# # **REST\_ASSURED\_API\_TESTING by Yk**

## **Section 1**

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

## **Section 2**

Java Basics

## **Section 3**

Postman

## **Section 4: Rest Assured Setup**

### What is Rest Assured?

Rěst-Assured is a Java-based library|that is used to test RESTful

Web Services/API's

### 3 Simple Steps to Setup Rest Assured Project -

. Install Java and set in in System Variables

. Install Eclipse and Create Java Maven Project

. Add Dependency  
 1. Rest Assured  
 2. TestNG  
 3. Hamcrest  
 4. JSON Simple

. Configure Rest Assured Jars into Project

Example:

import io.restassured.RestAssured;

import static io.restassured.RestAssured.\*;

public class Basics {

public static void main(String[] args) {

// Valdiate if Add PlaceAPI is working as expected

//given - all input details

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

//Then - valdiate 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\": \"Yashwant Kumar\",\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: Validating the Rest API Responses**

import io.restassured.RestAssured;

import io.restassured.path.json.JsonPath;

import junit.framework.Assert;

import static io.restassured.RestAssured.\*;

import static org.hamcrest.Matchers.\*;

import static org.testng.Assert.***ARRAY\_MISMATCH\_TEMPLATE***;

import files.Payload;

import files.ReUsableMethods;

public class Basics {

public static void main(String[] args) {

// Valdiate if Add PlaceAPI is working as expected

//Add Place -> Update Place with address -> Get Place to validate id new address us present in the response

//given - all input details

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

//Then - valdiate the response

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

//post

String response =

*given*()

.log().all()

.queryParam("key","qaclick123")

.header("Content-Type","application/json")

.body(Payload.*AddPlace*())

.when()

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

.then()

//.log().all()

.assertThat().statusCode(200)

.body("scope",*equalTo*("APP"))

//.header("server", "nginx");

.extract().asString();

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

JsonPath js = new JsonPath(response); // for parsing JSON

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

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

//update place

String newAddress= "Bhilai CG";

*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()

.log().all()

.assertThat().statusCode(200)

.body("msg",*equalTo*("Address successfully updated"));

//get

String getPlaceResponse = *given*()

.log().all()

.queryParam("key","qaclick123")

.queryParam("place\_id",placeId )

.when()

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

.then()

.log().all()

.assertThat().statusCode(200)

.extract().asString();

JsonPath js1 = ReUsableMethods.rawToJson(getPlaceResponse);

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

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

Assert.*assertEquals*(actualAddress, newAddress);

}

}

--------------------------------------------------------------------------------------------------------------------------------------

package files;

import io.restassured.path.json.JsonPath;

public class ReUsableMethods {

public static JsonPath rawToJson(String response)

{

JsonPath js=new JsonPath(response);

return js;

}

}

--------------------------------------------------------------------------------------------------------------------------------------

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\": \"Yashwant\",\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"

+ "";

}

}

## Section 6: Diving in Depth Automating REST API

{

"dashboard": {

"purchaseAmount": 910,

"website": "rahulshettyacademy.com"

},

"courses": [

{

"title": "Selenium Python",

"price": 50,

"copies": 6

},

{

"title": "Cypress",

"price": 40,

"copies": 4

},

{

"title": "RPA",

"price": 45,

"copies": 10

}

]

}

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

Answer:

import files.Payload;

import io.restassured.path.json.JsonPath;

public class ComplexJsonParse {

public static void main(String[] args) {

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

// 1. Print No of courses returned by API

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

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

// 2.Print Purchase Amount

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

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

// 3. Print Title of the first course

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

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

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

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

{

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

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

}

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

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

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

{

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

if (courseTitiles.equalsIgnoreCase("RPA"))

{

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

break;

}

}

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

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

int purchaseAmount=0;

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

{

purchaseAmount += (js.getInt("courses["+i+"].price")\*js.getInt("courses["+i+"].copies"));

}

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

}

}

## **Section 7: Handling Dynamic Json Payloads with Parameterization**

* Advanced Payload Creation Strategies:
* Dynamically build json payload with external data inputs
* Parameterize the API Tests with multiple data sets
* How to send static Json files(payload) directly into Post
* Method of Rest Assured
* Feed Json payload from Using Excel using HashMap

Library API -

Addbook  
120  
Bookname, author name isbn

import org.testng.annotations.DataProvider;

import org.testng.annotations.Test;

import static io.restassured.RestAssured.given;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Paths;

import io.restassured.RestAssured;

import io.restassured.path.json.JsonPath;

import io.restassured.response.Response;

import files.ReusableMethods;

import files.payLoad;

public class DynamicJson {

@Test(dataProvider="BooksData")

public void addBook(String isbn,String aisle)

{

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

Response resp=given().

header("Content-Type","application/json").

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

when().

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

then().assertThat().statusCode(200).

extract().response();

JsonPath js= ReusableMethods.rawToJson(resp);

   String id=js.get("ID");

   System.out.println(id);

   //deleteBOok

}

@DataProvider(name="BooksData")

public Object[][]  getData()

{

//array=collection of elements

//multidimensional array= collection of arrays

return new Object[][] {‌{"ojfwty","9363"},{"cwetee","4253"}, {"okmfet","533"} };

}

}

import static io.restassured.RestAssured.given;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Paths;

import org.testng.annotations.Test;

import files.ReusableMethods;

import files.payLoad;

import io.restassured.RestAssured;

import io.restassured.path.json.JsonPath;

import io.restassured.response.Response;

public class StaticJson {

@Test

public void addBook() throws IOException

{

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

Response resp=given().

header("Content-Type","application/json").

body(GenerateStringFromResource("C:\\Users\\rahul\\Documents\\Addbookdetails.json")).

when().

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

then().assertThat().statusCode(200).

extract().response();

JsonPath js= ReusableMethods.rawToJson(resp);

   String id=js.get("ID");

   System.out.println(id);

   //deleteBOok

}

public static String GenerateStringFromResource(String path) throws IOException {

    return new String(Files.readAllBytes(Paths.get(path)));

}

}

## **Section 8: Automate Jira**

Steps to Automate in Jira Application:

1. Login to Jira to Create Session using Login API

2. Add a comment to existing Issue using Add Comment API

3. Add an attachment to existing Issue using Add Attachment

API

4. Get Issue details and verify if added comment and

attachment exists using Get Issue API

### New Topics which are covered from above examples:

How to Create Session Filter for Authentication in Rest Assured Automation

Introducing Path Parameters and Query Parameters together in Single Test

Sending Files as Attachments using Rest Assured with Multipart method

Parsing complex Json and limiting Json response through Query Parameters

Handling HTTPS Certification validation through Automated Code

Code:

package files;

import static io.restassured.RestAssured.\*;

import java.io.File;

import org.testng.Assert;

import io.restassured.RestAssured;

import io.restassured.filter.session.SessionFilter;

import io.restassured.path.json.JsonPath;

public class JiraTest {

public static void main(String[] args) {

// TODO Auto-generated method stub

RestAssured.baseURI="http://localhost:8080";

//Login Scenario

SessionFilter session=new SessionFilter();

String response=given().relaxedHTTPSValidation().header("Content-Type","application/json").body("{\r\n" +

"    \"username\": \"RahulShetty\",\r\n" +

"    \"password\": \"XXXX11\"\r\n" +

"}").log().all().filter(session).when().post("/rest/auth/1/session").then().log().all().extract().response().asString();

String expectedMessage ="Hi How are you?";

//Add comment

String addCommentResponse = given().pathParam("key", "10101").log().all().header("Content-Type","application/json").body("{\r\n" +

"    \"body\": \""+expectedMessage+"\",\r\n" +

"    \"visibility\": {\r\n" +

"        \"type\": \"role\",\r\n" +

"        \"value\": \"Administrators\"\r\n" +

"    }\r\n" +

"}").filter(session).when().post("/rest/api/2/issue/{key}/comment").then().log().all()

.assertThat().statusCode(201).extract().response().asString();

JsonPath js=new JsonPath(addCommentResponse);

String commentId= js.getString("id");

//Add Attachment

given().header("X-Atlassian-Token","no-check").filter(session).pathParam("key", "10101")

.header("Content-Type","multipart/form-data")

.multiPart("file",new File("jira.txt")).when().

post("rest/api/2/issue/{key}/attachments").then().log().all().assertThat().statusCode(200);

//Get Issue

String issueDetails=given().filter(session).pathParam("key", "10101")

.queryParam("fields", "comment")

.log().all().when().get("/rest/api/2/issue/{key}").then()

.log().all().extract().response().asString();

System.out.println(issueDetails);

JsonPath js1 =new JsonPath(issueDetails);

int commentsCount=js1.getInt("fields.comment.comments.size()");

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

{

String commentIdIssue =js1.get("fields.comment.comments["+i+"].id").toString();

if (commentIdIssue.equalsIgnoreCase(commentId))

{

String message= js1.get("fields.comment.comments["+i+"].body").toString();

System.out.println(message);

Assert.assertEquals(message, expectedMessage);

}

}

}

}

## **Section 9: Handling OAuth 2.0 Authorization Grant type - Client Credentials**

import io.restassured.RestAssured;

import io.restassured.path.json.JsonPath;

import static io.restassured.RestAssured.\*;

public class oAuthTest {

public static void main(String[] args) {

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

String response = *given*()

.relaxedHTTPSValidation()

.formParam("client\_id", "692183103107-p0m7ent2hk7suguv4vq22hjcfhcr43pj.apps.googleusercontent.com")

.formParam("client\_secret","erZOWM9g3UtwNRj340YYaK\_W" )

.formParam("grant\_type","client\_credentials")

.formParam("scope", "trust")

.when()

.log().all()

.post("/oauth2/resourceOwner/token").asString();

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

JsonPath js = new JsonPath(response);

String accessToken = js.get("access\_token");

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

String response2 = *given*()

.queryParam("access\_token", accessToken)

.log().all()

.when()

.get("/getCourseDetails?access\_token="+accessToken+"")

.asString();

System.***out***.println(response2);

}

}

## **Section 10: Deserialization using POJO classes with Rest Assured**

Serialization & Deserialization of Request/Responses

with POJO classes

Serialization in Rest Assured context is a process of converting a Java object into Request

body (Payload)

Rest Assured also Supports deserialization by converting Response body back to Java object

### Advantages:

Easy to parse and extract response (Json/XML) values if they are wrapped as Java object.

User friendly Methods can be created which makes code more readable.

### Design Approach:

Java object is constructed with the support of POJO classes

POJO classes are created based on the request/Response payload.

### What Additional Libraries required?

For JSON you need to have either Jackson, Jackson2, Gson or Johnzon in the classpath and

for XML you need JAXB.



import io.restassured.RestAssured;

import pojo.AddPlace;

import pojo.Location;

import static io.restassured.RestAssured.\*;

import java.util.ArrayList;

import java.util.List;

public class SerializeTest {

    public static void main (String agrs[]) {

        AddPlace p = new AddPlace();

        p.setAccuracy(50);

        p.setAddress("29, side layout, cohen 09");

        p.setLanguage("French-IN");

        p.setPhone\_number("(+91) 983 893 3937");

        p.setWebsite("https://rahulshettyacademy.com");

        p.setName("Frontline house");

        List <String> myList = new ArrayList<String>();

        myList.add("shoe park");

        myList.add("shop");

        p.setTypes(myList);

        Location L = new Location();

        L.setLat(-38.383494);

        L.setLng(33.427362);

        p.setLocation(L);

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

        String res = given()

            .queryParam("key",

        "qaclick123")

            .body(p)

        .when()

            .post("/maps/api/place/add/json?key=qaclick123")

        .then()

            .log().all()

            .assertThat().statusCode(200)

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

    }

}

package pojo;

import java.util.List;

public class AddPlace {

    private Location location;

    private int accuracy;

    private String name;

    private String phone\_number;

    private String address;

    private List<String> types;

    private String website;

    private String language;

    public Location getLocation() {

        return location;

    }

    public void setLocation(Location location) {

        this.location = location;

    }

    public int getAccuracy() {

        return accuracy;

    }

    public void setAccuracy(int accuracy) {

        this.accuracy = accuracy;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getPhone\_number() {

        return phone\_number;

    }

    public void setPhone\_number(String phone\_number) {

        this.phone\_number = phone\_number;

    }

    public String getAddress() {

        return address;

    }

    public void setAddress(String address) {

        this.address = address;

    }

    public List<String> getTypes() {

        return types;

    }

    public void setTypes(List<String> types) {

        this.types = types;

    }

    public String getWebsite() {

        return website;

    }

    public void setWebsite(String website) {

        this.website = website;

    }

    public String getLanguage() {

        return language;

    }

    public void setLanguage(String language) {

        this.language = language;

    }

}

-----------------------------------------------------------------------------------------------------

package pojo;

import java.util.List;

public class AddPlace {

    private Location location;

    private int accuracy;

    private String name;

    private String phone\_number;

    private String address;

    private List<String> types;

    private String website;

    private String language;

    public Location getLocation() {

        return location;

    }

    public void setLocation(Location location) {

        this.location = location;

    }

    public int getAccuracy() {

        return accuracy;

    }

    public void setAccuracy(int accuracy) {

        this.accuracy = accuracy;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getPhone\_number() {

        return phone\_number;

    }

    public void setPhone\_number(String phone\_number) {

        this.phone\_number = phone\_number;

    }

    public String getAddress() {

        return address;

    }

    public void setAddress(String address) {

        this.address = address;

    }

    public List<String> getTypes() {

        return types;

    }

    public void setTypes(List<String> types) {

        this.types = types;

    }

    public String getWebsite() {

        return website;

    }

    public void setWebsite(String website) {

        this.website = website;

    }

    public String getLanguage() {

        return language;

    }

    public void setLanguage(String language) {

        this.language = language;

    }

}

package pojo;

public class Location {

private double lat;

private double lng;

public double getLat() {

return lat;

}

public void setLat(double lat) {

this.lat = lat;

}

public double getLng() {

return lng;

}

public void setLng(double lng) {

this.lng = lng;

}

}

## Section 12: Understand Request and Response Spec Builders in Rest Assured

Request and Response Spec Builders :

ADD PLACE :

RestAssured.*baseURI*="XXXX";

Response res=*given*().queryParam("key", "qaclick123").header("Content-Type","application/json")

.body(add\_place\_json)

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

then().assertThat().statusCode(200). contentType("application/json")

extract().response();

GET\_PLACE

RestAssured.*baseURI*="XXXX";

Response res=*given*().queryParam("key", "qaclick123").header("Content-Type","application/json")

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

then().assertThat().statusCode(200).contentType("application/json").extract().response();

DELETE\_PLACE

RestAssured.*baseURI*="XXXX";

Response res=*given*().queryParam("key", "qaclick123").header("Content-Type","application/json")

.body(“delete\_Place\_json”)

.when().post("/maps/api/place/delete/json").

then().assertThat().statusCode(200). contentType("application/json").extract().response();

Build -Request Spec Builder-

req= **new** RequestSpecBuilder().setContentType(ContentType.***JSON***)

.setBaseUri("XXXX")

.addQueryParam("key","qaclick123")

.build();

*given*().spec (req ).body(add\_place\_json) .post(“/maps/api/place/add/json).

Build Response Spec Builder:

**res = new** ResponseSpecBuilder().expectStatusCode(200).expectContentType(ContentType.***JSON***). build();

then().spec(re).extract().response();

Rewriting Test with Request and Response Spec Builder :

*given*().spec(req).body(add\_place\_json).post(“/maps/api/place/add/json).

then().spec(res).extract().response();

import io.restassured.RestAssured;

import io.restassured.builder.RequestSpecBuilder;

import io.restassured.builder.ResponseSpecBuilder;

import io.restassured.http.ContentType;

import io.restassured.response.Response;

import io.restassured.specification.RequestSpecification;

import io.restassured.specification.ResponseSpecification;

import pojo.AddPlace;

import pojo.Location;

import static io.restassured.RestAssured.\*;

import java.util.ArrayList;

import java.util.List;

public class SpecBuilderTest {

public static void main (String agrs[]) {

AddPlace p = new AddPlace();

p.setAccuracy(50);

p.setAddress("29, side layout, cohen 09");

p.setLanguage("French-IN");

p.setPhone\_number("(+91) 983 893 3937");

p.setWebsite("https://rahulshettyacademy.com");

p.setName("Frontline house");

List <String> myList = new ArrayList<String>();

myList.add("shoe park");

myList.add("shop");

p.setTypes(myList);

Location L = new Location();

L.setLat(-38.383494);

L.setLng(33.427362);

p.setLocation(L);

RequestSpecification req = new RequestSpecBuilder().setBaseUri("https://rahulshettyacademy.com").addQueryParam("key", "qaclick123")

.setContentType(*ContentType*.***JSON***).build();

ResponseSpecification resspec = new ResponseSpecBuilder().expectStatusCode(200).expectContentType(*ContentType*.***JSON***).build();

RequestSpecification res = *given*().spec(req).body(p);

Response response = res.when().post("/maps/api/place/add/json?key=qaclick123")

.then().spec(resspec).extract().response();

String responseString = response.asString();

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

}

}

## Section 13: End to End ecommerce API example with Automation Concepts

## Section 14: Cucumber BDD API Framework Development from Scratch – 1

### What is Gherkin?

It is a Business Readable, Domain Specific Language that lets you describe software's behavior.

Example: Pop up messaged is displayed when buttons are clicked and errors are gone

Developer :Pop up message is displayed and errors are gone on button clicked

QA- Pop up message is displayed -only when buttons are clicked and errors are gone

Keywords Used in Cucumber: Scenario, Feature, Feature file, Scenario outline, Step Definition

### Scenarios:

In Cucumber Testcases are represented as Scenarios.

Scenarios contain Steps which are equivalent to test Steps and use the following keywords (Gherkin syntax) to denote them: Given, When, Then, But, and And (case sensitive).

**Given:** Preconditions are mentioned in the Given keyword

**When:** The purpose of the When Steps is to describe the user action.

**Then**: The purpose of Then Steps is to observe the expected output. The observations should be related to the business value/benefit of your Feature description.

Example:  
  
 **Scenario**: Make Minimum Due payment

**Given** user is on Pay credit card page

**When** user fills all details and select Minimum amount option

**And** User clicks on Pay button

**Then** Credit Card confirmation page is displayed

**AND** User clicks on Pay Button

**But** Error message is not displayed

When we specify a business requirement, sometimes there are multiple pre-conditions, user actions, and expected outcomes  
   
we are going to add one more Scenario and will use the And and But keywords:  
  
 And: This is used for statements that are an addition to the previous Steps and represent

positive statements.

But: This is used for statements that are an addition to previous Steps and represent negative statements.

### Feature and Feature File:

Feature represents Business requirement.

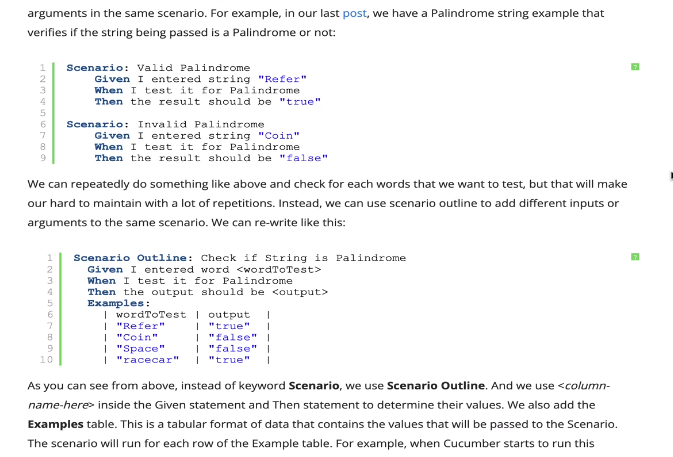
Feature File acts as a Test Suite which consists of all Scenarios.

In Cucumber, Feature files contain Scenarios. We can simply create feature file with. feature extension

Scenarios belonging to specific area of Application will be grouped into one Feature file

The text that immediately follows the Feature keyword, and is in the same line, is the Title of the Feature file

Feature file should contain either Scenario or Scenario Outline. The naming conventions for Feature files should be lowercase with. feature extension



### How to run project from cmd

* mvn compile - for compiling
* mvn test - to execute project
* mvn verify - to execute project and generate report
* mvn test verify -Dcucumber.options="--tags @Regression"  
  for running particular tag
* maven cucumber reporting – for html report

**Jenkins**

* java -jar jenkins.war -- --httpPort=8080
* java -jar jenkins.war --enable-future-java

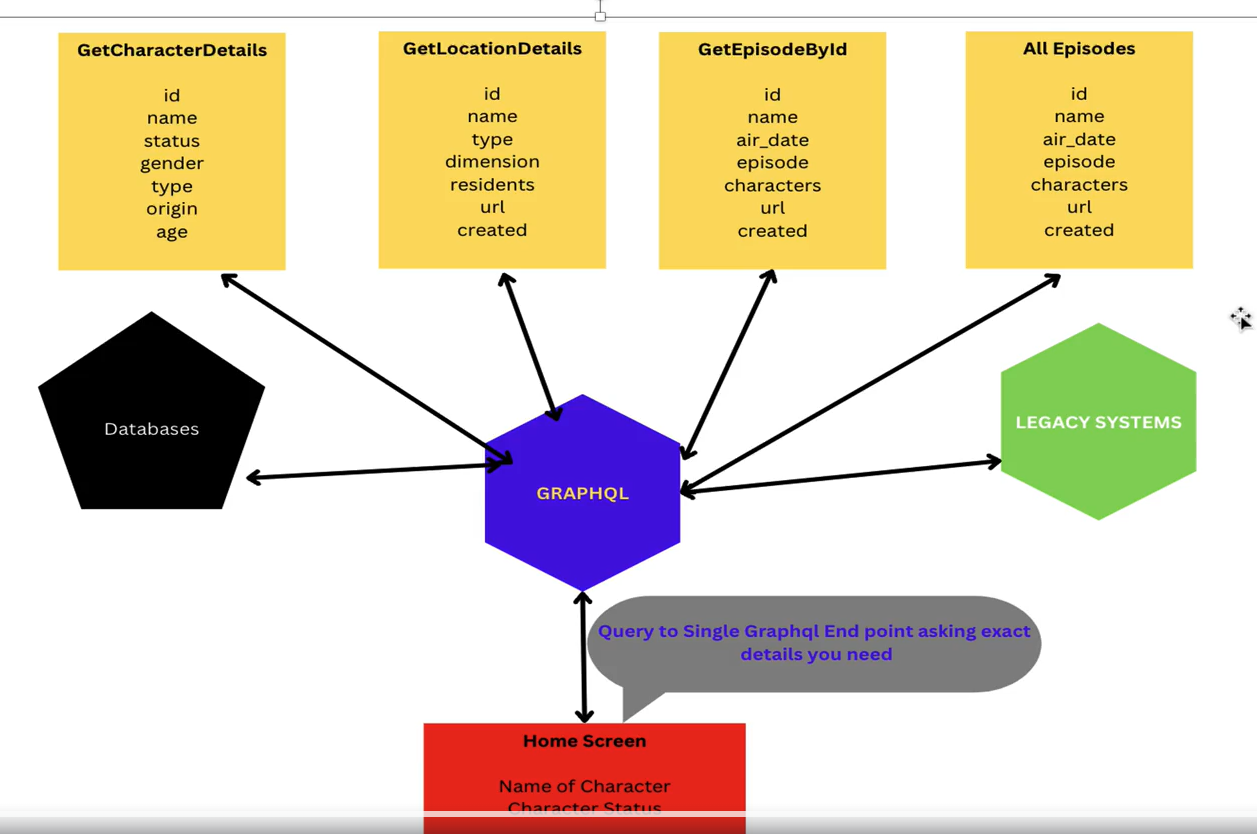
## Section 16: Learn GraphQL from Scratch and Testing with Rest Assured

### What is GraphQL?

GraphQL is a query language and server-side runtime for fulfilling those queries on

your existing data. GraphQL isn't tied to any specific database or storage engine

and is instead backed by your existing code and data



url: <https://rahulshettyacademy.com/gq/graphql>  
(Server explorer link)

## Section 18: GIT

Example:-

X - Writes some code and creates testcases "A" - EST //2nd component ( 5-10) - GITHUB

X1B

X2B

X3A

Y - IST -- "A"- // 1- 5lines (first component)

Tool Plugin:-

Tortoise Git-  
Eclipse -  
Intellij

GIT Basic Command  
**URL:** [**https://confluence.atlassian.com/bitbucketserver/basic-git-commands-776639767.html**](https://confluence.atlassian.com/bitbucketserver/basic-git-commands-776639767.html)

You have a code

Initialize it as Git repository - git init

Staging - Commit

GITHUB

### Basic Git commands

|  |  |  |
| --- | --- | --- |
| **Git task** | **Notes** | **Git commands** |
| [**Tell Git who you are**](https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-config) | **Configure the author name and email address to be used with your commits.**  **Note that Git**[**strips some characters**](http://stackoverflow.com/questions/26159274/is-it-possible-to-have-a-trailing-period-in-user-name-in-git/26219423#26219423)**(for example trailing periods) from user.name.** | **git config --global user.name "Sam Smith"**  **git config --global user.email sam@example.com** |
| [**Create a new local repository**](https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-init) |  | **git init** |
| [**Check out a repository**](https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-clone) | **Create a working copy of a local repository:** | **git clone /path/to/repository** |
| **For a remote server, use:** | **git clone username@host:/path/to/repository** |
| [**Add files**](https://www.atlassian.com/git/tutorials/saving-changes#git-add) | **Add one or more files to staging (index):** | **git add <filename>**  **git add \*** |
| [**Commit**](https://www.atlassian.com/git/tutorials/saving-changes#git-commit) | **Commit changes to head (but not yet to the remote repository):** | **git commit -m "Commit message"** |
| **Commit any files you've added with git add, and also commit any files you've changed since then:** | **git commit -a** |
| [**Push**](https://www.atlassian.com/git/tutorials/syncing#git-push) | **Send changes to the master branch of your remote repository:** | **git push origin master** |
| [**Status**](https://www.atlassian.com/git/tutorials/inspecting-a-repository#git-status) | **List the files you've changed and those you still need to add or commit:** | **git status** |
| [**Connect to a remote repository**](https://www.atlassian.com/git/tutorials/syncing#git-remote) | **If you haven't connected your local repository to a remote server, add the server to be able to push to it:** | **git remote add origin <server>** |
| **List all currently configured remote repositories:** | **git remote -v** |
| [**Branches**](https://www.atlassian.com/git/tutorials/using-branches) | **Create a new branch and switch to it:** | **git checkout -b <branchname>** |
| **Switch from one branch to another:** | **git checkout <branchname>** |
| **List all the branches in your repo, and also tell you what branch you're currently in:** | **git branch** |
| **Delete the feature branch:** | **git branch -d <branchname>** |
| **Push the branch to your remote repository, so others can use it:** | **git push origin <branchname>** |
| **Push all branches to your remote repository:** | **git push --all origin** |
| **Delete a branch on your remote repository:** | **git push origin :<branchname>** |
| [**Update from the remote repository**](https://www.atlassian.com/git/tutorials/syncing) | **Fetch and merge changes on the remote server to your working directory:** | **git pull** |
| **To merge a different branch into your active branch:** | **git merge <branchname>** |
| **View all the merge conflicts:**  **View the conflicts against the base file:**  **Preview changes, before merging:** | **git diff**  **git diff --base <filename>**  **git diff <sourcebranch> <targetbranch>** |
| **After you have manually resolved any conflicts, you mark the changed file:** | **git add <filename>** |
| **Tags** | **You can use tagging to mark a significant changeset, such as a release:** | **git tag 1.0.0 <commitID>** |
| **CommitId is the leading characters of the changeset ID, up to 10, but must be unique. Get the ID using:** | **git log** |
| **Push all tags to remote repository:** | **git push --tags origin** |
| [**Undo local changes**](https://www.atlassian.com/git/tutorials/undoing-changes) | **If you mess up, you can replace the changes in your working tree with the last content in head:**  **Changes already added to the index, as well as new files, will be kept.** | **git checkout -- <filename>** |
| **Instead, to drop all your local changes and commits, fetch the latest history from the server and point your local master branch at it, do this:** | **git fetch origin**  **git reset --hard origin/master** |
| **Search** | **Search the working directory for foo():** | **git grep "foo()"** |

### GIT Command for create New Project

* **git config --global user.name “Admin”**
* **git config --global user.email “yashurawte@gmail.com”**
* **git init**
* **git add \* / file name - (add files to stage)**
* **git status**
* **git remote add origin “URL”**
* **git push origin (Login GIT)**
* **git push origin  
  OR  
  git push --set-upstream origin master**

### GIT Command to clone Project and push latest changes

* **cd directory path**
* **git clone “URL” - To download the Project**
* **git add \* - To Push latest changes**
* **git status**
* **git commit -m “msg”**
* **git remote add origin “URL”**
* **git push origin**

### GIT Command to pull latest changes

* **cd DIR**
* **git pull origin master**

### GIT Branching