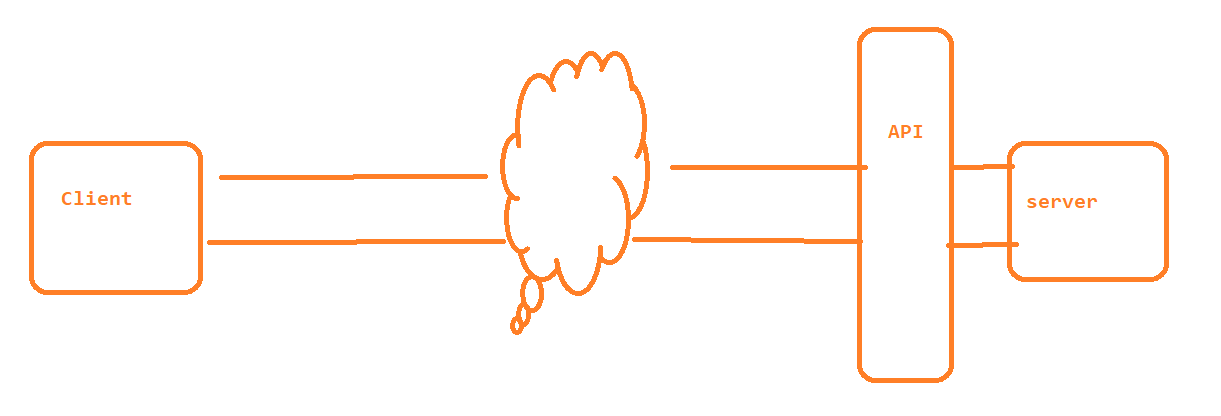
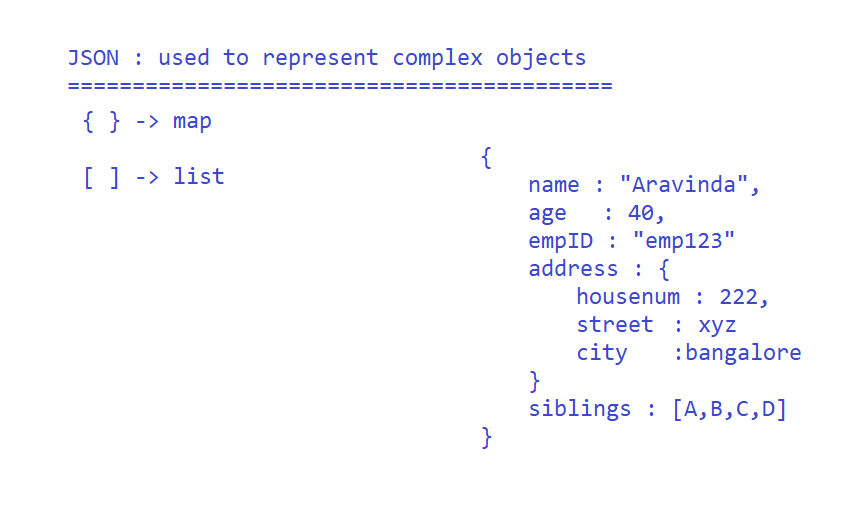
WebServices Introduction

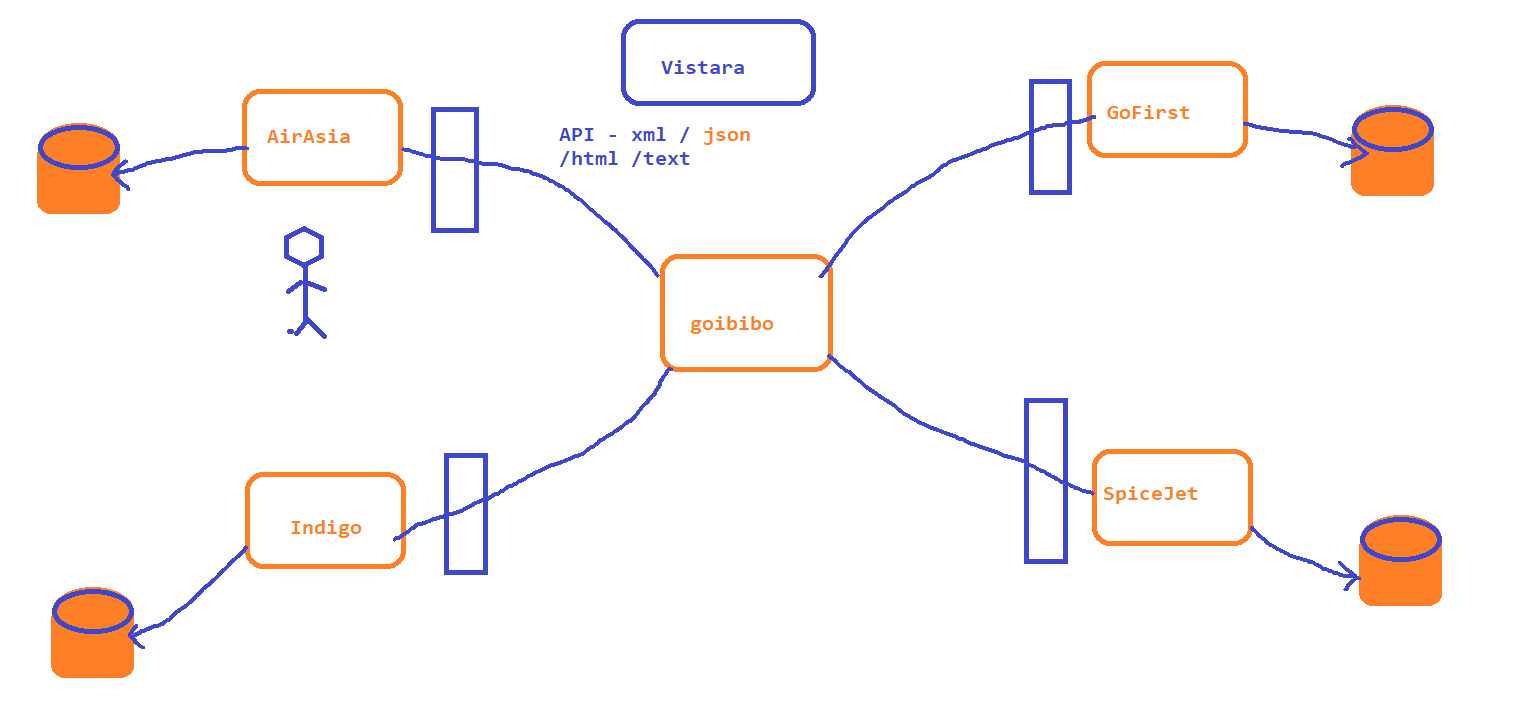


**Services** available over the web to enable the communication between applications over the web.

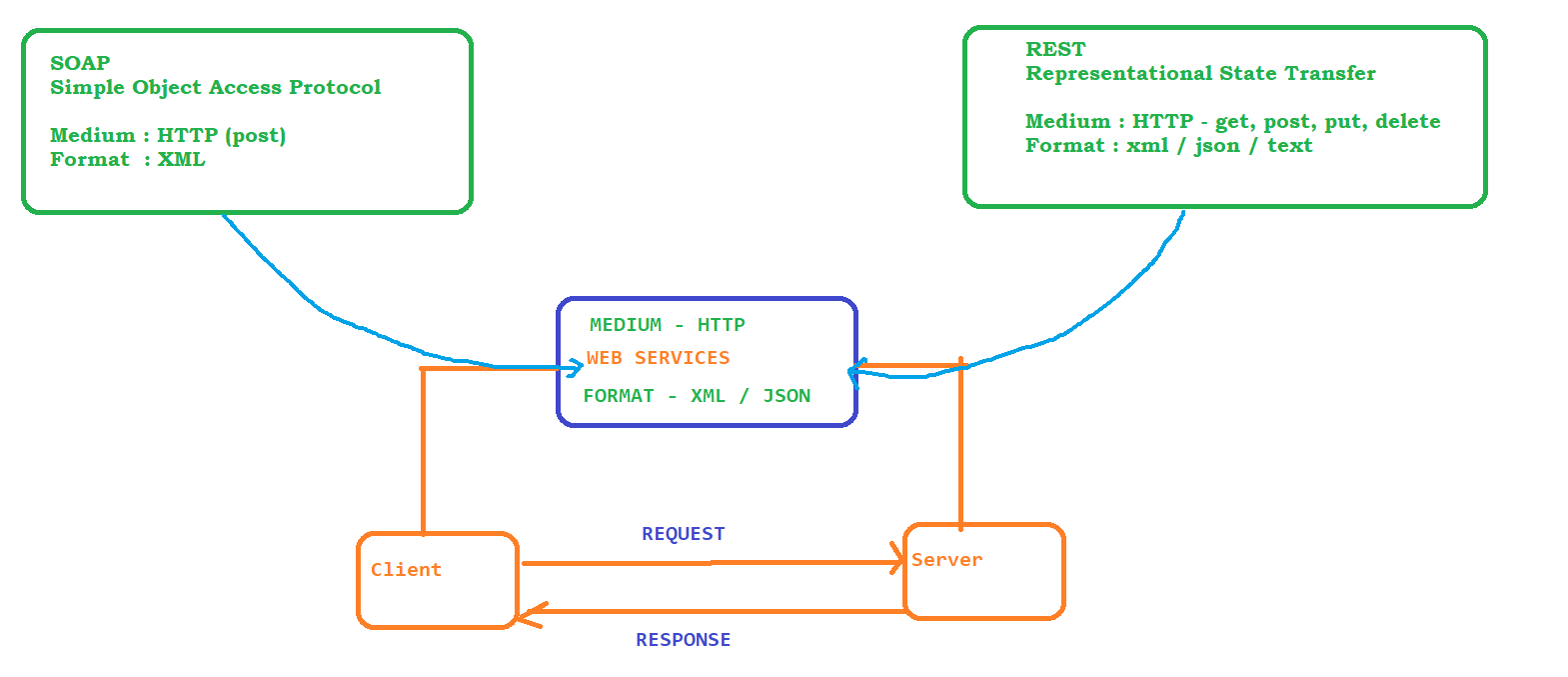
JSON :



How WebServices works ???

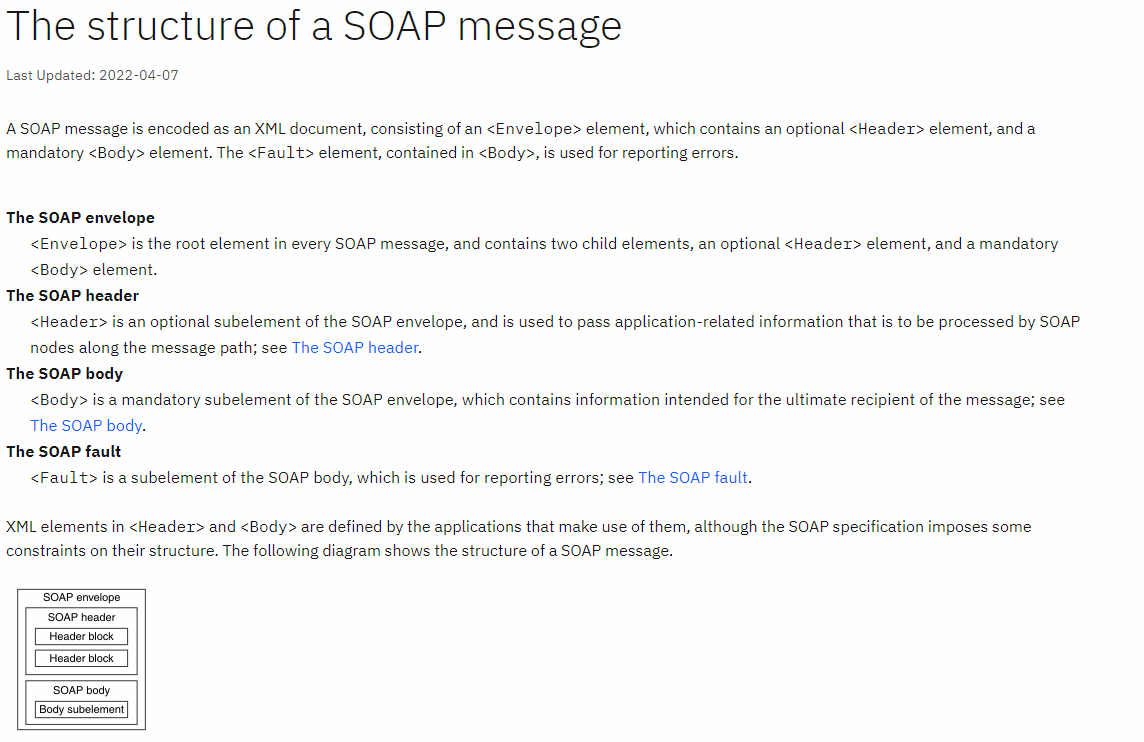


High-level Architecture



SOAP Example : <http://www.dneonline.com/calculator.asmx>

<https://www.ibm.com/docs/en/integration-bus/10.0?topic=soap-structure-message>



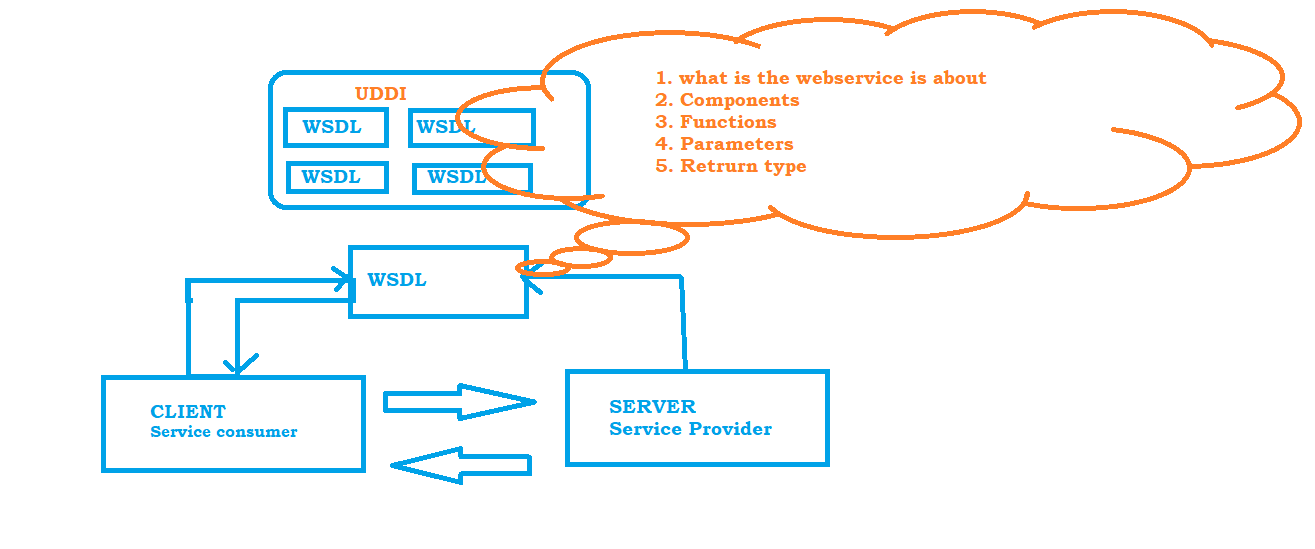
REST Example :

<http://httpbin.org/#/>

WSDL and UDDI

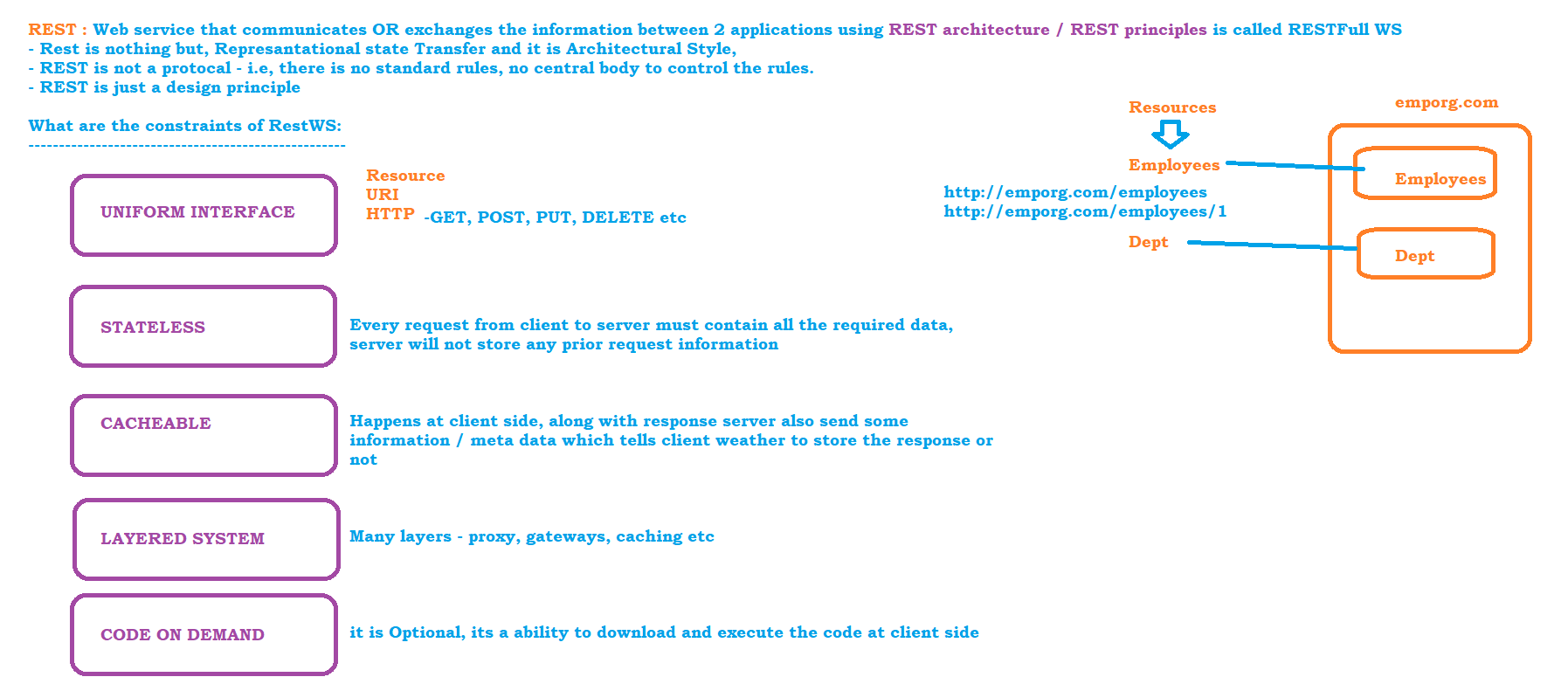
**Web Services Description Language**, Service providers who own the application publish the description about the web services. WSDL is a interface to the WebService which describes all the attributes, functions, components, parameters and return type of the web service

<http://www.dneonline.com/calculator.asmx?WSDL>



**Universal Description, Discovery and integration,**  is a XML Based standard for publishing and finding web services.

REST



PostMan

* Installation :
  + <https://www.postman.com/downloads/>

# Rest API Elements:

HTTP Request:

* Request Line : specifies the method token (GET, PUT, POST...) followed by the Request URI
* Request Header : Optional, contain zero or more request Headers can be passed in any request. this will hold additional information about the request to the server
* Request Body : Optional, its a part of the HTTP Request where additional content to the server is passed

HTTP Response:

* Request Line :
* Headers
* Optional Body
* Status Code :

**Resource :** Any information stored in a Server, which is requested by client

**Resource Identifier :** To identify the resource uniquely we use resource identifier. – this is actually a complete url

**Representation :** The actual data returned by the server in any format – HTML, XML, JSON, TEXT

**URI :**

**SCHEME ://AUTHORITY / PATH or QUERY PARAMETER**

**SCHEME :** Represents which protocol is used – HTTP,HTTPS,FTP,HTTP2

**AUTHORITY :** Domain

**PATH PARAMETER :** represents the resource, we can have periodic / to represent the resource

**QUERY PARAMETER : ?**

**HTTP Methods :**

**GET :** to retrieve the data from the server

**POST :** To insert the data

**DELETE :** To Delete the record

**PUT :** is to update the record

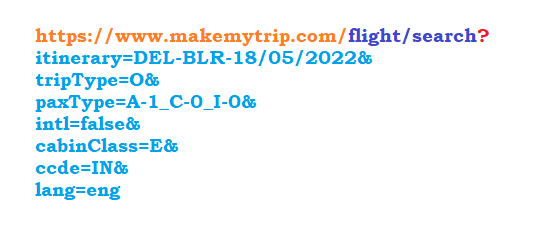
**PATCH :** is to update the record, where partial information is sufficient

**Path Parameter: PP** are variable parts of the URL path.Typically it is used to point to a specific resource within a collection.

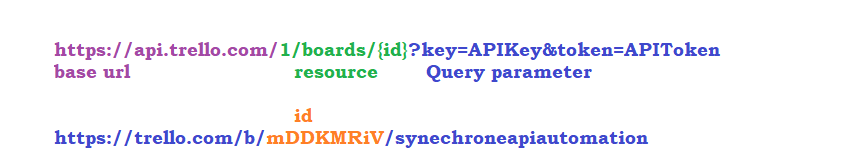
<https://jsonplaceholder.typicode.com/users> => collection

<https://jsonplaceholder.typicode.com/users/2> => getting one record

**Query Parameter :** Query parameters are used to sort / filter the resources. usually query parameter starts with ‘**?’**

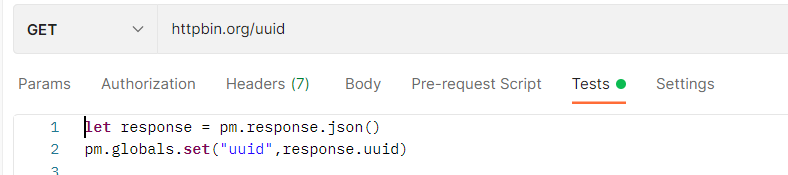
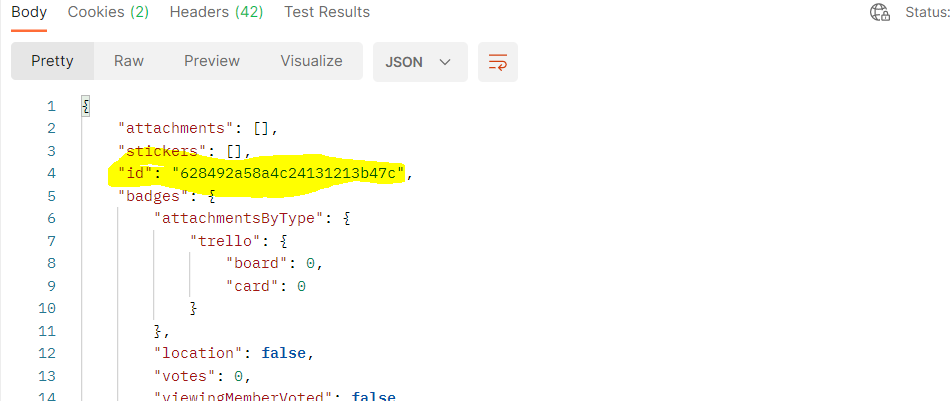


Trello API

* Register to Trello - <https://trello.com/>
* Create a Sample Board
  + List
  + Card
* Go to the api documentation of trello.
* <https://developer.atlassian.com/cloud/trello/rest/>
* To use the api’s we need to get the authentication token, because every api requires the token
* To generate API Key and Token
  + click on TRello Rest API From left panel
  + Click on API introduction
  + click on <https://trello.com/app-key> in the introduction page to get the key
  + click on token in the same page <https://trello.com/app-key> , to generate the token
* 

POST-MAN

VARIABLSE IN POSTMAN

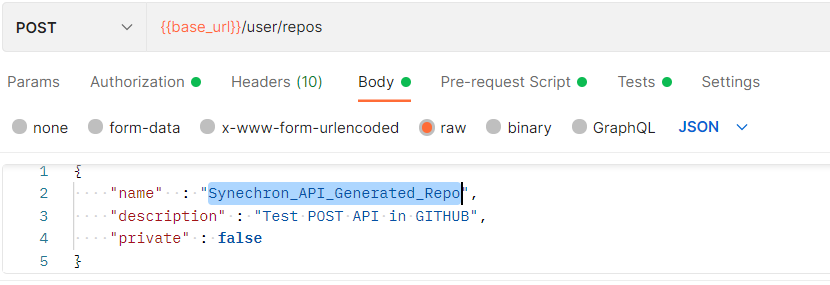
* GLOBAL VARIABELS
  + variables common to all sub-projects in the postman
* SAVING GLOBAL VARIABLES
  + RESPONSE
  + 
  + Test to save
  + 
* ENVIRONMENT VARIABLES
  + variables specific to project
* SAVING ENVIRONMENT VARIABLES
  + RESPONSE
  + 
  + TEST TO SAVE
* 
* PRE-REQ SCRIPT IN POST-MAN
  + this feature is used to create dynamic data to the test. pre-request script will be executed before the test is executed or before API is triggered
* 

GIT-HUB

* create a git-hub account - <https://github.com/signup>
* open google page and search for : github developer api
* <https://docs.github.com/en/rest>
* Get the authentication token
  + click on the usericon in github.com
  + select settings from the list
  + click on Developer Settings form the left panel
  + click on personal access token
  + click on generate new token
  + select all check box except admin related
  + copy the token generated and use in postman as Barrier Token

Body Parameter

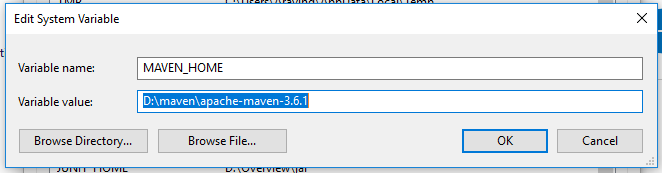
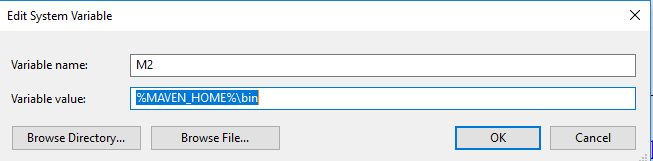
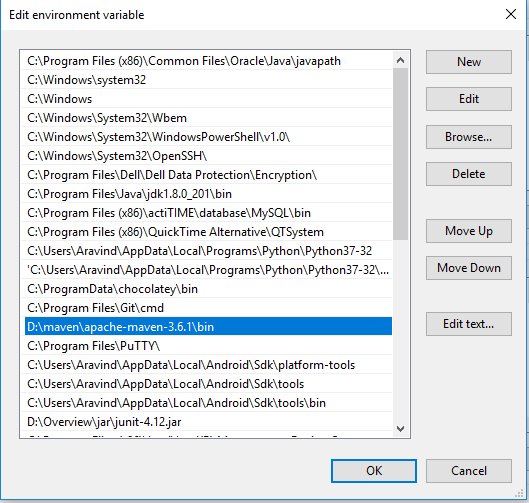
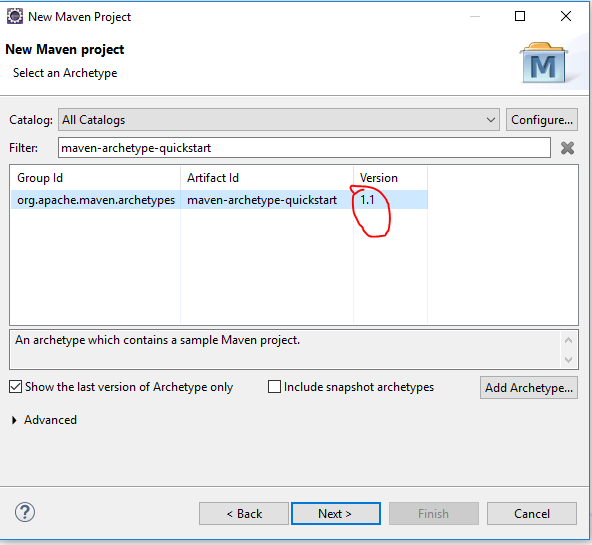
json file to be passed from the body of PostMan



Rest Assured :

Rest assured is a java library built on top of HTTP Builder. All HTTP Methods are available inside Rest Assured so we can easily automate HTTP Requests and Response.

# Setup :

* Download and install java
* Download and install Eclipse : <https://www.eclipse.org/downloads/packages/>
* Download and install maven : <https://maven.apache.org/download.cgi>
* 
* 
* 
* Open Eclipse and start creating a Maven project
* 
* Update the JDK and JRE in the project
* Update the pom.xml with latest dependencies <https://github.com/rest-assured/rest-assured/wiki/GettingStarted> OR <https://mvnrepository.com/>

Terminologies in REST

* **Base URI** – its a END Point
* **given()** – is the first method to be called in REST Assured, which defines the pre-condition to the REST Call.
  + given holds path/query parameters, request headers, cookies information etc
* **when()** – holds the request type. get, post, delete, patch ..
* **then()** – holds the validation - All the assert statements for the response
* **extract()** – whenever we want to extract the response and play with json

Validation in REST

We can use **Hamcrest** package to validate the response

# Rest assured Test Format:

1. Given when then format



2. RestAssured class format



3. given expect when format



# POST Request

while sending post request we can send data using

* path parameter
* query parameter
* body parameter

body parameter can be passed as String or payload

payload is nothing but a java object, which is having getters and setters.

Passing dynamic data - Faker API

* search for Faker api and go to github link
* <https://github.com/DiUS/java-faker>
* Update pom.xml
* Refer the documentation on how to use Faker API

Understanding complex JSON

json is a key value pair file, which can be used to represent a complex objects

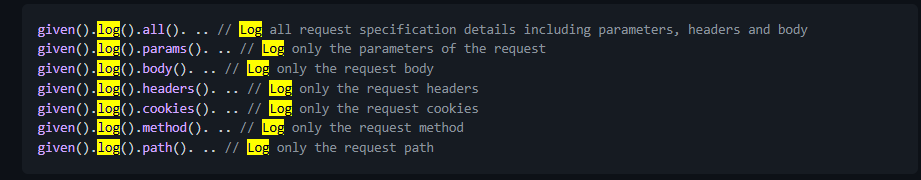
json can be a Map – { ----- }

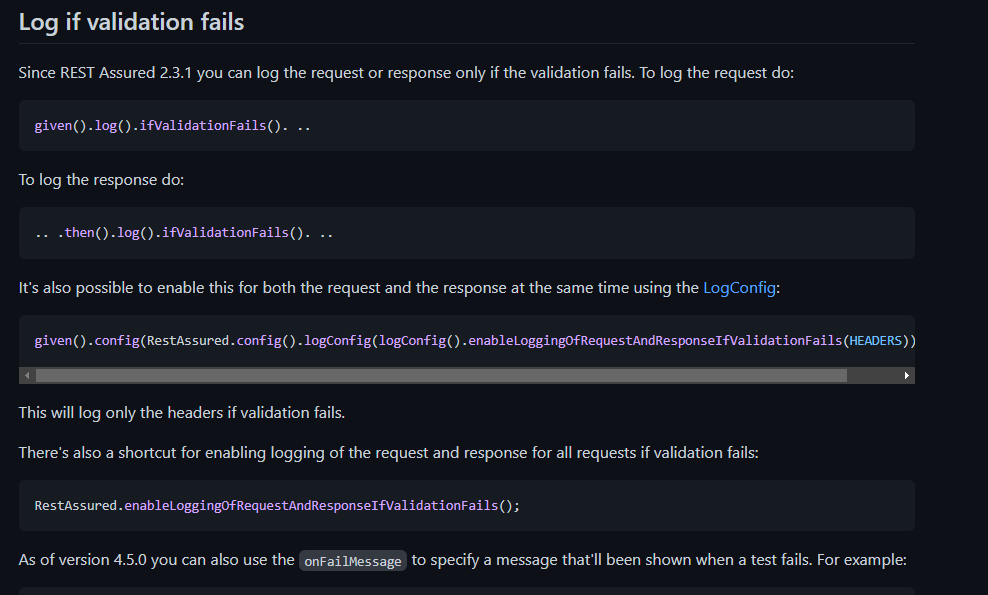
Array of map – [{`````},{``````}]

<https://jsoneditoronline.org>

Enabling Logs in REST

REST API has a logger class which can be used to log,





Using Filters in REST

To store the body, request headers, response headers, proxy information then we have to go with filters.

<https://mvnrepository.com/artifact/commons-io/commons-io/2.11.0>



Request and Response Specifications

To reduce code effort/ duplicate code in the tests we can go with request and response specification objects





Validation in REST Assured

* Jayway Json path – Java Syntax
  + <https://github.com/json-path/JsonPath>
  + validate the expression in <https://jsonpath.herokuapp.com/>
    - extract the response
    - convert it to string
    - call read method present in side *com.jayway.jsonpath.JsonPath; and pass the expression*
* JsonPath (Inbuilt) – Groovy
  + extract the response
  + convert it to string
  + create a object to **io.restassured.path.json.JsonPath**
  + call get method and pass the key

# ValidatableResponse :

if we want to validate the collections then instead of extracting as Reponse object, extract it as ValidatableResponse Object



Static Imports:



Root Path

Root path can be used for setting the root element to perform validation

