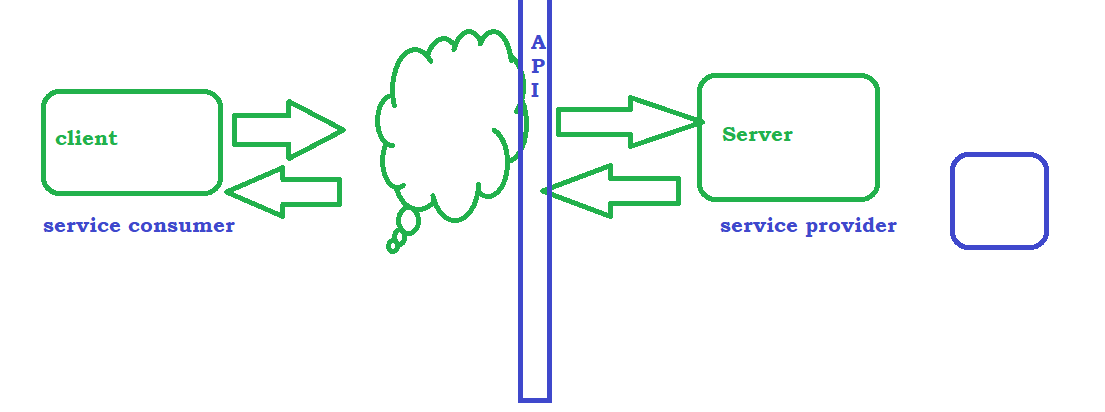
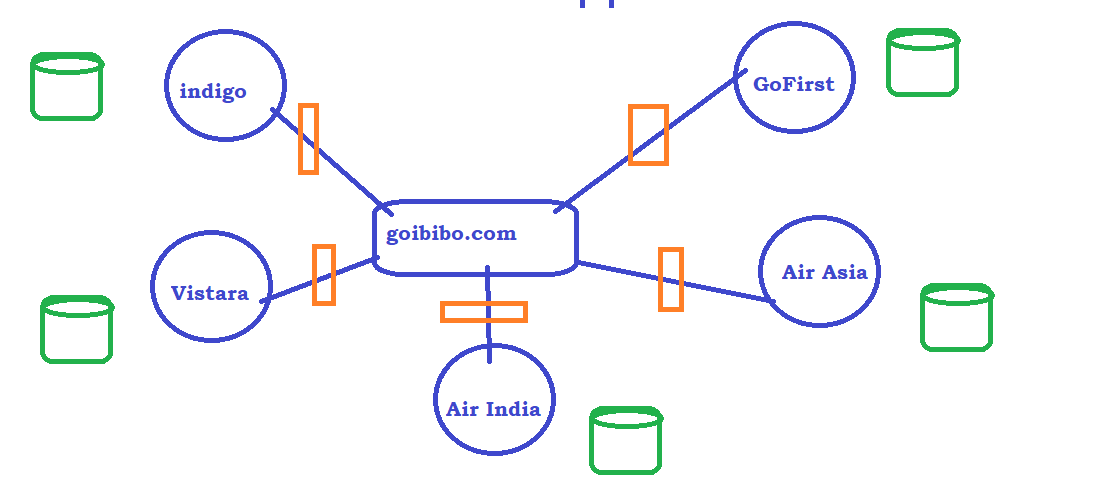
Web Services :

Services available over the Web, these web services are provided by Service Providers.



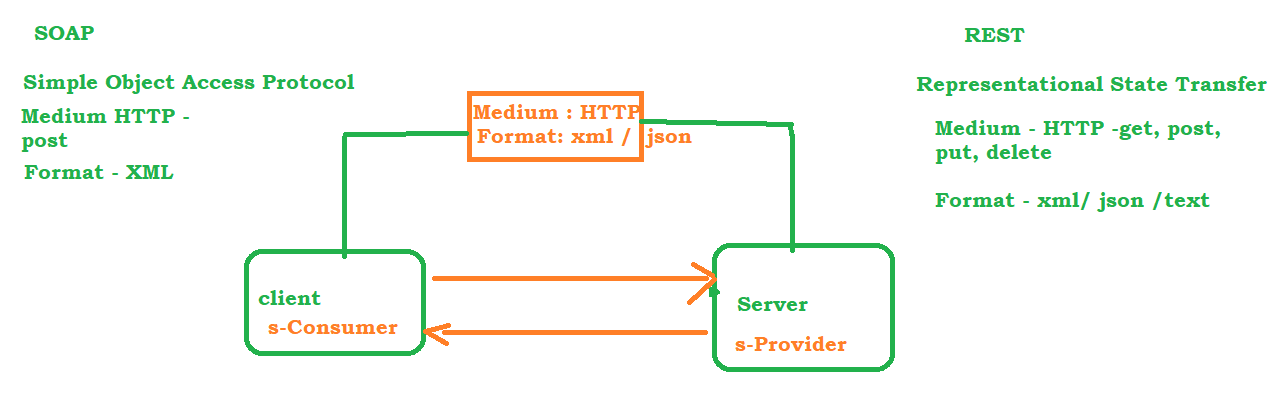
Communication :

We Use Language to communicate with other person, We use GUI to communicate with PC / Laptop, similarly from one machine to another machine if we want to communicate then we use API

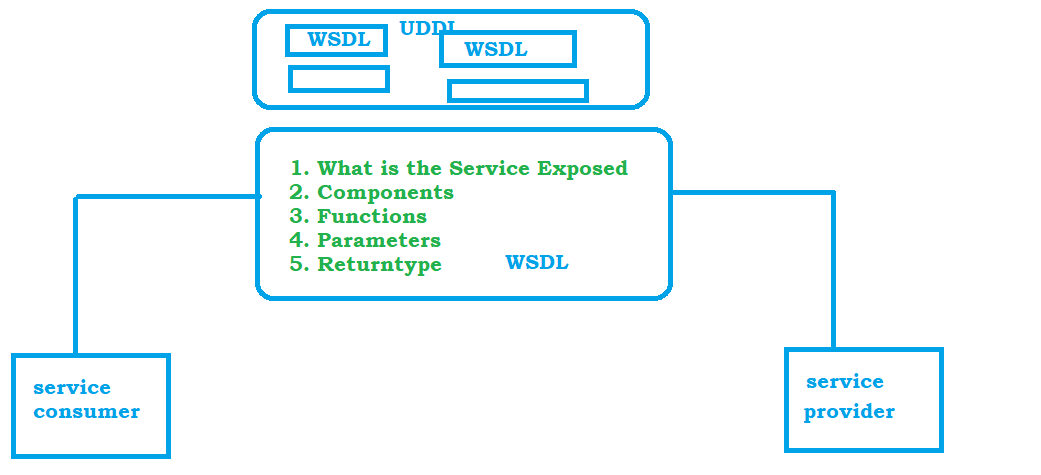


# Types :

* SOAP
* REST



SOAP : WSDL and UDDI

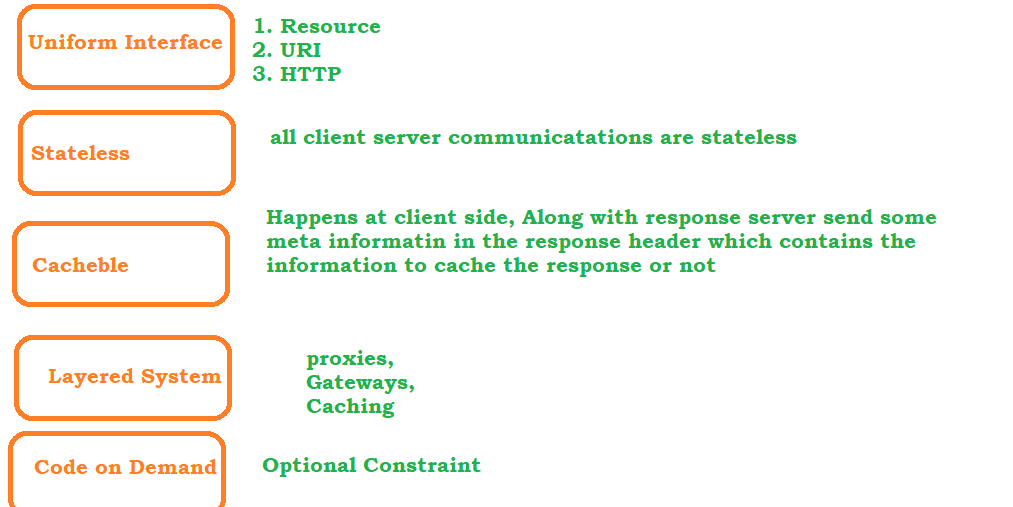


SOAP –



REST

Any WebService that communicates / exchanges information between 2 application using REST principles area called RESTful WebServides



# HTTP Request –

its a packet of information that one computer send to another

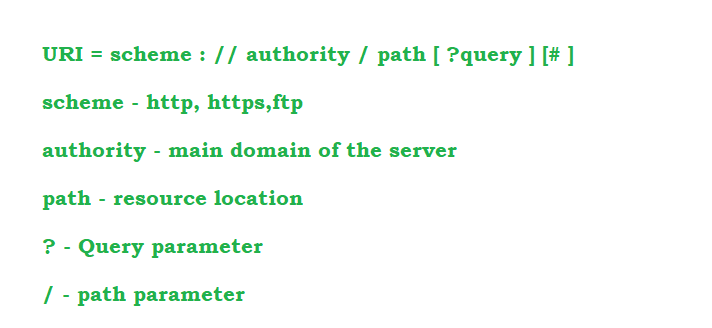
* Request Line Http method (GET, POST) , Request URI, HTTP Protocol
  + <https://jsonplaceholder.typicode.com/todos/2>
* Header / Headers - 0 or more header information which contains additional information (user credentials )
* Optional Body – Its a part of HTTP Request where additional content can be sent

# HTTP Response –

its a packet of information sent by the server in response to the request made by client.

* Request Line Http method (GET, POST) , Request URI, HTTP Protocol
  + <https://jsonplaceholder.typicode.com/todos/2>
* Header / Headers - 0 or more header information which contains additional information (user credentials )
* Optional Body – Its a part of HTTP Request where additional content can be sent

# URI



HTTP Methods

* C – Create U – Update R- Read D - Delete operations
* GET – to read , 200 status code with response
* POST – is used to create new record in the collection of resources –201SC
* PUT – is used to update the existing record.
* DELETE – delete the record

Manual Testing of APIs

## PostMan :

* go to <https://www.postman.com/downloads/> and download the software
* install the software on your machine

Trello

1. Register to Trello and login to application
2. Create a board of your own - Create list – Card etc
3. Refer the official API Documentation <https://developer.atlassian.com/>
4. Navigate to <https://developer.atlassian.com/cloud/trello/guides/rest-api/api-introduction/>

key - c5f676759b86029624f7dcb31ccf655e

token - 9b60bd7325defc221aa0203822765f426ecb134adcf46f7ca823c569ced3a2f3

PostMan – Variables

# Global Variables – Accessible throughout the Collections

# Environment Variables - Specific to Project

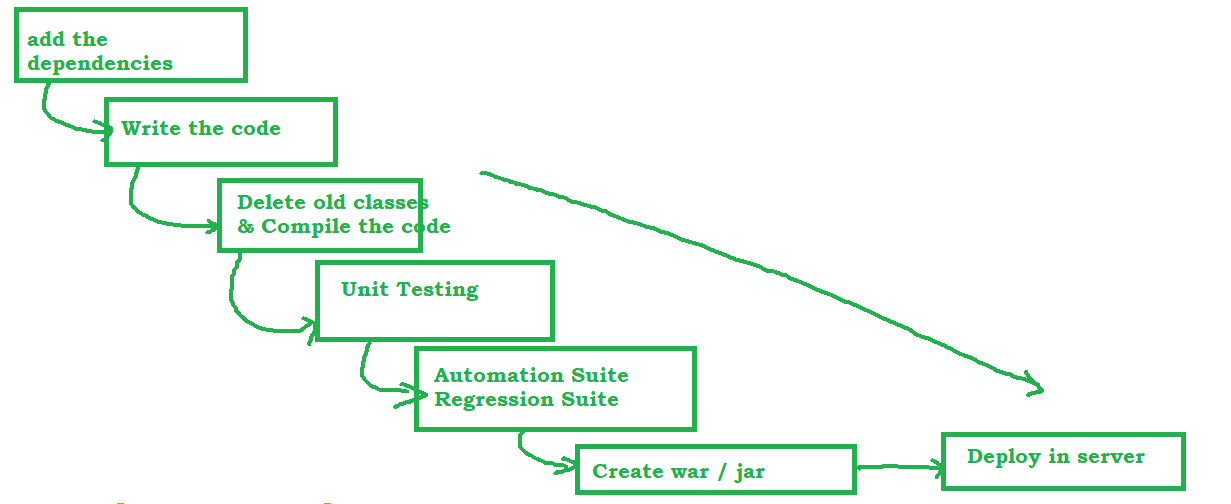
# Tests

# Pre Request Script

GIT HUB APIs

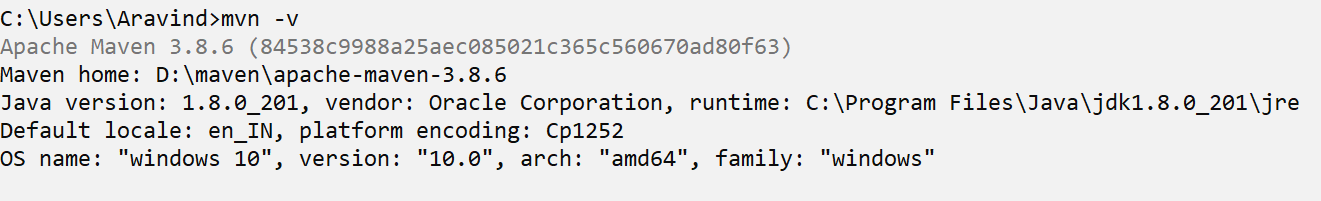
* go to github.com
* click on signup and create a account
* go go developer website <https://docs.github.com/en/rest>
* We should have authentication token to execute the api’s
* Go to your Account -> Settings -> Developer Settings -> Personal Access tokens -> Generate the token

Maven – Setup and Intro



* Download and unzip <https://maven.apache.org/download.cgi>
* set the Environment variables
* MAVEN\_HOME - D:\maven\apache-maven-3.8.6
* M2 -D:\maven\apache-maven-3.8.6\bin
* PATH - D:\maven\apache-maven-3.8.6\bin

Open command prompt and type mvn –v

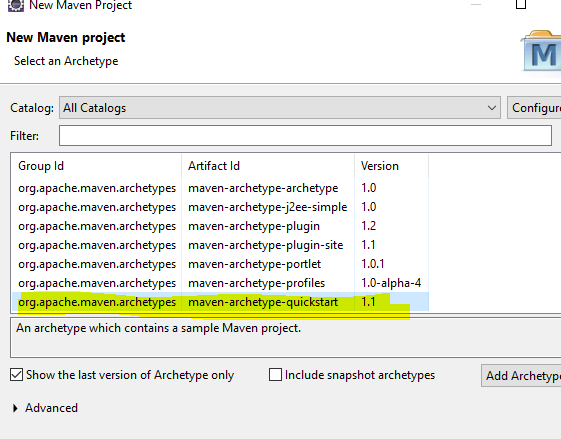


# POM.xml

Pom file is the heart of Maven project, which contain all the information about the project.

pom file holds all the required plugins and dependencies for the project

Configure Eclipse with Maven and REST

* Create a Maven project
* 
* <https://mvnrepository.com/artifact/io.rest-assured/rest-assured>
* <https://mvnrepository.com/artifact/io.rest-assured/json-path>
* <https://mvnrepository.com/artifact/io.rest-assured/xml-path>
* json-schema-validator
* spring-mock-mvc

# Rest Assured Methods :

* given() – first method to be called, which holds the api information like header, parameters, cookies
* when() – holds the type of request, get, post, delete, put...
* then() – holds the validation code
* extract() – to save the response for further processing

# Different ways / Style we can write REST code

1. BDD style – give when then format
2. RestAssured Style
3. given expect when format

# Automate Get Request

# Automate POST Request

# Automated DELETE Request

# Extract the Response from the Request.

TestNG

* <http://www.testng.org>
* Execute the tests one after another
* Default HTML output file will be generated
* option to rerun the failed tests
* TestNG Runtime environment for Executing the tests
* Annotations

# Log the Request and Response

<https://github.com/rest-assured/rest-assured/wiki/Usage>

Request Logs :

1. Request Headers
2. Request Body
3. Request All
4. Request Parameters

Response Logs :

1. Response Headers
2. Response Body
3. Response All
4. Log only if fail

POST with Payload and POJO

# Payload :

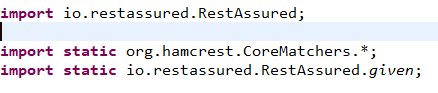
* go to json viewer - <http://jsonviewer.stack.hu/>
* copy paste the json from the body of postman
* remove the whitespaces
* copy paste the string to java file and store it in a String Variable

# POJO

* POJO – Plain old Java Object
* A Class contain only getters and setters

Static imports

static imports are supported in java to import the methods directly to the class so that we no need to specify the class name whenever we are using the methods.



Passing Dynamic Data using faker api

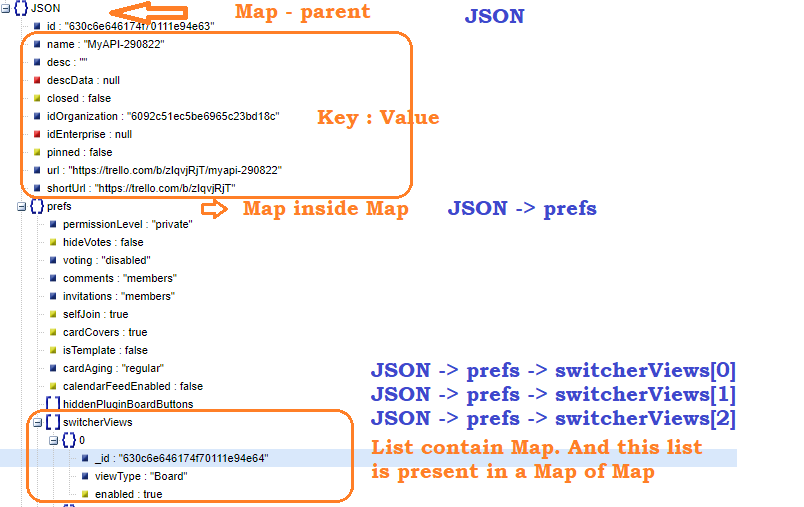
* Search for java faker api in google and select the link to official website
* <https://github.com/DiUS/java-faker>
* use Faker API to Create dynamic data

Understanding Complex JSON

Simple text file with .json extension and it contain only 2 type of collections

1. List : [ ]

2. Map : { }



Extract Response OR Validations in REST Assured

# Jaway JsonPath Library

* + Its a Third party library which uses JAVA Syntax
  + update pom.xml <https://github.com/json-path/JsonPath>
  + Use <https://jsonpath.herokuapp.com/> to validate the Expression
* Sample JSON And Expressions

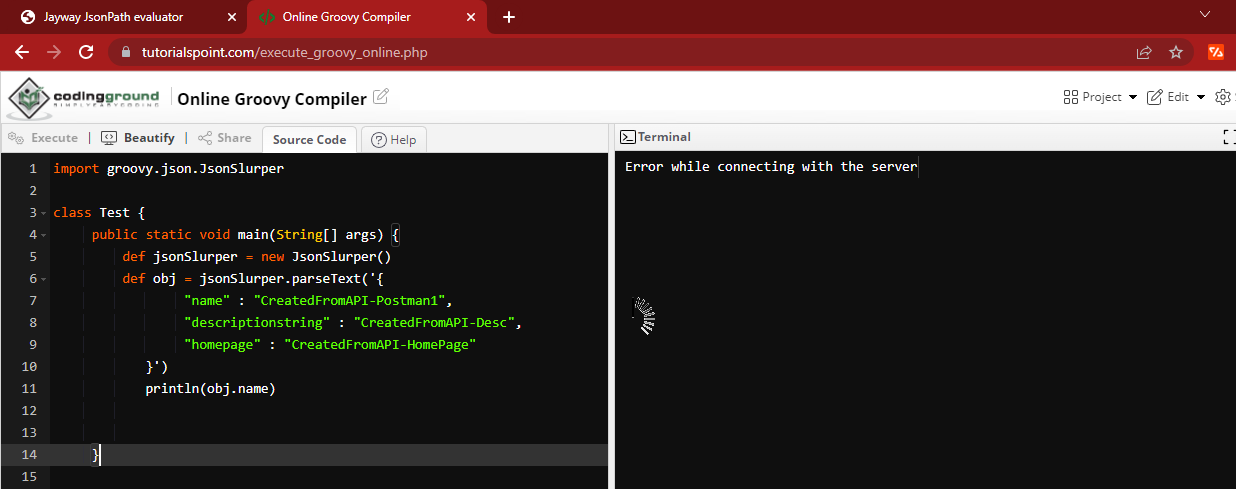
{"store":{"book":[{"category":"reference","author":"Nigel Rees","title":"Sayings of the Century","price":8.95},{"category":"fiction","author":"Evelyn Waugh","title":"Sword of Honour","price":12.99},{"category":"fiction","author":"Herman Melville","title":"Moby Dick","isbn":"0-553-21311-3","price":8.99},{"category":"fiction","author":"J. R. R. Tolkien","title":"The Lord of the Rings","isbn":"0-395-19395-8","price":22.99}],"bicycle":{"color":"red","price":19.95}},"expensive":10}

* Expressions

|  |  |
| --- | --- |
| Element | Expression |
| ROOT | $ |
| Expensive | $.expensive |
| Bycicle color | $.store.bicycle.color |
| Author of first book | $.store.book[0].author |
| Author of last book | $.store.book[-1].author |
| All authors | $.store.book[\*].author |
| All Price | $.store.book[\*].price |
| All Price from All nodes | $..price |
| Find Author whose price is less than 10 | $.store.book[\*].**[?(@.price<10)]**.author |
| Min value | $..price.min() |
| Max value | $..price.max() |

# Inbuilt JsonPath Library

* + Uses Groovy Syntax
  + there is no way we can test the expressions we write, if we want to do that then there is a workaround., internally Jsonpath library uses JsonSlurper class, we can create an obj and test the expressions.
  + <https://www.tutorialspoint.com/execute_groovy_online.php>



import groovy.json.JsonSlurper

class Test {

public static void main(String[] args) {

def jsonSlurper = new JsonSlurper()

def obj = jsonSlurper.parseText('{

"name" : "CreatedFromAPI-Postman1",

"descriptionstring" : "CreatedFromAPI-Desc",

"homepage" : "CreatedFromAPI-HomePage"

}')

println(obj.name)

}

}

# Extracting using ValidateResponse