API (Application Programming Interface)

: Intermediate btn two application where application may differ in their platform or in terms of technology.

→ Its bridge btn 2 application to communicate with each other

→ Its Responsibility Includes fetch the data from backend & display on frontend or take the data from front end & insert into back end

→ If Api present in Internet are called as webservice

~~→ All apis are~~

→ All Web Services are apis but all api are not webservice.

→ Once Api moved to production/Web Environment So it need Network for its operation

---

API - Testing :

→ Type of Testing where API are tested to Determine if they meet expectation for functionality, reliability, performance & Security

→ OR
Testing the application in Source code layer (Business layer)

→ OR
Testing the Interface between two applications.

→ Testing done without browsen.

Rest API methods / http request

get() → To Retrieve the data

post() → To create resource in the Server

put() → To update the resource in the Server

patch() → To update the partial resource in the Server

delete() → To delete the resource inside Server

HTTP / HTTPS : (Hyper text transfer protocol Secure)

These are the protocal useed by Web application (Api) to communicate btn Client & Server

HTTP : data is transmitted through network with original formate (less Secure)

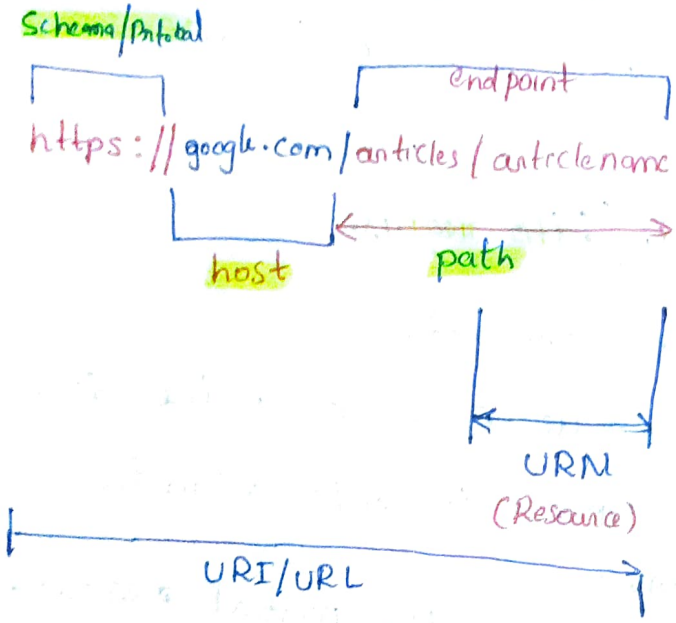
HTTPS : data is transmitted through network in encrypted formate (more Secure)

Terminologies:

URI : Uniform Resource Identifier

URL : Uniform Resource Locator

URN : Uniform Resource Name



Resource : (data/Info)

Resource : whatever we requesting for, that is present in the Server.

endpoints : other than host, rest of the thing
end point is the specific address in URL where Api operation takes place

URI = URL + URN

1) URL : It tells us to where to find Resource & how to access it
→ Include Info like protocal, domain, & path

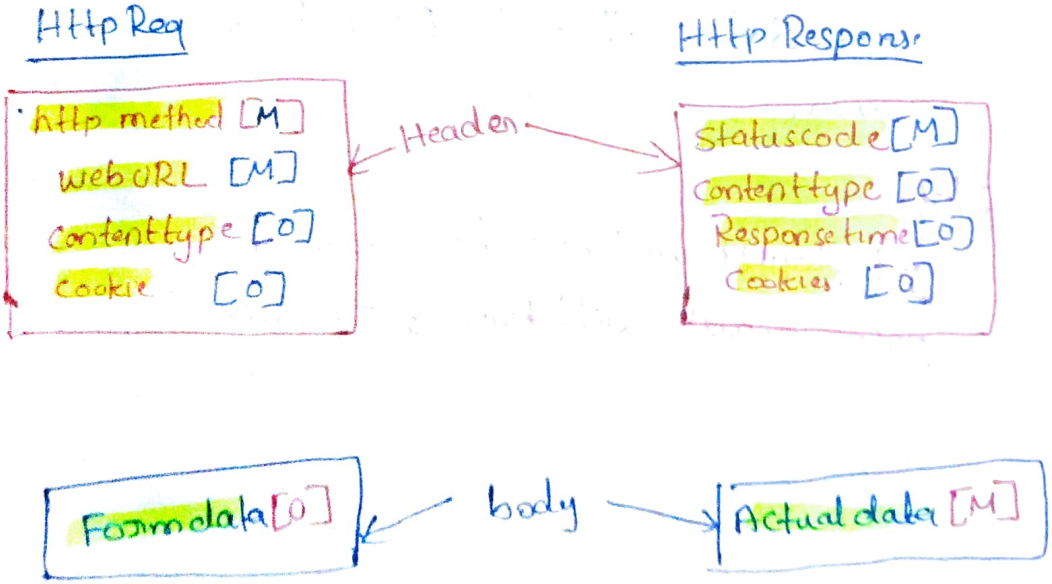
2] URN : tells the Name of Resource (Unique) but doesnot provide location.

3] URI → Its Superset that Include both URL & URNs
→ URI Can be either a URL, URN or both

Payload : ~~XA~~ means body in the
Data which we Sending with Request or data which Coming along with Response
like request payload & response payload

Http structure :

- It consists of header & body



1) Http method : (Request header)

→ It is a mandatory Information present in the header of the http request

→ It is the first element in http request used to specify the type of request which is sent by Client (Browser → Server)

2) Web URL : (Request header)

→ It is the mandatory Information present in the header of the request

→ It is used to Identify the specific web resource inside the web server.

3) Form data : (Request Body)

→ data collected using html form or data collected from browser to Server.

Note :

get Request will not have form data
post request will have form data.

4] Status code : (Response header)

→ It is the mandatory information present in the header of response

→ It will represent the status of request

Status codes:

100 – Series :

1] 100 → Continue

---

200 – Series → Success

1] 200 (ok) → able to read the resource from Server

2] 201 (created) → able to create resource Inside Server

3) 202 (Accepted) → able to accept the permission request inside the Server. (But need more time to process the Request)

4) 203 (Non Authorative Info) → payload has been modified by proxy or Intermediary server.

5] 204 (No content) → got the response from server but no content

---

300-Series – (Redirected) → further action need to be taken by the user in order to full fill the request

400 - Series (Client Side error)

1] 400 (Bad request) → The request could not be Understood by due to malformed Syntax

2] 401 (Unauthorized) → Request is not authorized to access the resource inside the Server

3] 403 (Forbidden) → Server Understood the Requst but not authorized person to access

4] 404 (Not found) → Not able to find the resource Inside the Server

5] 405 (method Not allowed)

429 → To many Requst

6] 409 (Conflict) → Duplicate request

7] 415 (Unsupported media type)

500 - Series (Server Side Error)

1] 500 (Intenal Server Error) → problem from Server Side

2] 502 (Bad Gateway) → proxy/firewall issue in Server Side

3] 503 (Service Unavailable): The Server is currently Unable to handle the request

4] 504 (Gateway Timeout) → Server not respond even waiting for long time

---

Authentication:

Used to check whether User Valid or not

Authorization: Used to check the permission or access to the particular resource in the Server.

Types of ~~Authentication~~ Authentication. Authorization

1] **Basic auth**:
→ Send Request using Username & password

2] **Bearer Token**:
→ Send request using token ID to access api

3] **OAuth 1.0**: Two level authentication
[ConsumerID, Consumer Secrete, access ID, access Secrete]

4] **OAuth 2.0**: One level authentication
[Client ID & Client Secrete]

JSON : (Java script object Notation)

Datatypes:

1] String
2] Number
3] array
4] boolean
5] null
6] Object
7] array of object

Types of Parameter

1] Path Parameter – only get
2] Qury Parameter – only get
3] Form parameter – put, patch, post
4] Param parameter

1] Path parameter:

These are part of end point URL & are used to Identify a specific resource.

ex: http:// example.com/user/{userId}

2] Qury Parameter:

These are the key-value pairs that appended to end of URL using ? & are used to filter the resource

3] Form parameter

→ It is mainly used when we want to send structured data such as form submission or when uploading file

→ Or used to send sensitive data

→ It will be sent with body of request & encoded key-value format.

Ex: multipart/form-data → used to upload file.

4] param parameter:

→ Generic method available in Rest assured

→ Param parameter with get will act like query parameter

→ param parameter with post will act like form param

Request chaining

Fetch the value from one API response & pass same data into another Request is called Request chaining

Rest Assured :

→ Rest assured is an Java based Library that is used to test restful Webservice. this library behaves like a head less Client to access Rest web service

→ It also provide ability to validate the Rest Http response recave from Server.

eg: we can validate Status code, status message, header & even the Body of response.

This makes Rest assured a very flexible library that can be used for testing

Why Rest Assured popular ? / Advantages

1] It can Integrate Seamlessly with existing Java based framework like

1] TestNG, JUNIT, BDD
2] Selenium WebDriver
3] JDBC

2] We can automate E2E busness workflow which Include all layer.

3] It support All http methods

4] Rest Assured provide Inbuilt method to create request header & body

5] Rest Assured provide Inbuilt method to validate response header & body

6] It allows us to manage different types of authentication like., Basic Auth, Bearer token, Oauth 1.0, oauth2.0

7] Open source headless Client

8] Framework Can Integrate with CI/CD

Ways to Handle Post Request

1] Using Json object

2] Using HashMap

3] Using JSON file

4] Using POJO Class

POJO class: (Plane old Java object)

→ class in Java

It is a Java Service Enginee where Rest api properties are difined & ~~working~~ use of and create getter & setter for all the properties

- It will help us to achive Parsing.

What is the Use of RequestSpecBuilder ?

→ It is a class available in the Rest assured using RequestSpecBuilder we can put common configuration which is required for api test script like

→ ContentType
→ Authentication
→ baseURI

Ex: [We can Create in Base Class]

```
private static RequestSpecification requSpec
        =new RequestSpecBuilder()
```

- setBaseUri("http://api.com")
- addHeader("Authorization", "Bearer token")
- addHeader("Content-Type", "application/json")
- build()

In test class In given method have spec(requSpec) → we can use like this

Post Request

```
RestAssured.baseURI = "http://beta...        -internal/component"

Response response = given().
    • body(pojoObj)
    • header("Authorization", "Bearer " + bearerToken)
    • header("Content-Type", ContentType.JSON)
    • when()
    • post("/addmarqui");

response.body(JsonSchemaValidator.matchesJsonSchema(file));

SoftAssert.assertEquals(response.getStatusCode(), 201);

SoftAssert.assertEquals(response.jsonPath().getString("message"),
                        "created");

SoftAssert.assertEquals(response.jsonPath().getString(elements[1].type),
                        "title");

softAssert.assertEquals(response.jsonPath().getString("type"), "standard");
```

Get Request

```
Response res = given()
    .header("Authorization", "Bearer " + bearerToken)
    .when()
    .get("/getmarque");

SoftAssertion.assertEquals(res.getStatusCode(), 200);
```

---

* What all the Info pass in given()

1] Content type
2] Set cookies
3] add param
4] Set header Info etc...
5] Authentication.

} all the prerequisite

when(): Request type we will keep in in when

-> get(), post(), put(), delete

then(): here we will do validation.