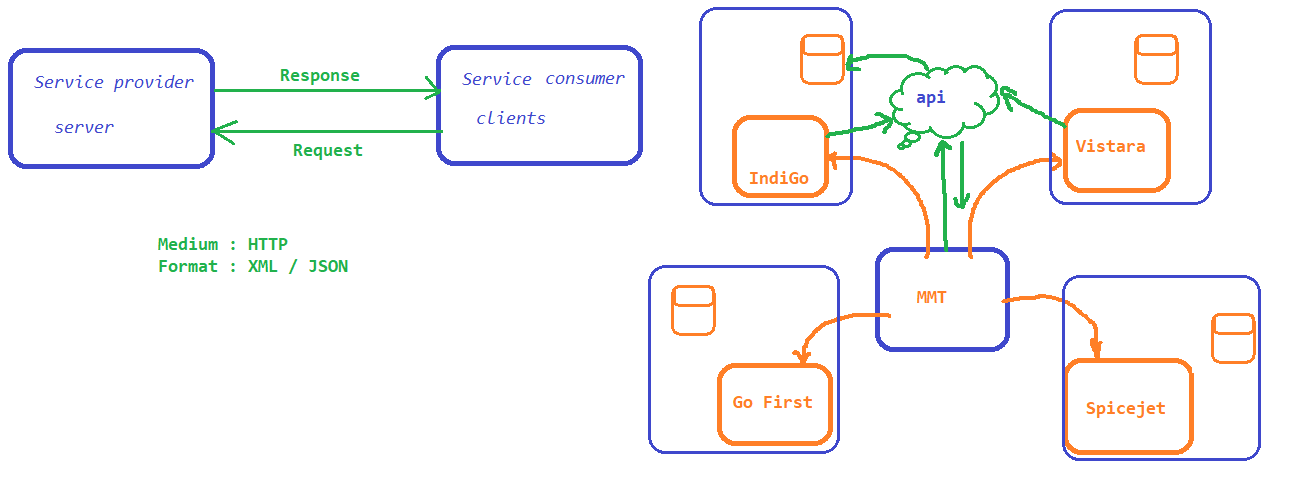
Web Services :

services available over the web, which enables the communication between 2 applications.

Web Services follow standard protocol or format for communication.



Types of WS :

1. SOAP : Simple Object Access Protocol

Medium : HTTP (POST)

Format : XML

2. REST : Representational State Transfer, not a protocol its a Design approach

Medium : HTTP – get, post, put, delete

Format : XML, JSON, TEXT, ...

WSDL and UDDI :

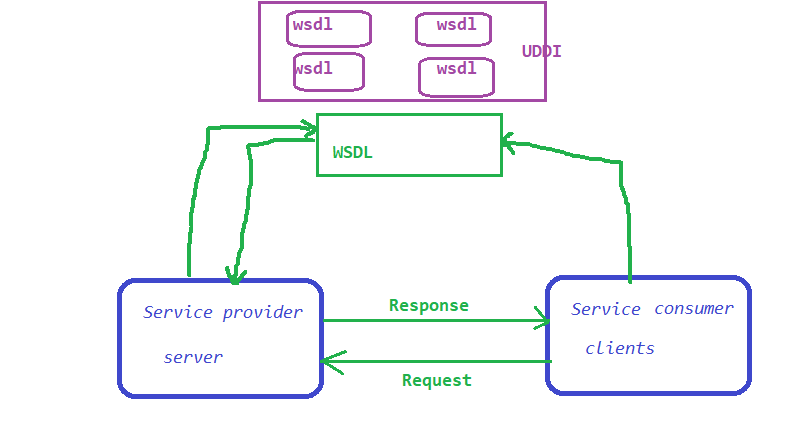
WSDL Stands for Web Services Description Language. its a xml document which will have complete information about the api which are exposed by the service provider.

Information :??? ->

* what is the webservice for
* what are the components it has
* functions
* parameters to pass
* what is the return type

UDDI :

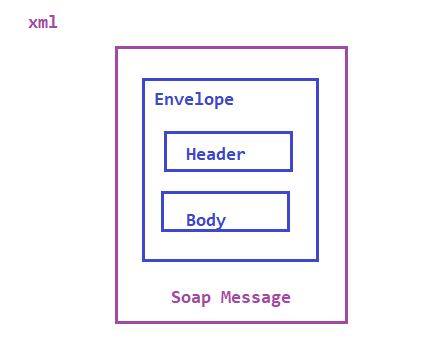
Universal Description, Discovery and Integration,it is a global directory / location where all service providers will expose there WSDL.



SOAP :

Any Web Service that follow the standard guidelines specified by SOAP WS Specification is a SOAP Web Service.

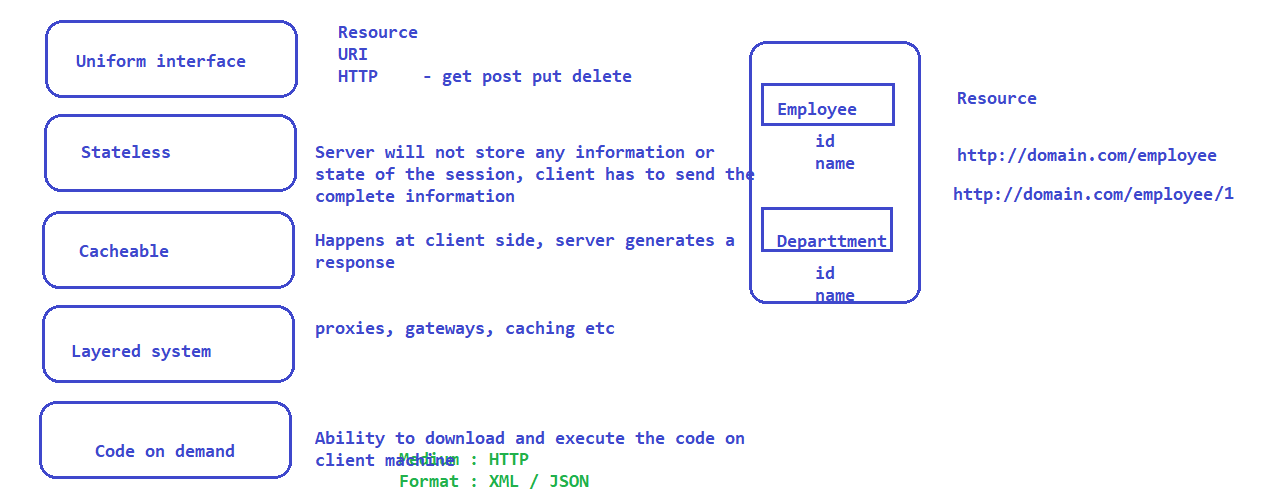
There is a international body that develops open standard for WWW and its called W3C.



REST :

A Web Service that communicates / exchanges information between 2 applications using REST architecture / principles are called RESTFul webServices.

REST is not a protocol, there is no strict specifications or no central body to control the specifications. REST is just a design method or design principle to create Web Services. WHEN THESE PRINCIPLES ARE USED WHILE DESIGNING A WEB SERVICE IT BECOMES RESTFUL WEB SERVICE.



HTTP Request

packet of information one computer send to another to communicate

HTTP Request contains

- Request Line - https://jsonplaceholder.typicode.com/posts/1

- Header / Headers

- Optional body

HTTP Response

packet of information one computer send to another to communicate

HTTP Response contains

- Request Line - https://jsonplaceholder.typicode.com/posts/1

- Header / Headers

- Optional body

- Resource : Information stored in server, which can be requested by a client

<https://twitter.com/> is the base url and home login etc are the resources

URI :

scheme : authority / path[ ? query parameters ]

http/https/ftp : domain / resource ? any parameters to be passed to the api

**HTTP Methods**

--

*HTTP GET* – is used to retrieve the information from the server. we can not do any modifications to the data which is retrieved

If the resource found, then we get a status code of 200, else we get different status code along with the response .

404 – resource not found

501 – Authentication error

*HTTP POST*

post is used to create a new record / create a new resource into the collection of resources

if resource is created successfully then we get a status code of 201

*HTTP PUT*

put is used to update the existing record. post and put are used to insert or update the record .

*HTTP DELETE:*

is to delete the resource

Parameters :

1. path parameter :

* are separated by /
* path parameters are the variable part of URL. these are specifically used to point specific resource within a collection of resources
* Ex :
  + <https://www.google.com/images/123>
  + https://www.google.com/images/456

2. Query parameter :

* separated by ?
* used to filter the resource
* Ex :
  + https://www.google.com/search?q=yash

Headers / Cookies :

Headers represents some additional information which is sent along with the request

Ex : Authentication information

PostMan Installation

1.

Applications :

1. HTTPBIN
2. trello
3. github

Trello :

1. Register with Trello - <https://trello.com/>

2. Create a workspace and then create board

3. Go to developer website of trellio - <https://developer.atlassian.com/cloud/trello/>

4. Get the api key and token - <https://developer.atlassian.com/cloud/trello/guides/rest-api/api-introduction/>

5. Refer the API’s from Reference section <https://developer.atlassian.com/cloud/trello/rest/api-group-actions/>

GitHUB :

1. Create a account in github.com

2. go to github developer api : https://docs.github.com/en/rest

3. <https://docs.github.com/en/rest/overview/resources-in-the-rest-api>

4. Get the authentication key

* click on user icon on top right corner and go to settings
* click on developer settings
* click on personal access token
* Generate new token
* keep the token safe

Rest Assured Library :

Prereq : Java and Eclipse

Configuring Eclipse

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

Rest Assured Automation :

/\*

\* 1. BDD : Given when then format

\* 2. RestAssured Class Format

\* 3. Given Expect When format

\*

\*/

* given() : this is the first method to be called, we can pass BASE URI, parameters, headers, cookies
* when() : specifies the type of call
  + get
  + put
  + post
  + delete
* then() : Assertion on the response
* extract() : to extract the response

Validation using Hamcrest :

Hamcrest package provides lot of functions through which we can validate the response.

GET :

POST :

- while submitting a post request the parameters should be of type query / path based on what is given in the documentation. whereas in case of get it doesn’t matter. irrespective of get or post we can send as a param.

Static imports to make code simple :

POJO Class to pass payload :

Faker API to generate dynamic data

Extracting the Response :

Jayway Json Path

* search for jayway jsonpath evaluator
* official link to download dependencies <https://github.com/json-path/JsonPath>

JsonPath

Generating logs for the request and response :

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

JSON Assert : to validate the complete JSON File.

* <https://github.com/skyscreamer/JSONassert>
* update pom.xml

Setting Root PATH

Request Specifications :

Response Specifications:

Validating Response time :

Proxy: