].copies");

System.***out***.println(copies);

break;

}

}

}

}

SumValidation.java:

package restapiautomation;

import org.testng.Assert;

import org.testng.annotations.Test;

import files.Payload;

import io.restassured.path.json.JsonPath;

public class SumValidation {

*@Test*

public void sumOfCourses()

{

//6. Verify if Sum of all Course prices matches with Purchase Amount

System.***out***.println("6. Verify if Sum of all Course prices matches with Purchase Amount");

int sum = 0;

JsonPath js=new JsonPath(Payload.*CoursePrice*());

int count= js.getInt("courses.size()");

for(int i=0;i<count;i++)

{

int price=js.getInt("courses["+i+"].price");

int copies=js.getInt("courses["+i+"].copies");

int amount = price \* copies;

System.***out***.println(amount);

sum = sum + amount;

}

System.***out***.println(sum);

int purchaseAmount =js.getInt("dashboard.purchaseAmount");

Assert.*assertEquals*(sum, purchaseAmount);

}

}

# SECTION 7: Parameterization

DynamicJson.java:

package files;

import io.restassured.RestAssured;

import io.restassured.path.json.JsonPath;

import static io.restassured.RestAssured.\*;

import org.testng.annotations.DataProvider;

import org.testng.annotations.Test;

public class DynamicJson {

@Test(dataProvider="BooksData")

public void addBook(String isbn, String aisle)

{

RestAssured.baseURI="http://216.10.245.166";

String response = given().header("Content-Type","application/json")

.body(Payload.Addbook(isbn,aisle))

.when().post("/Library/Addbook.php")

.then().log().all().assertThat().statusCode(200)

.extract().response().asString();

JsonPath js = ReUsableMethods.rawToJson(response);

String id = js.get("ID");

System.out.println(id);

}

@DataProvider(name="BooksData")

public Object[][] getData()

{

return new Object[][] {{"first","101"},{"second","102"},{"third","103"}};

}

}

Payload.java:

public static String Addbook(String isbn, String aisle)

{

String payload = "{\r\n"

+ " \"name\":\"Learn Appium Automation with Java\",\r\n"

+ " \"isbn\":\""+isbn+"\",\r\n"

+ " \"aisle\":\""+aisle+"\",\r\n"

+ " \"author\":\"John Foe\"\r\n"

+ "}";

return payload;

}

Handle Static JSON from External source:

Basics.java:

package restapiautomation;

import io.restassured.RestAssured;

import io.restassured.path.json.JsonPath;

import static io.restassured.RestAssured.\*;

import static org.hamcrest.Matchers.\*;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Paths;

import org.testng.Assert;

import files.Payload;

import files.ReUsableMethods;

public class Basics {

public static void main(String[] args) throws IOException {

// validate adding place then update place using place id and then get place id and assert it with actual

//given - all input details

//when - Submit the API -resource,http method

//Then - validate the response

//ADD PLACE

RestAssured.baseURI= "https://rahulshettyacademy.com";

String response = given().log().all().queryParam("key", "qaclick123").header("Content-Type","application/json")

//one way by passing Json file to body from our system

.body(new String(Files.readAllBytes(Paths.get("C:\\RESTAPIAutomation\\addPlace.json")))).when().post("maps/api/place/add/json")

//other way by passing payload file and calling method from it to body

//.body(Payload.AddPlace()).when().post("maps/api/place/add/json")

.then().assertThat().statusCode(200).body("scope", equalTo("APP"))

.header("server", "Apache/2.4.18 (Ubuntu)").extract().response().asString();

System.out.println(response);

JsonPath js = new JsonPath(response);

String placeId = js.getString("place\_id");

System.out.println(placeId);

//UPDATE PLACE

String newAddress = "Winter Walk, Africa";

given().log().all().queryParam("key","qaclick123").header("Content-Type","application/json")

.body("{\r\n"

+ "\"place\_id\":\""+placeId+"\",\r\n"

+ "\"address\":\""+newAddress+"\",\r\n"

+ "\"key\":\"qaclick123\"\r\n"

+ "}\r\n"

+ "")

.when().put("maps/api/place/update/json")

.then().assertThat().log().all().statusCode(200).body("msg", equalTo("Address successfully updated"));

//GET PLACE

String getPlaceResponse = given().log().all().queryParam("key", "qaclick123")

.queryParam("place\_id", placeId)

.when().get("maps/api/place/get/json")

.then().assertThat().log().all().statusCode(200).extract().response().asString();

JsonPath js1 = ReUsableMethods.rawToJson(getPlaceResponse);

String actualAddress = js1.getString("address");

System.out.println(actualAddress);

Assert.assertEquals(actualAddress, newAddress);

}

}

# SECTION 8: Jira API

Download JIRA:

Jira Trial licence code:

B0LE-N6WI-ZHAD-HHUV

Jira app: username: harsha, password: shree, email: [prince.harsha6@gmail.com](mailto:prince.harsha6@gmail.com)

C:\Users\admin\Atlassian\Jira

Double click on start\_service to start service and then open localhost:8080 in browser to start JIRA

JIRA APIs link:

<https://docs.atlassian.com/jira-software/REST/7.3.1/>

POSTMAN:

**JIRAAUTH**:

POST request, Url: <http://localhost:8080/rest/auth/1/session>

Body: raw json type:

{ "username":"harsha","password":"shree7117"}

Response:

{

    "session": {

        "name": "JSESSIONID",

        "value": "50C84CCE0CA7345765F1F9B6B29E48FB"

    },

    "loginInfo": {

        "loginCount": 3,

        "previousLoginTime": "2022-03-10T20:21:34.140+0530"

    }

}

**CREATEISSUE**: Get session id from above and use for cookie value

POST request, Url: <http://localhost:8080/rest/api/2/issue>

Header: Key: Cookie, Value: JSESSIONID=50C84CCE0CA7345765F1F9B6B29E48FB

Body: raw json type:

{

    "fields":{

        "project":

        {

            "key":"RSA"

        },

        "summary":"Harsha Credit card defect",

        "description": "Creating my first bug",

        "issuetype":{

            "name":"Bug"

        }

    }

}

Response:

{

    "id": "10000",

    "key": "RSA-1",

    "self": "http://localhost:8080/rest/api/2/issue/10000"

}