# Web services:

Web services are services that are exposed to internet, for programmatic access. They are online api.

Ex. Facebook and Twitter published online api so other application can post data on that.

Web Page: Is for Human consumptions, so we use CSS, HTNL to format

Web –api: It is not for human consumptions. IT is for other applications to read.

Response data is in XML/JSON.

Type of web-services: Rest and SOAP

REST ful web services:

(Representational State Transfer) Founder: Roy Fielding Simple Characteristics: Modern, light-weight

## Characteristics:

1. All web services, so exchange of data happens over the web, over HTTP

Client send HTTP request->so in return server send HTTP response.

1. Communication between client and server is form of some message. I.e. some time refer as protocol. In rest XML, JSON, text or any other as long as both can communicate.

SOAP: Uses standard protocol, Simple object access protocol.(name now discontinued)

1. Client and server can exchange any HTTP method. I.e Post, get, Put. But there are some guidelines regarding which should we used.
2. Service definition: details about web services like inputs return type. Mostly no service definition. SOAP have WSDL.(document which give details regarding SOAP web service).

Best REST web service should have less documentation.

SOAP: SOAP follows SOAP specifications, which are set of rule. Design by community and maintained by community. If SOAP web services do not follow any one rule not followed, it is not SOAP.

REST: It does not have any specifications or rule. It is concept, idea.

Style/Architecture + web services= Restful Web Services.

## Some basic concept of HTTP as HTTP is base of it.

HTTP (Hypertext transfer protocol): language or mechanism for communications.

Hypertext is structure form of text, which contain logical link to other text. Links called as Hyperlinks.

Language used to write text is HTML (Hypertext markup language)

## Develop:

1. Resource based address: URI are on based upon resources which give proper meaning.

Weatherapp.com/Zipcode/123456

1. HTTP methods: Get, Post, Put, Delete
2. Metadata: Status code, Response/message header

200 Successes

500 Server Errors

404 Not found on server

Header value of content type specify the communication way. Format of mesaage.

Text/xml for XML and Application/json

Client and server read header value, and then decide type of communications.

In restful web services, status code is send as this other applications is going to use this data, it is not designed for the human consumptions.

Content negotiation: What kind of data is required by client, because one method can return multiple types of data.

## Design:

### Resource based address/URI:

Think it as static pages.

#### Instance resource uri:

Mostly it contain noun, not verbs.

Ex. /profiles/{profile-name}

Profiles: consider it as directory so naming will be simple. Also give name for plural

Benefits: doesn’t care what framework is used. So even if internal framework changes no need to change api.

Suppose there are message and comments. To preserve relationship we can create like

/messages/1/comments/2

/messages/{ messages -ID}/ comments/{ comments -ID}

In short: single instance can be accessible by this URI

#### Collection resource URI:

/messages/1/comments

It will get all comments from messages.

### Filtering results or pagination:

/messages

It will give lot of data. This data should be filter for client.

WE can do this using query parameter.

Standard practice providing pagination like below

/messages? Offset=30&limit=10

Offset=30 : It is start point of message. i.e message will start from 30 (message id)

Limit=10: this is pagination. That is next 10 messages will be shown. It is page size.

#### Custom filtering:

/messages?year=2014

Same can be used with pagination, like

/messages?year=2014& Offset=30&limit=10

Operations: CRUD operation

Most common HTTP methods, Get, Post, Put, Delete

Rarely used: Head, Options

To get, delete, update message we know the message ID but to create we don’t know. So this done by application. Due to this for create resource always collection URI used.

So to create Messages we use

/Messages So on post request response will contain messageID

HTTP methods for the following operations will be same.

1. Get

/messages/10

1. Put

/messages/10

1. Delete

/messages/10

1. Post

/messages/10

## IDEMPOTENCE:

Read-only: get

Write: Post, Put, Delete

For continues request;

Repeatable: Get, Delete, Put It is call IDEMPOTENT

Non-repeatable: Post

So due to this resource creation should use POST. As resource creation is non idempotent.

Caching Get Response: After get request result is cached. So if request is repeated it does not compute all, it just gives cache.

Browsers refresh button:

If get, put, delete request then it do but if it is post then it will ask to confirm. E.g. Form submission.

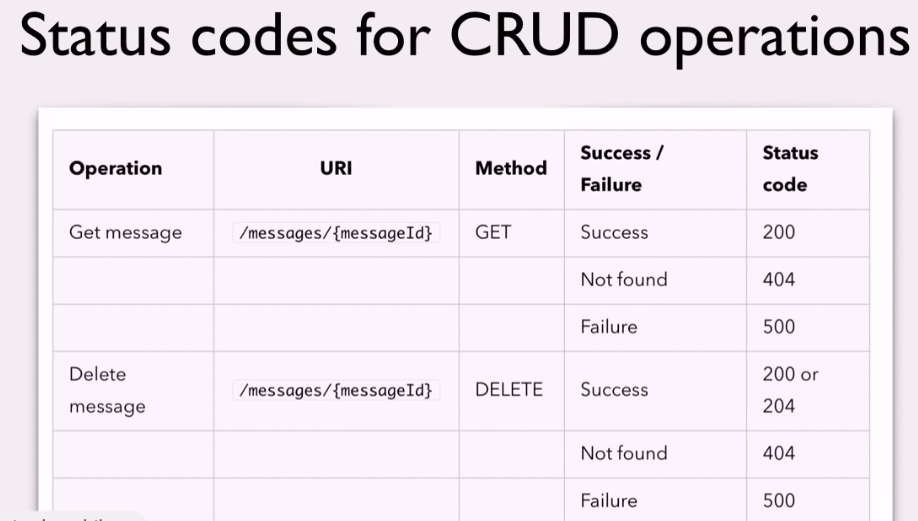
IDEMPOTENCE is property of certain operations, that can applied multiple times without changing the result beyond the initial application.

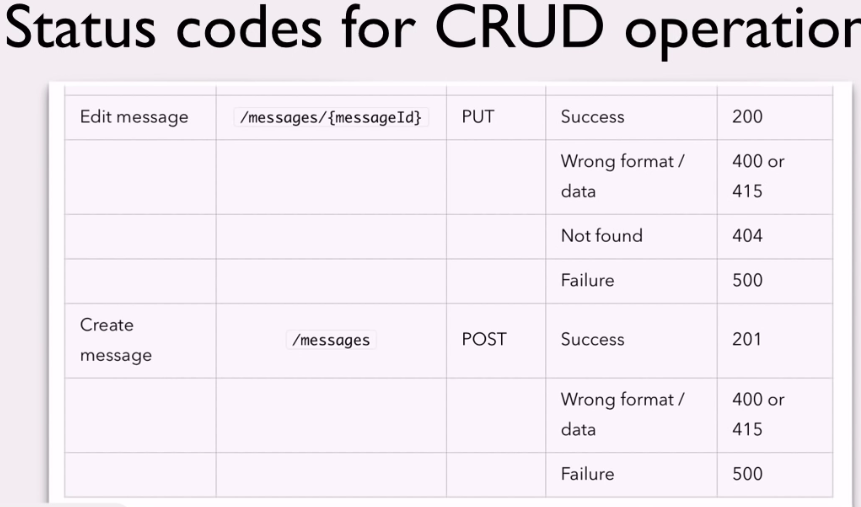
## REST Response:

Response header contains content type and status code. Which gives the information about content type of response i.e Json or xml or any other?

There are classification of response header into 5 .

1. 1XX: Informational
2. 2XX: Success ex. 200 OK, 201 Created, 204 not content(for delete)
3. 3XX: Redirection ex. 302 found, 307 temporary redirect,304 not modified (for repeated request if not any change)
4. 4XX:Client error if client request having error ex. 400 Bad request, 401 unauthorized, 404 not found, 415 unsupported media type, 403 forbidden
5. 5XX: Server error ex. 500 internal server error





## HATEOS:

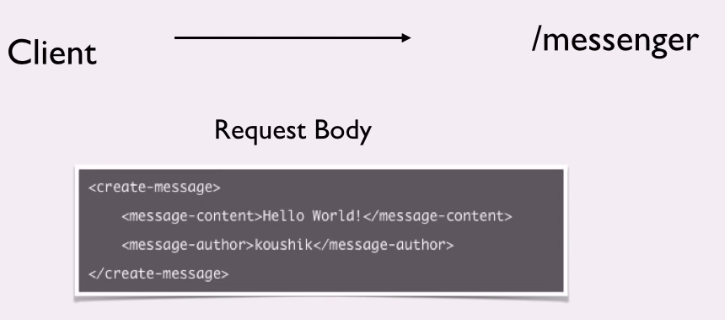
Hypermedia As the engine of application state.

## Richardson Maturity Model:

This is not rule but guidelines of design.

There are total four levels; one is for not restful api and other for restful api.

Level 0: not restful api.



Everything is given in request body. That is it is create-message request. So any http method can be used.

Also called swamp of POX (Plain old XML)

Level 1:

Individual URI for each resource

Message still contain which operation need to do. It can use any http method.

Level 2:

Individual URI for each resource

Different http methods are used for operations.

Standard HTTP methods are used. Like idempotent or non idempotent.

Level 3: Also called fully restful api

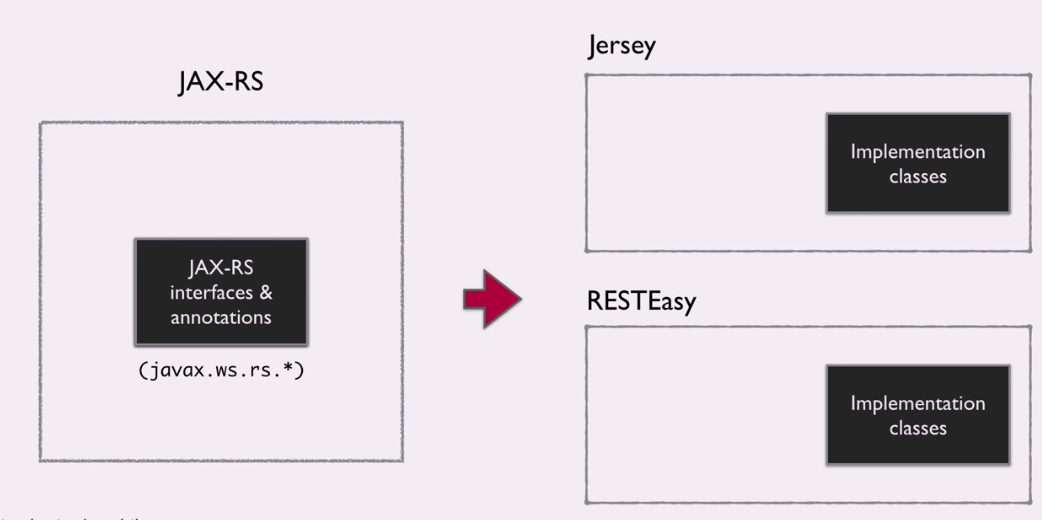
Implement HATOS.

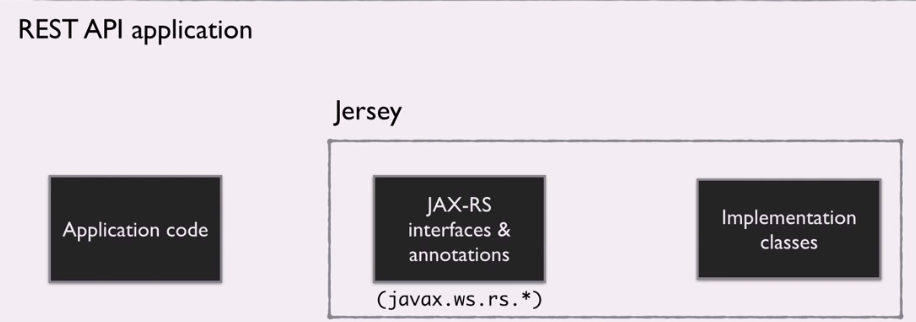
Response have links that client can use. As shown below commentsuri, likesUri



## JAX-RS:

Jersey: reference implementation





Now all jar internally have Jax-rs. Jax RS version 2.0.

Jax-rs provide some annotations to aid in mapping a resource class as a web resource.

Setup:

## Maven:

project management utility,

There are only two way to handle request in web applications servlet and filter

Web.xml have entry of URI and servlet. Also have entry of packages.

Once api called ex. ../webapi/messages then jersey will look entries of packages in web.xml , then from packages’ it will search for class which having annotation @path(“messages”)

So for working api following things should be there,

1. Package entry in Web.xml with above all entries. url-pattern also set in web.xml i.e /webapi/\*
2. Then @path, @get, @produces annotations.

## Rest Api Client:

Browser usually capable of handling get and post request

To more control and get more information we use REST Api client

Chrome’s plugin: POSTMAN

Jersey allows path annotation to method also. So it become easy to identify

Conversion from message instance to xml is done by Jax-p i.e JAVA . There is no convertor for JSON in java. So we need to add jar. So open pom.xml and add dependency for JSON conversion (i.e MOXY)

Libraries need to include for different conversion. Some classes included in this library implement perticlular interface, this are messageBodyReader and messageBodyWriter.

MessageBodyWriter: It is a class, this convert from java type to xml or Json.