[AT Ralph] API Test

Target: Concepts about API test, SOA, restful (lots of concepts this session), try popular tools

For Challenge: Make automation API test suite by postman

For Cha-Challenge: Develop you own restful webservice

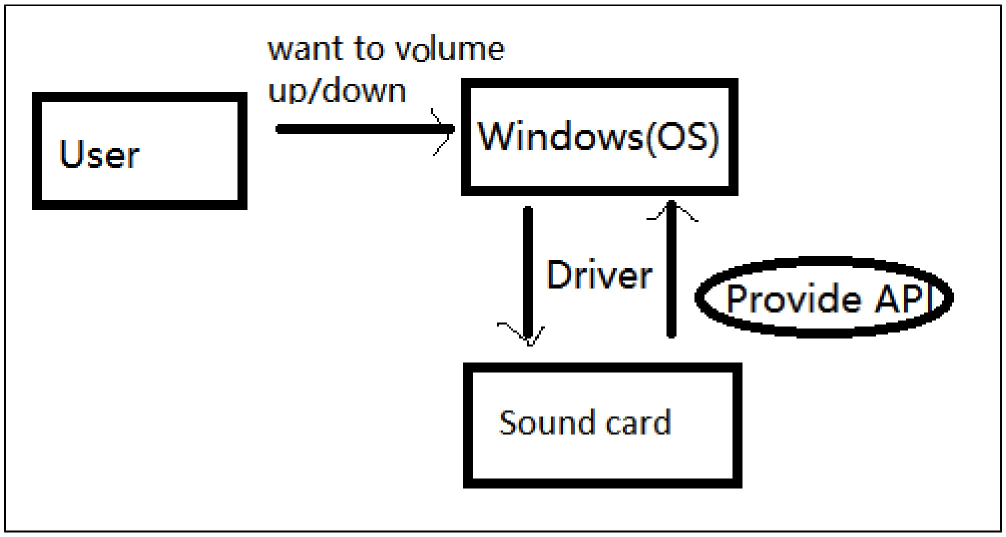
Raw doc(Chinese version): https://docs.google.com/document/d/1LtoPMYX-VVAt2dZ9lwyRvF5RscvxgNk21X7vJDfqxWY/view

Part 1: Introduce API

Wiki Definition:

*API means "Application Program Interface", In computer programming, an **application programming interface** (**API**) is a set of subroutine definit ions, protocols, and tools for building application software. In general terms, it's a set of clearly defined methods of communication between various software components.*

It is big concept, anything provide interface is API, for example, volume control flow:



Now we say API usually means the WebService API. API develop because clouding technology rapid development in previous years. Clouding really came to our life.

Cloud products:

Authentication / Authorization: stormpath

Analytics: Keen.io

CDN: CloudFlare, Fastly

CRM: Intercom

Dashboards: Ducksboard, Leftronic

Database: Bonsai, Heroku Postgres, MongoHQ, OpenRedis

Deployment: Heroku, Flynn

Email: Sendgrid

Log: Loggly

Monitor/Debug: New Relic, RunScope

Payment: Stripe, Coinbase

Storage: Amazon S3

Communicate: OpenCNAM, Twilio

Test: Travis CI

Example: play 3d game with 486

Part 2: Introduce SOA

Classification of Clouding:

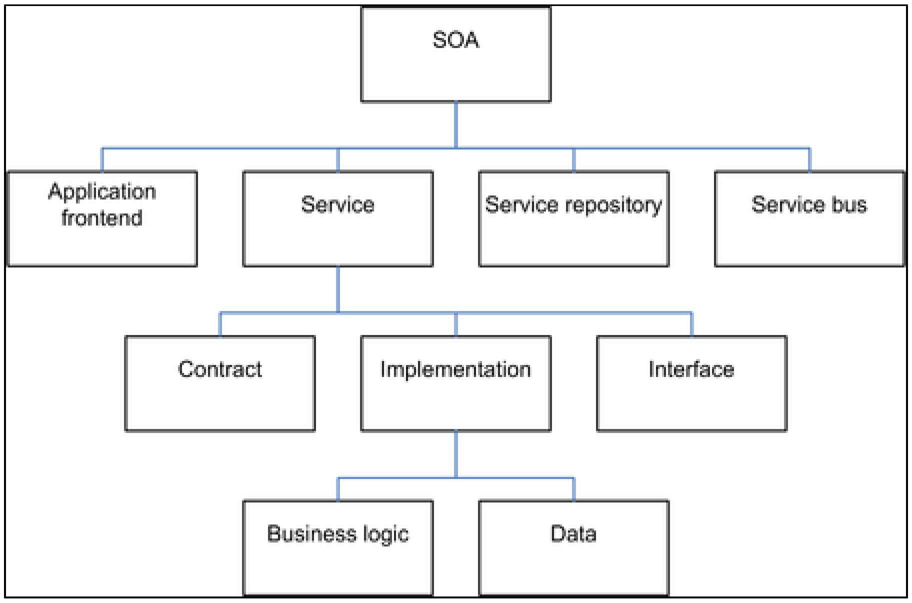
Infrastructure as a Service, IaaS: AWS, etc

Platform as a Service, PaaS: Google App Engine, etc

Software as a Service, SaaS: Salesforce, etc --> SOA related

Wiki Definition:

A ***service-oriented architecture*** *(**SOA**) is a style of software design where services are provided to the other components by* application components*, through a* communication protocol *over a network.*



It is a architecture, what we need is the SOA idea: What customers most need? services!

Example: relation of youtube and videos

SOA Story: **The Biggest Thing Amazon Got Right: The Platform**

https://gigaom.com/2011/10/12/419-the-biggest-thing-amazon-got-right-the-platform/

His Big Mandate went something along these lines:

1) All teams will henceforth expose their data and functionality through service interfaces.

2) Teams must communicate with each other through these interfaces.

3) There will be no other form of interprocess communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.

4) It doesn’t matter what technology they use. HTTP, Corba, Pubsub, custom protocols — doesn’t matter. Bezos doesn’t care.

5) All service interfaces, without exception, must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.

6) Anyone who doesn’t do this will be fired.

7) Thank you; have a nice day!

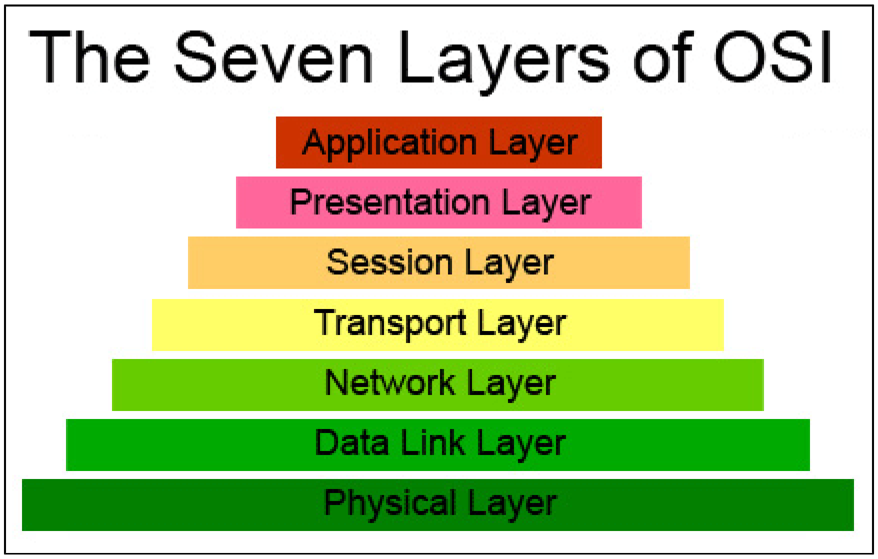
So,

In fact, SOA as a service-oriented architecture, is a software architecture design model and methodology. From a business point of view, all to maximize the "service" value as a starting point, SOA use of existing enterprise software system, re-integration and build a new software architecture. The software architecture can change with the business at any time the flexibility to combine existing services, the formation of new software, common services to the entire enterprise business system. Simple to understand, we can SOA as a modular component, each module can achieve independent functions, and the combination of different modules can provide different services, the interface between the modules to follow a unified standard, can be achieved Low-cost restructuring and reorganization. In the framework of SOA technology, can be a large and disorderly system integration into a comprehensive and orderly system, thereby increasing the business process in the application of business development flexibility, to achieve the greatest IT asset utilization.

Part 3: Restful API

So Based on SOA idea, software provide the services, no matter what language we use for server side, no matter which device you use for client(phone, web, desktop), we communicate by RPC on network and command protocol. Multiple services, Multiple clients.

Question: Which network layer is the best choice? which protocol is best choice?



Wiki Definition:

***Representational state transfer** (**REST**) or **RESTful** Web services are one way of providing interoperability between computer systems on theInt ernet. REST-compliant Web services allow requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations.*

For Example: Github URL https://github.com/ycj28c/WebServices

Restful Story: **Roy Thomas Fielding** (born 1965)

He is an American computer scientist, one of the principal authors of the HTTP (1.0, 1.1) specification, an authority on computer network architect ure, and co-founder of the Apache HTTP Server project.[1][2] Fielding works as a Principal Scientist at Adobe Systems in San Jose, California.[3]

Architectural Styles and the Design of Network-based Software Architectures

http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

(1) Resource - Each URI represents a resource

REST's name, "Presentation state transitions," omits the subject. "Presentation layer" actually refers to "resources" (Resources) of the "presentation layer." The so-called "resources", is an entity on the network, or a specific information on the network. It can be a piece of text, a picture, a song, a service, in short, is a specific reality. You can use a URI (Uniform Resource Locator) to point to it, each resource corresponds to a specific URI. To get this resource, access to its URI can be, so the URI has become the address of each resource or a unique identifier. The so-called "Internet", and the Internet is a series of "resources" interaction, call its URI.

(2) Representation - between the client and the server,the present layer of transmission for such resources

"Resource" is an information entity, it can have a variety of external manifestations. We present the form of "resources" in concrete terms, called its "representation". For example, the text can be used txt format, you can also use HTML format, XML format, JSON format, or even binary format; picture can be used JPG format, can also use PNG format. A URI represents only the entity of a resource and does not represent its form. Strictly speaking, the last ".html" suffix of some URLs is unnecessary because the suffix name indicates the format, which is the "presentation layer" category, and the URI should only represent the location of the "resource". Its specific form, should be in the HTTP request header and Accept-Content-Type field specified, these two fields is the "presentation layer" description.

(3) State Transfer - the client through four HTTP verbs, to operate the server-side resources, to achieve "State Transfer"

Access to a website, on behalf of the client and the server of an interactive process. In this process, is bound to involve changes in data and status. Internet communication protocol HTTP protocol, is a stateless protocol. This means that all the state is stored on the server side. Therefore, if the client wants to operate the server, it must be through some means, so that the server-side "State Transfer" (State Transfer). And this transformation is built on top of the performance level, so that is "the performance of state transformation." Client to use the means, can only be HTTP protocol. Specifically, that is, HTTP protocol, which means that the operation of the four verbs: GET, POST, PUT, DELETE. They correspond to four basic operations: GET to get resources, POST to create new resources (can also be used to update resources), PUT used to update resources, DELETE used to delete resources.

Restful API super popular, it is using everywhere, for Example: Slack, jira, confluence

Part 4: Rest HTTP protocol

So, to understand about the restful, we only need to understand HTTP.

What we know about HTTP: rfc2626

Well known status code
200 OK

401 Unauthorized

404 NOT FOUND

500 INTERNAL SERVER ERROR

https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

The protocol follow the rfc2626 is HTTP protocol, mainly talk about the HTTP verbs.

4 verbs

GET

PATCH/PUT

POST

DELETE

Other 2 not command verbs:

HEAD

OPTIONS

Compare with SQL command

GET(SELECT): get resource from server (one or multiple)

POST(CREATE): create new resource in server

PUT(UPDATE): update resource in server (client must provide the full resource information)

DELETE(DELETE): delete resource from server

Example: a zoo example

GET /zoos: list all the zoos

POST /zoos: create a new zoo

GET /zoos/ID: get information for particular zoo

PUT /zoos/ID: update all information for target zoo information

PATCH /zoos/ID: update some information for target zoo

DELETE /zoos/ID: delete a zoo

GET /zoos/ID/animals: list all the animals of the target zoo

DELETE /zoos/ID/animals/ID: delete target animal in target zoo

Real world example:

Insight new feature concurrency user, use insight-api in tomcat for user session control

```
select * from [redacted];
select * from [redacted];
```

```
$\> curl http://[redacted].1.[redacted].[redacted]/insight-api/get[redacted]Data/[redacted]/1

$\> curl -X
"DELETE" http://[redacted].1.[redacted].[redacted]/insight-api/delete[redacted]DataForUserId/[redacted]
```

POST and PATCH need payload, not present here

HTTPS:

`curl --insecure https://api.example.com/endpoint`

So,

How to do the API testing, you can curl every API if you want, but I suggest using tools.

Reference:

Introduction:

- Learn REST: A RESTful Tutorial

  http://www.restapitutorial.com/

- HTTP Status Codes

  https://httpstatuses.com/

- json-api

  http://jsonapi.org/

Books:

- Jersey 2.25.1 User Guide

  https://jersey.java.net/documentation/latest/user-guide.html

- Welcome to the REST CookBook

  http://restcookbook.com/