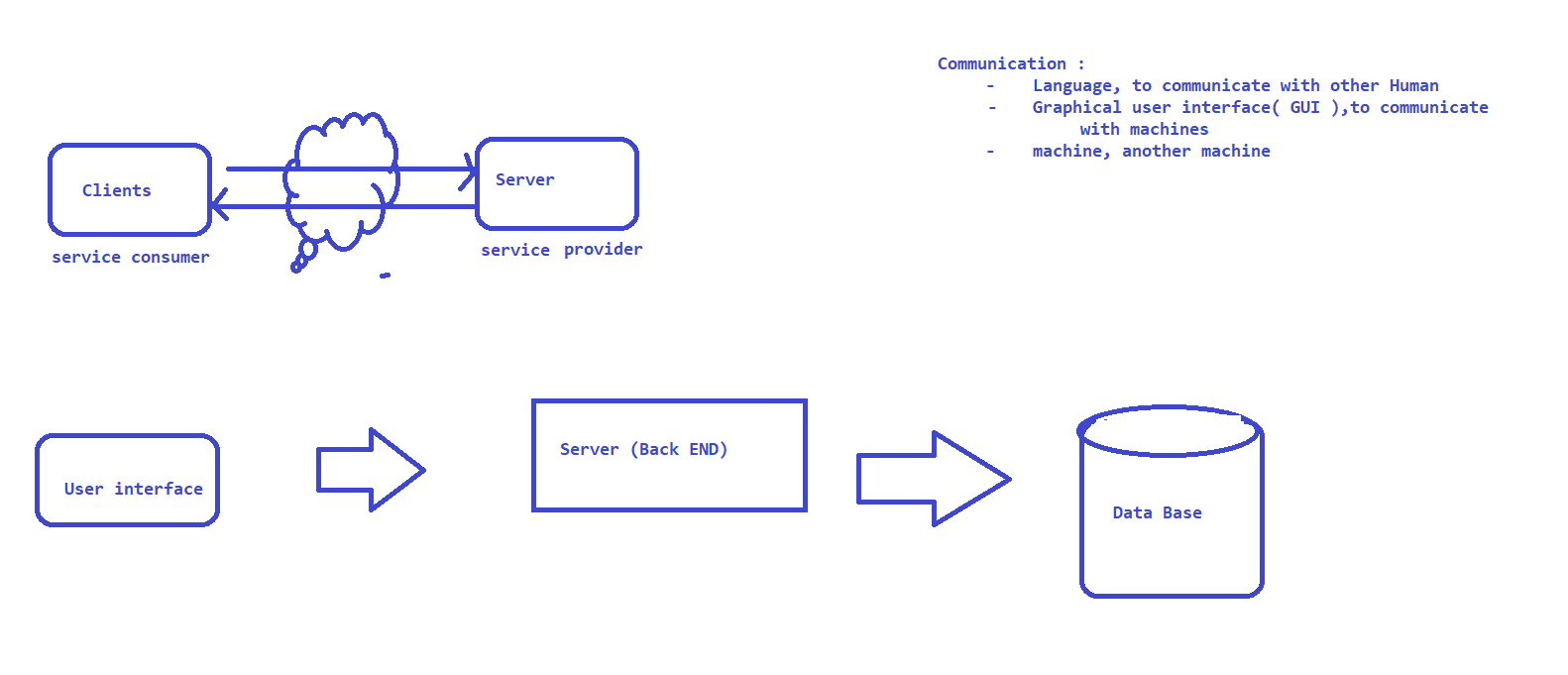
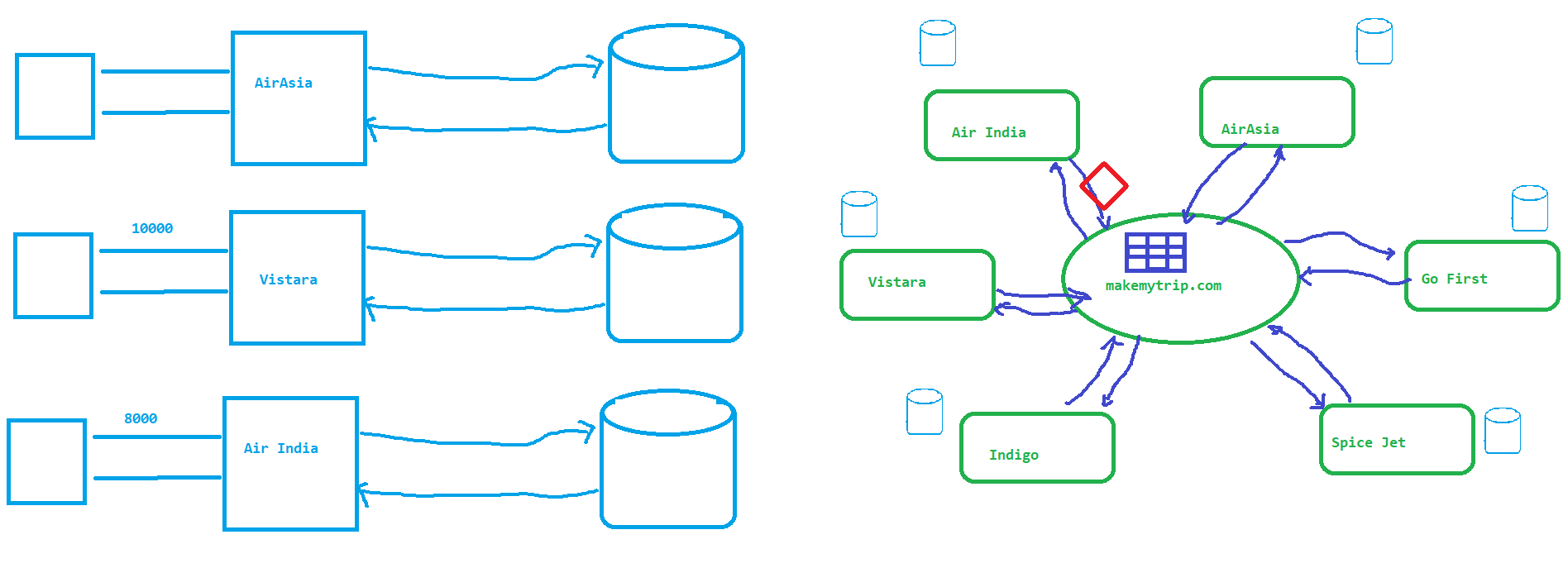
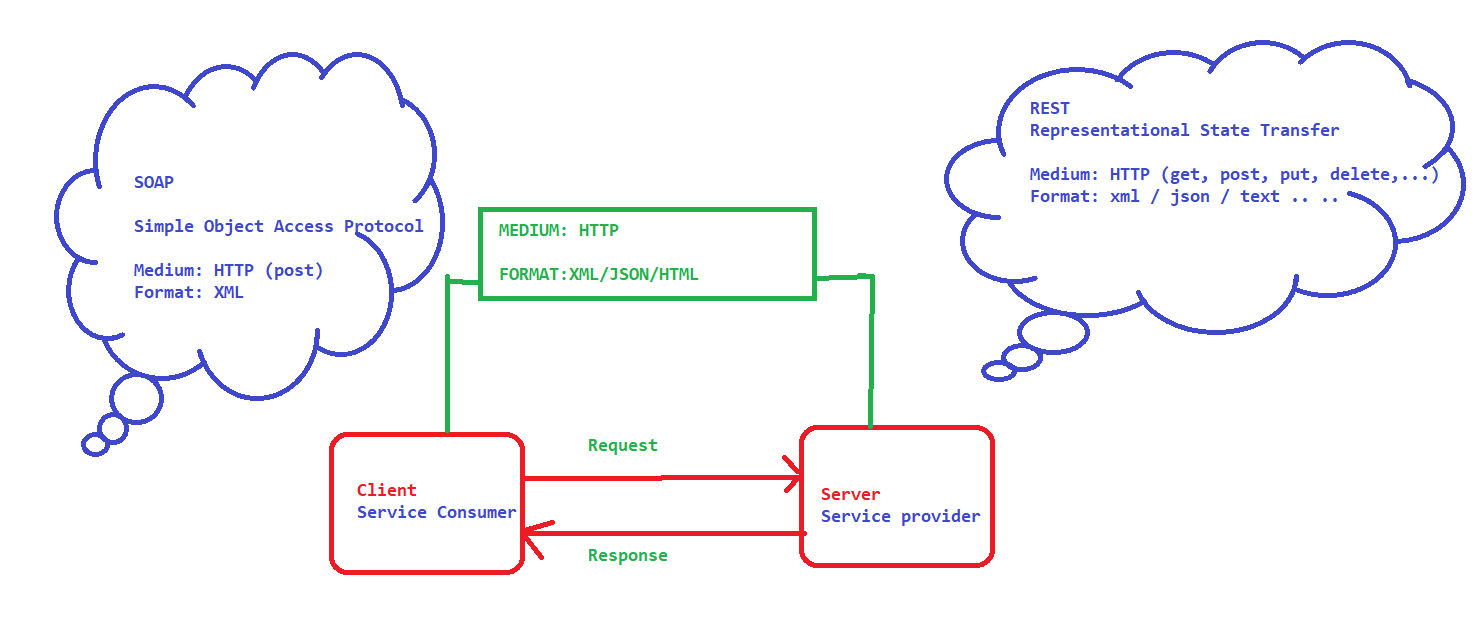
Web Services :

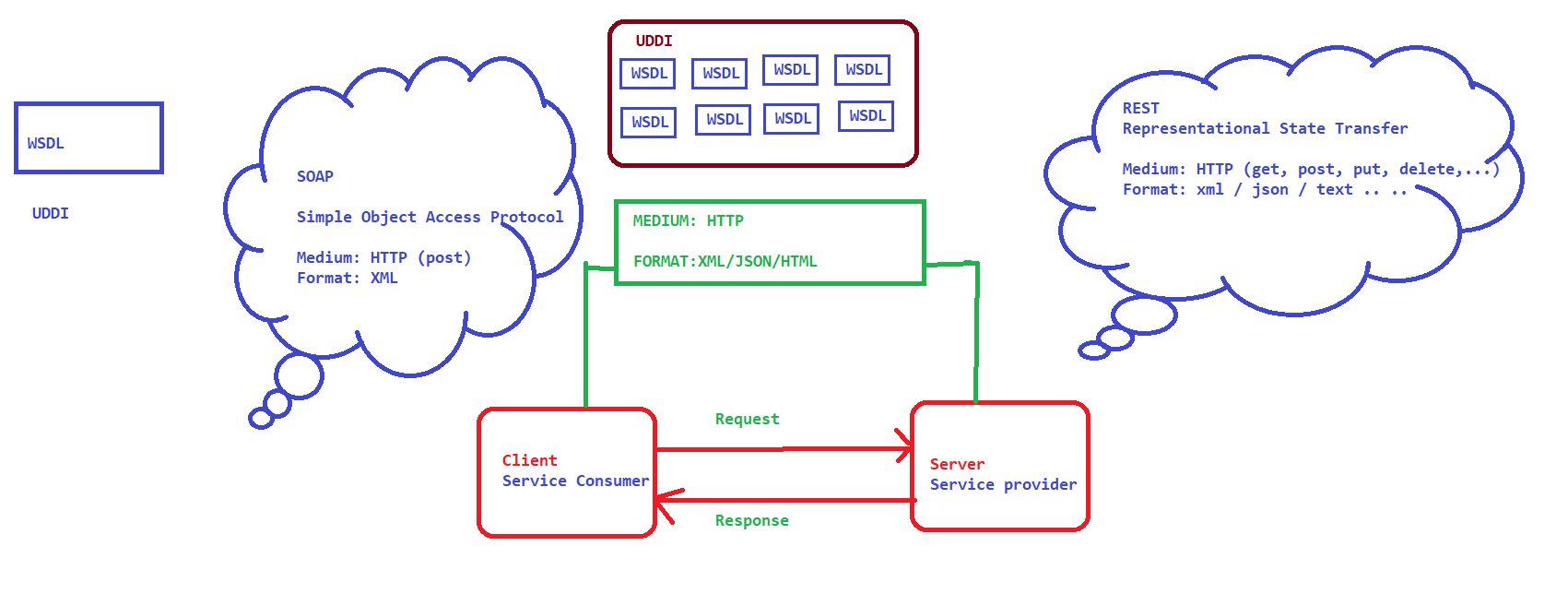




High level architecture :



WSDL AND UDDI

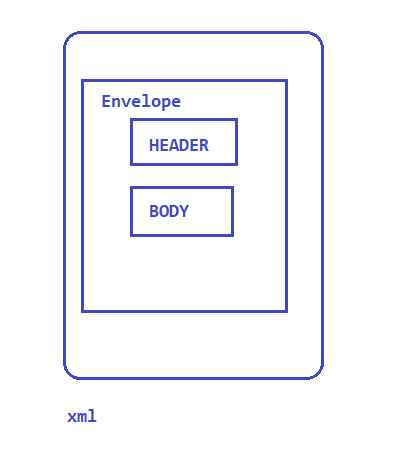


SOAP

Any Web Service that follow the guidelines specified by the SOAP Webservice specifications are called as SOAP WebService .

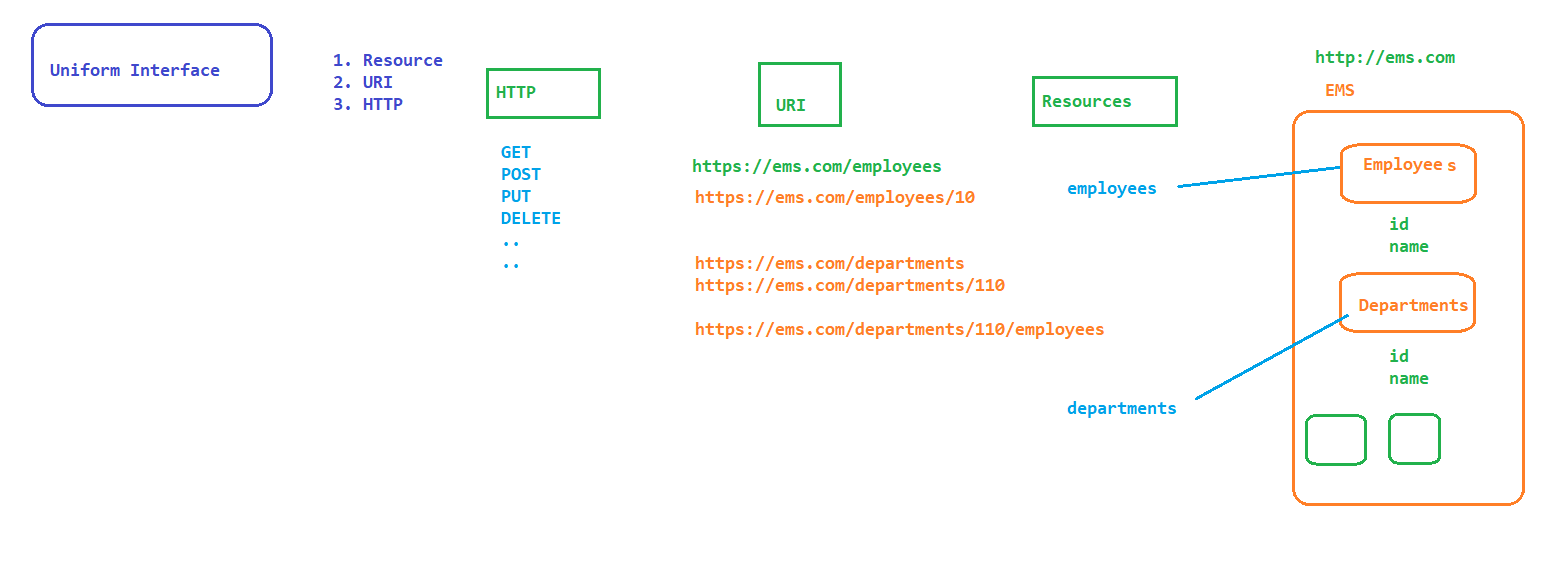
- W3C is a international body that develops standard for WWW

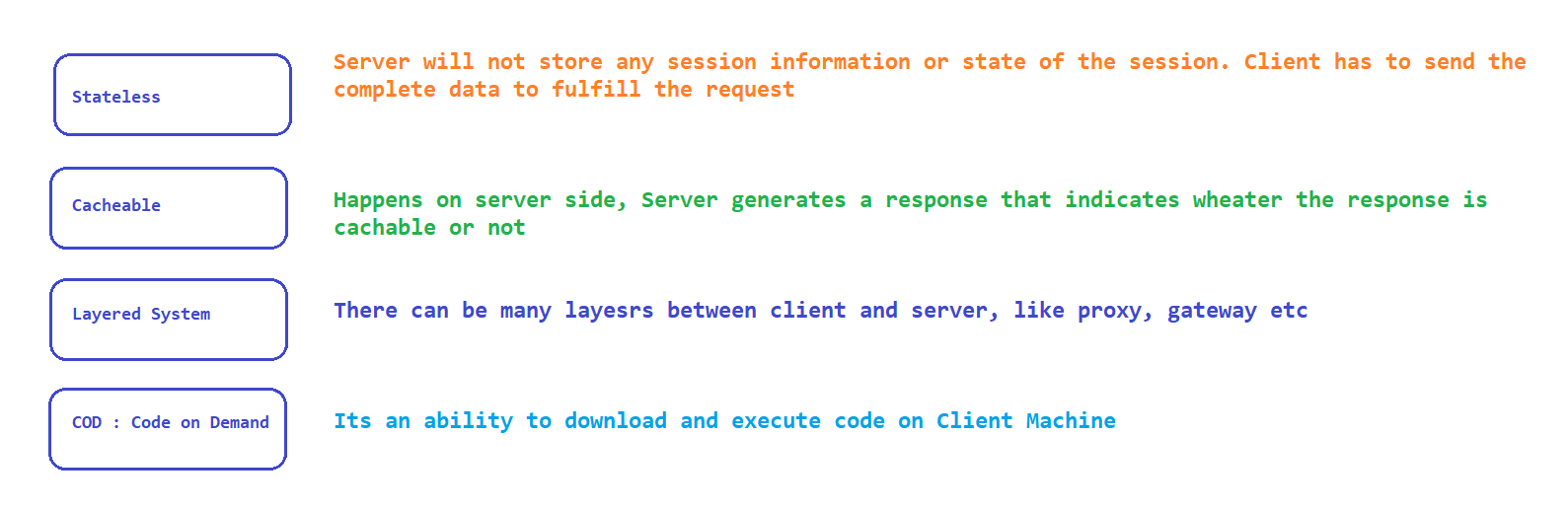
1. SOAP
2. WSDL
3. UDDI
4. WS-Security
5. WS-Policy ..



REST

* REST is an abbreviation of Representational state transfer
* REST is an architectural Style, it is not a protocol!
* There is no strict specification OR no central body to control the specifications
* REST is a principle or design method to create a Web Services. While developing the WS if we use these principles then it becomes Restful WS





HTTP Request : <https://jsonplaceholder.typicode.com/todos/1>

* Packet of information that one computer send to another to communicate.
* HTTP Request contains:
  + Request Line : Method token (get, post ,....) and Request URI
  + Headers : 0 or more Headers
  + Optional Request Body

HTTP Response :

* Packet of information sent by the server for the request made by client
* HTTP Response contains
  + Request Line :
  + Headers : 0 or more
  + Optional Response Body

Elements

1. Resource
2. Resource Identifier
3. Representation
4. Representation Metadata
5. HTTP Get
6. HTTP post
7. HTTP put
8. HTTP Delete

URI :

scheme://baseurl[/path parameters][?query parameters]

scheme : http, https, ftp, file://

path : defines the resource

POSTMAN

* Download and install the application from <https://www.postman.com/downloads/>
* Create Collection and start adding APIs
* <https://httpbin.org> : APIs in the Application

Adding Tests in PostMan

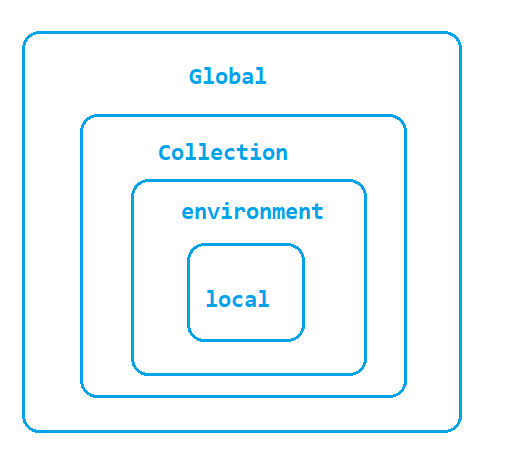
* click on Tests section
* use the template to add tests as per the requirements

using output of one test in another

## save the output as a global / environmental variable

## use it wherever we need

variables :

****

Trello – API Reference

1. Register to Trello, Login to Trello Application
2. Create a Board and play with the application
3. Go to official API Documentation <https://developer.atlassian.com/>
4. Navigate to <https://developer.atlassian.com/cloud/trello/guides/rest-api/api-introduction/>
5. click on <https://trello.com/app-key> to generate token
6. On the same page click on Token to generate the token
7. Click on Reference section on the top to see the APIs which are exposed

PostMan Tests

* Click on Tests
* Select the predefined tests from the Snippet section

pm.test("Status code is 200", **function** () {

    pm.response.to.have.status(200);

});

pm.test("Body matches string", **function** () {

    pm.expect(pm.response.text()).to.include("621460a0eee6fd57cc80ccd7");

});

pm.test("Status code name has string", **function** () {

    pm.response.to.have.status("OK");

});

Pre Request Script

To create dynamic data for the test we can make use of pre-request script in prostman

const cardName = "Card\_" + parseInt(Math.random() \* 1000)

pm.globals.set("dynamic\_cardname",cardName)

Pre-request bock will be executed before executing or sending the request

Git Hub APIs

* go to <https://github.com/>
* click on signup and create a account
* go to developer website <https://docs.github.com/en/rest>
* Get Authentication Key
  + click on profile icon
  + click on settings
  + click on developer settings
  + click on personal access token
  + click on generate token
* Access all the apis from <https://docs.github.com/en/rest/reference>
* in Postman under Authorization select Bearer Token and paste the token that you generated in git hub account and send the request

Eclipse Project Setup using Rest Assured

### Rest Assured is a Java Library for automating REST based webservices. Rest Assured is built on top of HTTP Builder so it supports all the HTTP Methods, GET, POST, PUT, DELETE, PATCH etc.

SETUP:

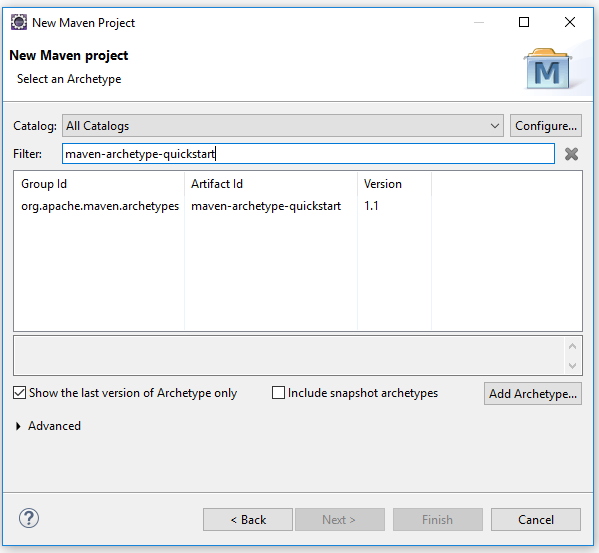
### Download and install JAVA (atleast 1.8)

### Set the Environment Variables

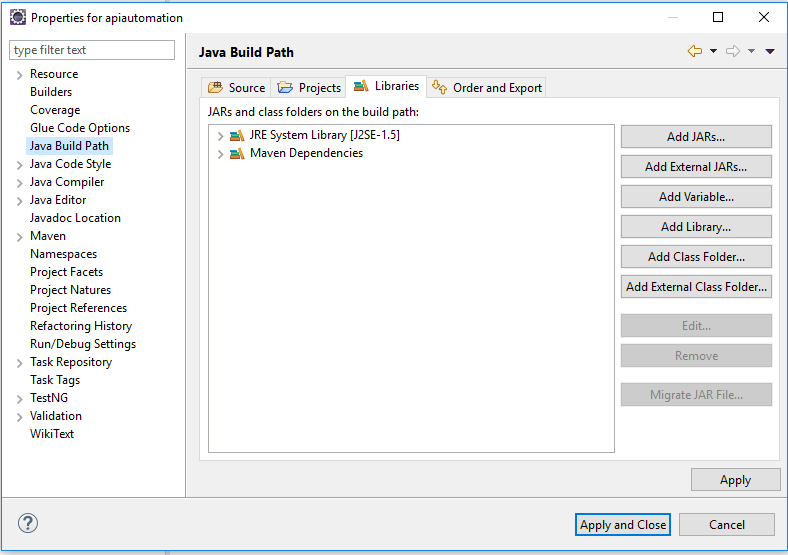
### Download Eclipse – for JAVA Developers

### Download Maven and set Environment Variables

* + MAVEN\_HOME – unzip location of maven
  + M2 – Bin directory of Maven
* Create a Eclipse Maven Project



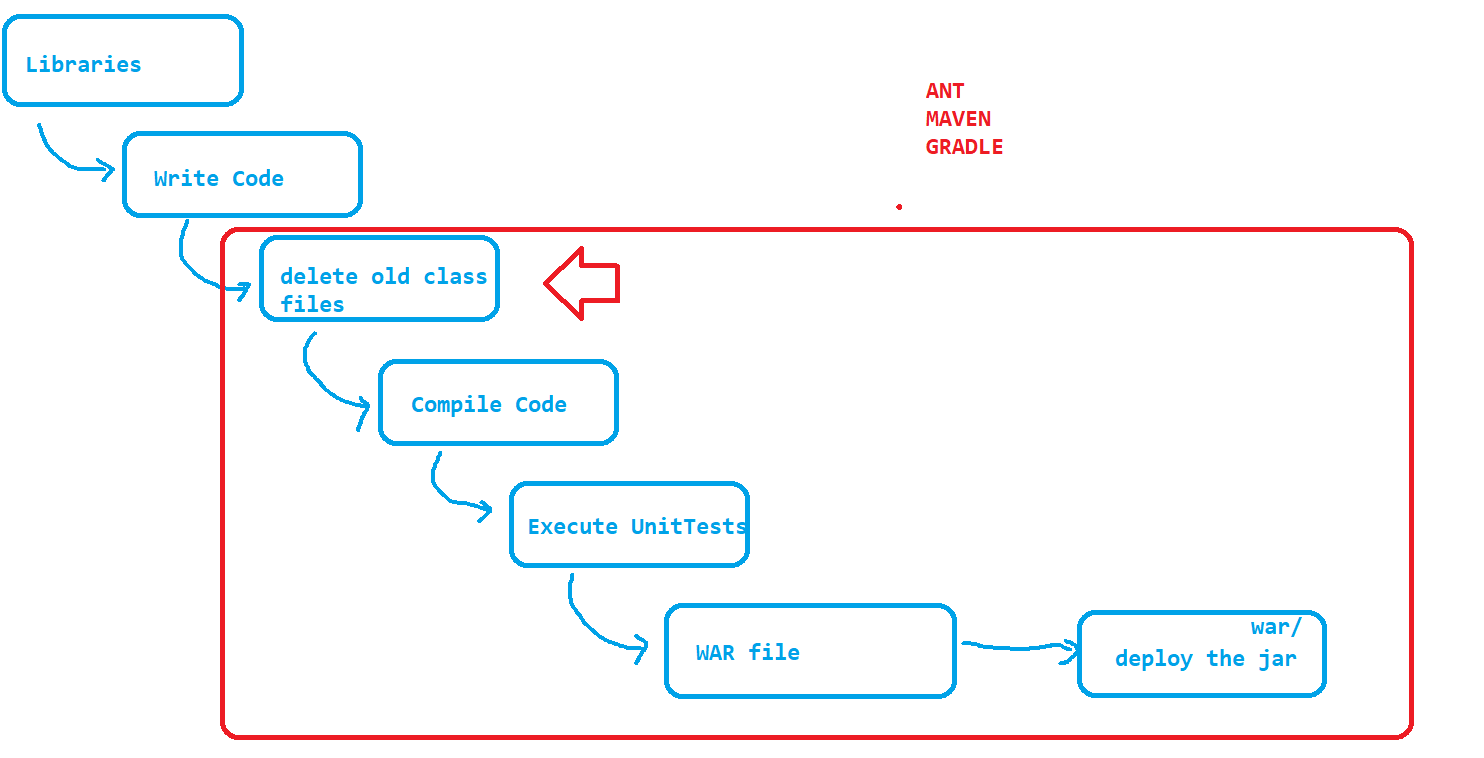
* Verify compiler version and the JRE Version used in the project if they are different then update it to the correct version
* Remove JRE And update the latest one

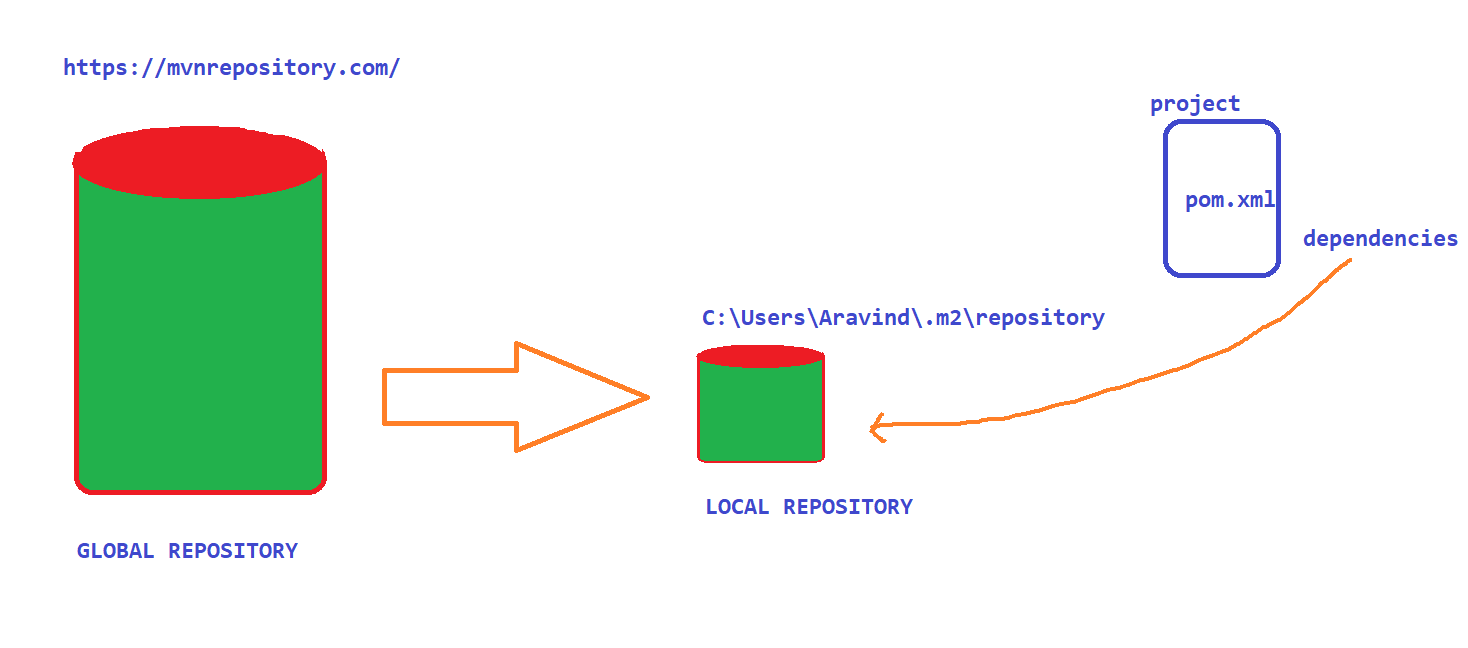


* sd

Maven

Maven is a Build Automation Tool.



* Complete dependency management can be handled by MAVEN
* MAVEN WORK FLOW
* 
* Update pom.xml
  + <https://mvnrepository.com/artifact/io.rest-assured/rest-assured/4.5.1>
  + <https://mvnrepository.com/artifact/io.rest-assured/json-path/4.5.1>
  + <https://mvnrepository.com/artifact/io.rest-assured/xml-path/4.5.1>
  + <https://mvnrepository.com/artifact/io.rest-assured/json-schema-validator/4.5.1>
  + <https://mvnrepository.com/artifact/io.rest-assured/spring-mock-mvc/4.5.1>

Terminologies in Rest Assured :

1. BASE URI

Rest Assured Methods :

* given()
  + first method to be called and it contains some information like, header, parameters, cookies
* when()
  + Holds information about the type of http method (get, post, put, delete.. )
* then()
  + Holds the Validation code
* extract()
  + We can extract / save the response from the request for further processing

Automate GET Request :



Different ways of Writing REST Code:

1. given when then format -> BDD Style of writing a code
2. RestAssured class format
3. given expect when format

Validation in REST using Hamcrest

Types of parameter

* path parameter
* queryparameter

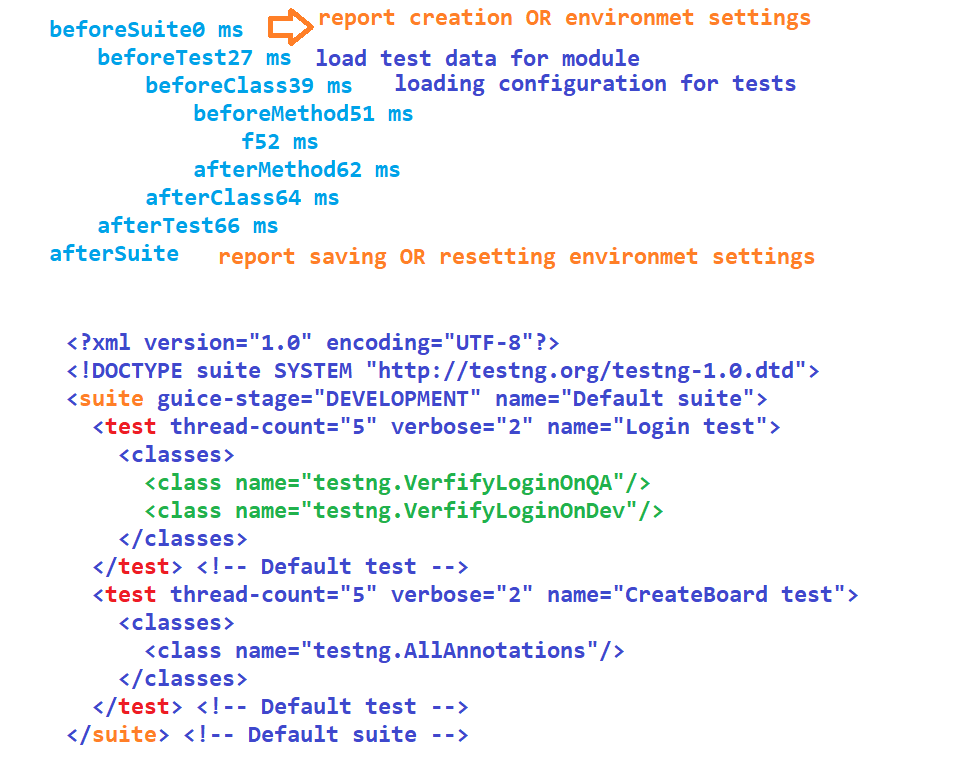
Passing Dynamic data – Faker API

* google for faker api
* <https://github.com/DiUS/java-faker>
* Update pom.xml – dependency
* Refer the documentation on how to use

Understanding Complex JSON

* Json is a key value pair file, which can be used to represent the complex objects
* <https://jsoneditoronline.org/#left=local.zimaja>
* Json object
* Json Array

TestNG – Framework



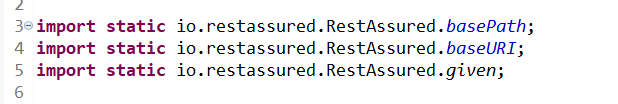
ValidatableResponse:

if we want to validate the resoponse returned by the API then we can use ValidatableResponse Class

Reusability with Rest Assured:

Update the static members of RestAssured – baseURI, basePath,port

# static import

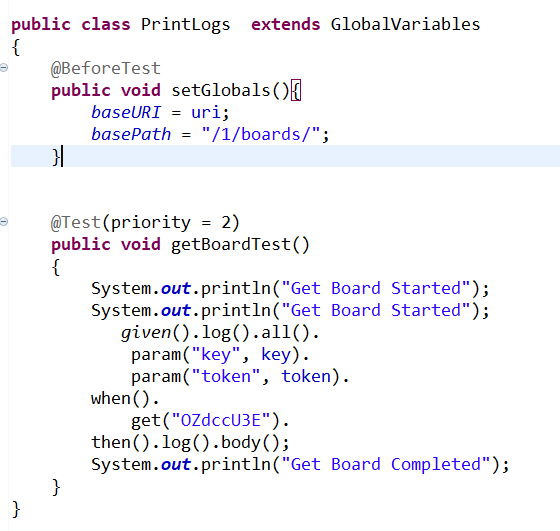


# Validation in REST Assured:

* Jayway JsonPath – Java Syntax
  + go to <https://github.com/json-path/JsonPath>
  + Update pom.xml
  + <https://jsonpath.herokuapp.com/> to validate the expression
* Inbuilt JsonPath – Groovy Syntax

# Putting Logs in REST

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

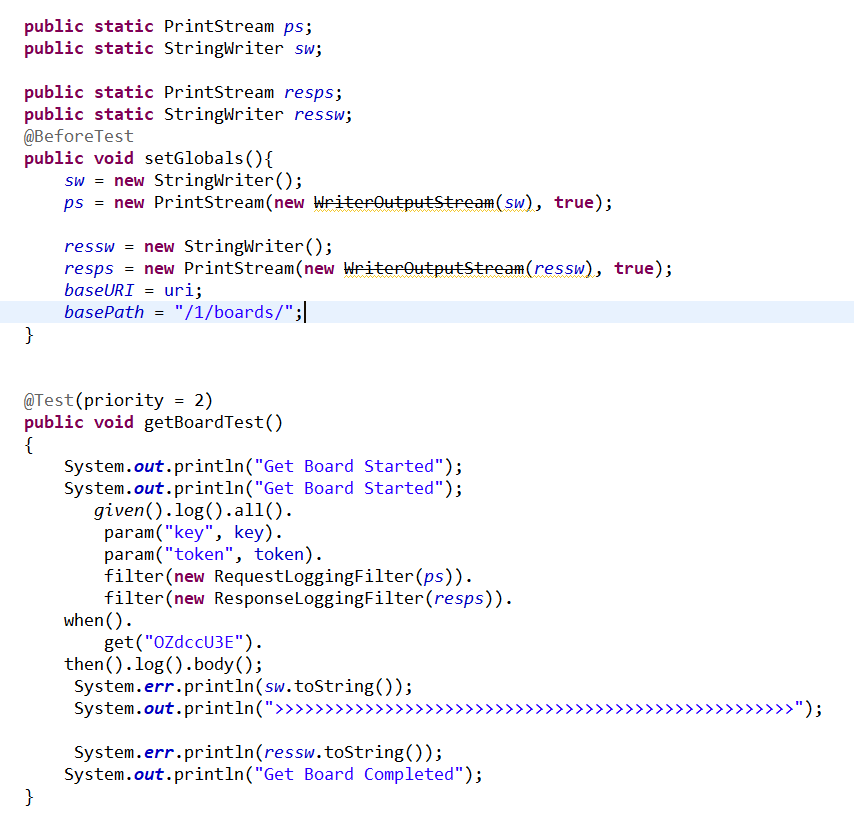


# Filters in REST

- using logs we can print the headrs,parameters and body in the console. But if we want to store in a external media like, textfile or log4j or any reporting then we logs are not sufficient.

we have to make use of filters to store the header body or response headres

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



# Request and Response Specifications

To reduce the code effort we can use request and response specification objects

to send the request and to validate the response

