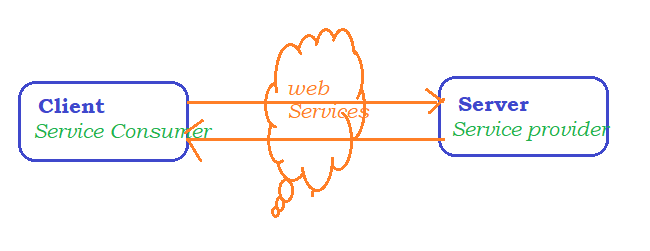
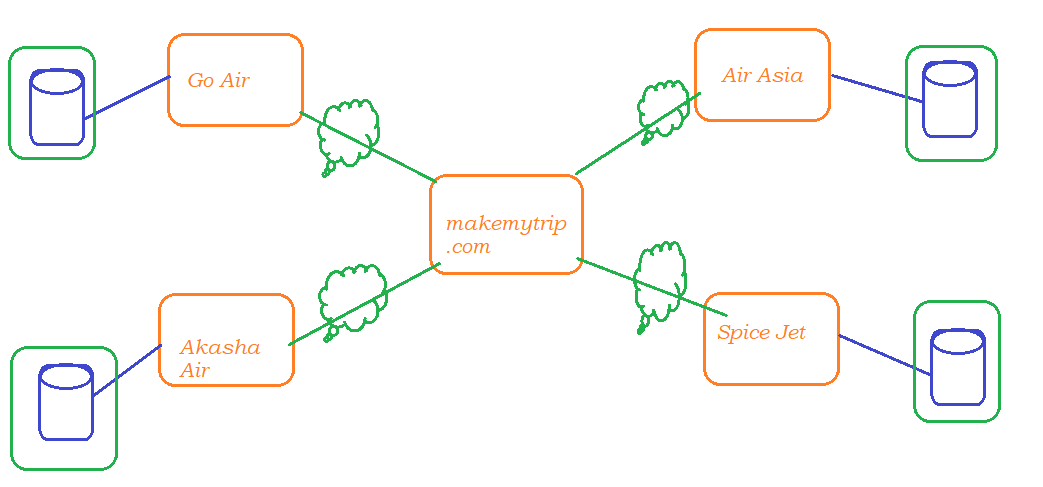
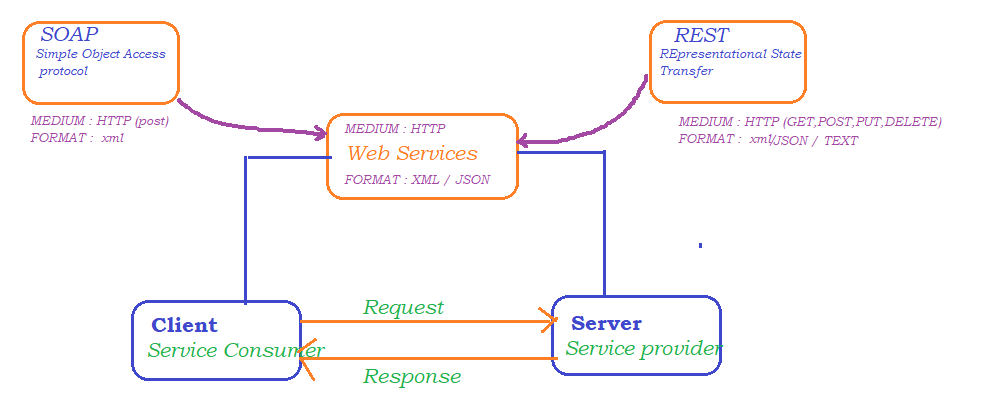
Web services Automation with REST

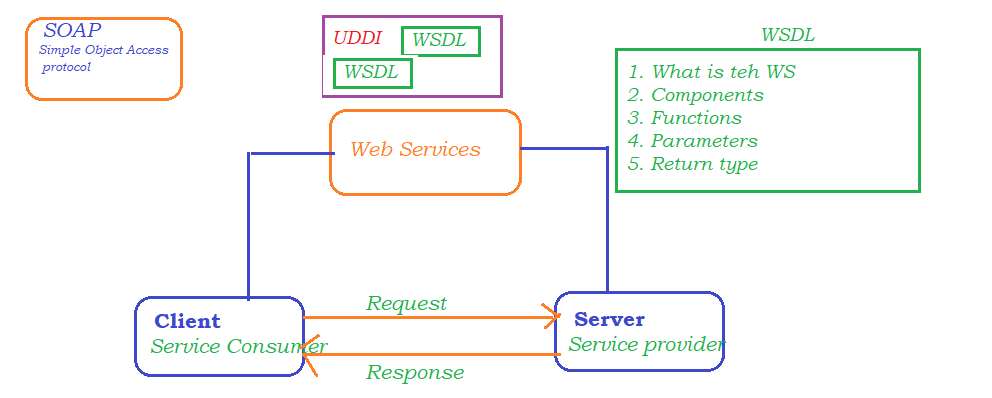


## High level architecture :





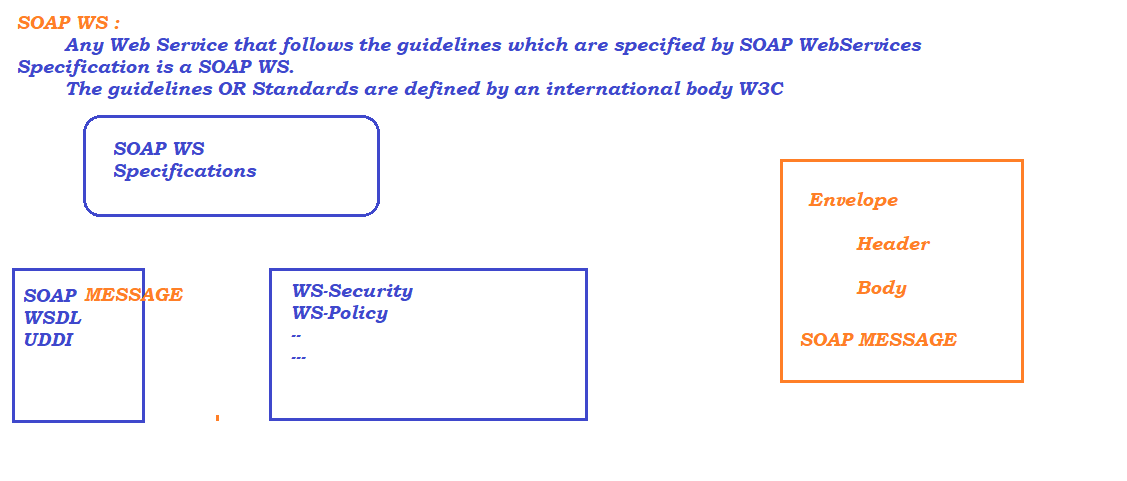
UDDI AND WSDL



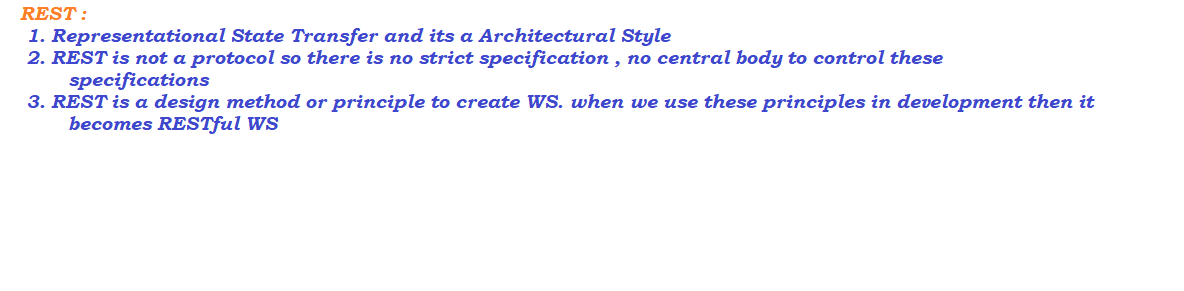
## W3C : (http://www.w3.org)

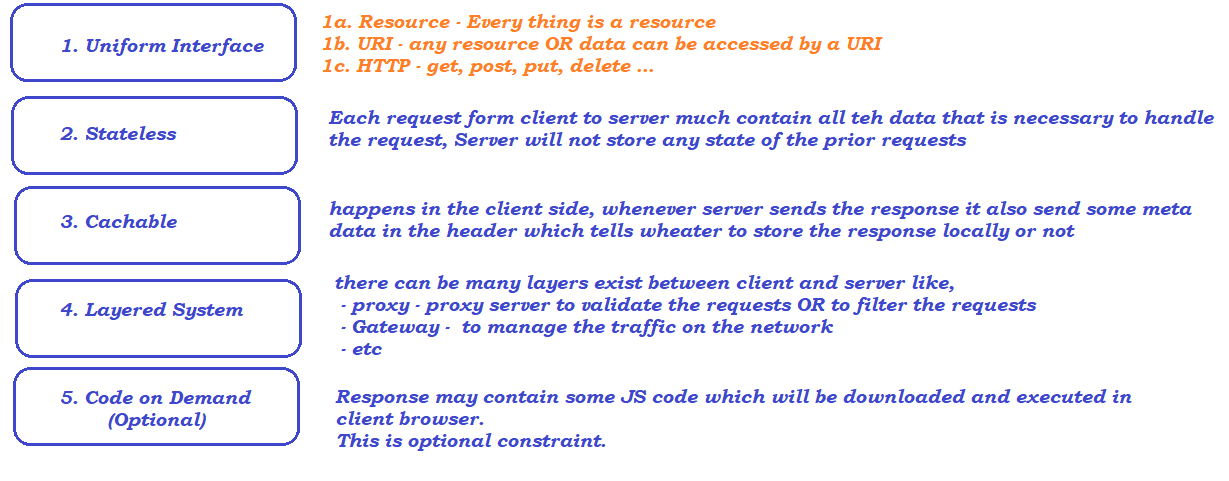
An international Community that develops Open Standards for WWW

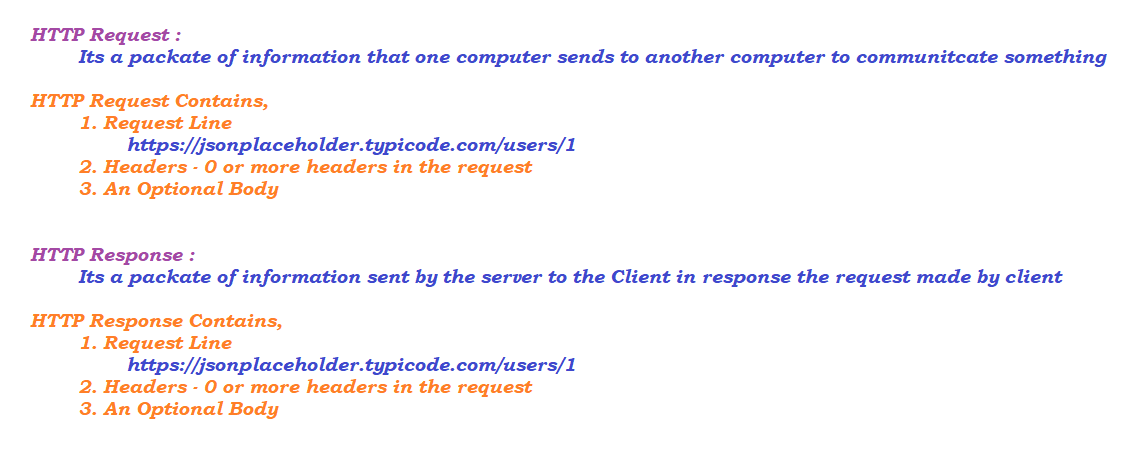
SOAP ARCHITECTURE



REST ARCHITECTURE







# Rest API Elements :

# Resource :

Information stored on a Server, which can be requested by Client later

# Resource Identifier :

Many resources available in the server, to identify the resources uniquely we go with Resource identifier

# Representation:

A resource is the actual data, Now this data can be represented as XML, HTML, or may be simple text

# Composition of URI :

URI = Scheme “://” authority / PATH

URI = Scheme “://” authority / PATH ? query

Scheme – protocol – http, https, ftp

Authority – main domain

path

query

# HTTP METHOD :

* HTTP Methods are used to perform Operation with REST API
* CURD Operations can be performed using http methods

HTTP GET :

* Mainly used to retrieve the information from the server
* Status Code :
  + 200 if resource found
  + 404 if resource not found
  + 500 if authentication errors

HTTP POST :

* Post methods are used to create a new resource into the collection of resources
* Status Code
  + 201 / 203 Created successfully

HTTP PUT :

* Update the existing Record
* main difference between the POST and PUT is POST is POST requests are made on resource collections and PUT Request are made on a single resource
* Status Code
  + 200 Created successfully (may vary)
  + 404 if resource not found
  + 500 if authentication errors

HTTP Delete :

* Used to delete the Record
* Status Code
  + 200 success code (may vary) / 203 / 204

Parameters

Path parameter [ / ]:

Variable parts of the URL path. These are typically used to point to a specific resource within a collection such as,

<https://jsonplaceholder.typicode.com/>

<https://jsonplaceholder.typicode.com/posts>

https://jsonplaceholder.typicode.com/posts/1

Query parameter [ ? ] :

Query parameter is used to filter / Sort the resources

<https://www.makemytrip.com/flight/search?itinerary=DEL-BLR-23/11/2022&tripType=O&paxTy>

API Tests using Post Man

* Download the latest software from
  + <https://www.postman.com/downloads/>
* Install the postman app

Using PostMan

* Creating Workspace
* Created Project
* Request
  + GET
  + POST
  + PUT
  + DELETE
* Parameters
  + Path parameter
  + Query parameter
* How to use output of one API in another API
  + Store the response output in a global variable

let response **=** pm.response.json();

pm.globals.**set**("uuid123", response.uuid);

* + use double curly braces with variable name in another API {{variable\_name}}
    - [http://httpbin.org/get?name=Aravinda&id=ab007&uuid={{uuid123}}](http://httpbin.org/get?name=Aravinda&id=ab007&uuid=%7b%7buuid123%7d%7d)

Trello – Web Automation

* is a project management application which allows user to create a project and maintain the day today activities on a daily basis
* Setup a Account
* Generate key and token to execute the apis
* <https://developer.atlassian.com/cloud/trello/guides/rest-api/api-introduction/#boards>
* Generate Key and Token
  + <https://trello.com/power-ups/admin>
  + click on new
  + provide the necessary information
  + generate key and token

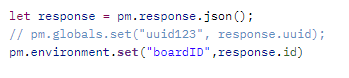
### API Reference

<https://developer.atlassian.com/cloud/trello/rest/>

# Adding output from API to Global Variables



# Adding output from API to Environment Variables



# Removing the environment Variable



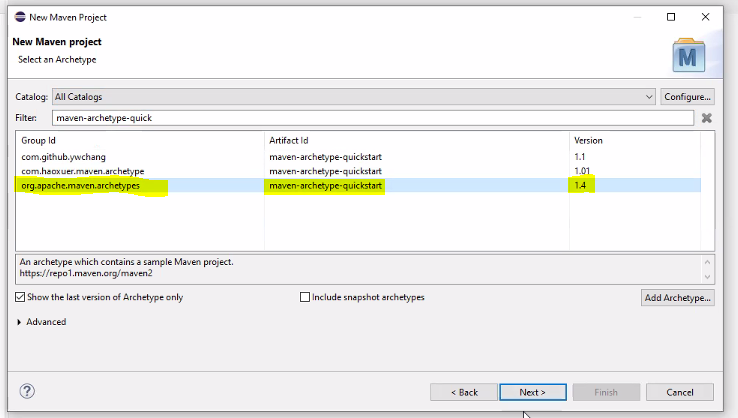
# Adding Dynamic data to the test using pre-request Script



Github API

* Create a github account
  + open <http://www.github.com>
  + click on sinup
  + follow the instructions to create account
* To use GitHub APIs
  + search for git hub developer api in google
  + <https://docs.github.com/en/rest>
* Generating token to execute the APIs
  + Click on your profile icon (present in top right corner)
  + Click on Settings
  + Scroll down and click on Developer Settings (present in bottom left corner)
  + Click on personal Access Token
  + Click on Generate New token
  + Copy and keep the token in notepad

Eclipse Setup with REST Assured lib

1. Create a Maven project, by selecting quickstart from the artifact
   1. 
   2. Update JDK and JRE to whatever is installed on the machine
   3. RightClick -> properties -> Java Compiler -> Select the Java suitable for your project and click on apply
   4. To update JRE, click on Java Build Path ->Libraries -> Remove the existing library -> Add library -> Java system library -> Click on installed JREs -> click on Add and browse (C:\Program Files\Java\jdk1.8.0\_201\jre) JRE from jdk location
2. Update the pom.xml with rest assured library
   1. <https://mvnrepository.com/artifact/io.rest-assured/rest-assured/5.3.0>

OR

* 1. <https://github.com/rest-assured/rest-assured/wiki/GettingStarted>

Writing First Test

Base URI -> domain name

* **given() – it is the first method to be called in REST Automation, using given we can pass below parameters**
  + **Request Header**
  + **Parameters**
  + **Cookies**
* **when() – represent the type of request we are making**
  + **get(resource\_path)**
  + **post(resource\_path)**
  + **put(resource\_path)**
  + **delete(resource\_path)**
* **then() – We can put Assertion statements here**
  + **Expected Result / Validate the Result**
* **extract() – To save the response for further calculation / validatin**
  + **To Save the whole response**

#### Simple Get Request

#### 

### Format :

1. **BDD Format – given when then**
2. RestAssured Class Format
   1. We need to create an object to RequestSpecification
   2. We need to get Response object
   3. ValidatableResponse and validate the result
3. Given Expect When Format

# Global Variables of REST Assured

# Validation in REST Assured

Hamcrest Package can be used to perform validation

Logging in REST

Using response object we can only print the response, if we want to save the response OR if we want to get complete Request and Response information like headers, cookies information then we need to go with Loggers.

Extracting Json Response

- Json Jayway path lib

- JsonPath from REST

- Validatable Response

# JsonPath from REST :

1. Get the response object

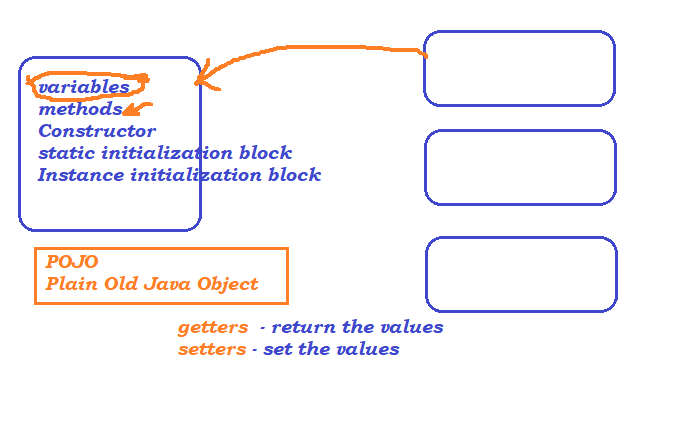
2. Create and object to JsonPath

3. use get Method to extract the values

# Json Jayway path lib

1. Update the pom.xml - <https://github.com/json-path/JsonPath>
2. validate the expression <https://jsonpath.herokuapp.com/>
3. Write the tests by passing valid expressions

Passing payload as object



1. Create a Java POJO class with all the Json Members
2. Create an object to POJO Class in the test where you want to send the payload as a body
3. Update pom.xml with jakson databind library <https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind/2.14.1>
4. pass the pojo object to the body method of post function

Assertion in REST using HAMCREST Lib

<http://hamcrest.org/JavaHamcrest/javadoc/1.3/org/hamcrest/Matchers.html>

### hasKey – to verify the key is present in a MAP

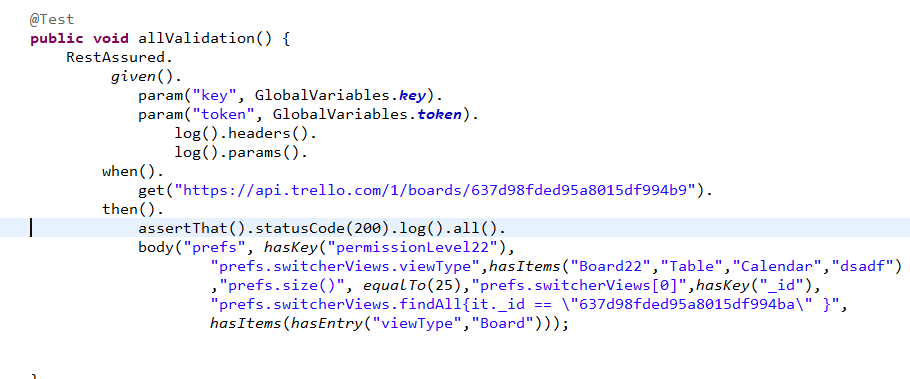
### hasValue - to verify the value is present in a MAP

### hasItem - to verify the Element is present in a Array of values

### hasItems- to verify the Elements is present in a Array of values

### to verify the key / value is present in a MAP

# SoftAsserts



# Validate the complete Json Response

1. we can make use of 3rd party library JSON Assert

2.

[https://github.com/skyscreamer/JSONassert](https://github.com/skyscreamer/JSONassert 3)

[3](https://github.com/skyscreamer/JSONassert 3). update pom.xml

<dependency>

<groupId>org.skyscreamer</groupId>

<artifactId>jsonassert</artifactId>

<version>1.5.1</version>

</dependency>

4. JSONAssert.*assertEquals*(expectedValues, actualValues, **true**);



Req and Res Specifications

When we are sending a request we may send headers, parameters , cookies along with the request.

Instead of sending the same repeated values in all the request we can go with specification object

Using Specification object we can reduce repetitive code. this also helps in maintenance. i.e., in future if we want to send more headers, parameters , cookies etc or if we want to reduce then we may have to update all the existing tests which is time consuming.

We can avoid the repetitive work using Specification Objects

Rootpath

rootpath to verify complex json

Response time

Faker API

<https://github.com/DiUS/java-faker>

Execute Tests From command prompt

<https://maven.apache.org/surefire/maven-surefire-plugin/examples/testng.html>