1. Spring Transaction
2. Micro service transaction using saga design pattern to communicate using Kafka
3. CQRS, apicomposer How } design database of other services
4. Apache Kafka
5. Eureka Saver internal, @EnableEureka., configuration
6. spring cloud gate way, Load balance, circuit breaker
7. Rest template, Spring Data JPA, Exception Handling

Using Spring cloud Gateway

~~Architecture~~ of microservices communication using API Gateway

→ what is API gateway?

→ client side load balance is not there also no problem, Do we need really a client side load balancer?

Ans If the client is not implementing ribbon as a Load balancer at the client side how does the load on the multiple instances of the micro services will get distributed? How we ensure that the client will use a load balancer across at all? what is the guarantee the client is going to be used load balancer? to overcome this problem we can use spring cloud Gateway. (common required gateway configurations)

Micro services architecture

20 DEC 21

→ we have multiple instances of microservices let assume.

→ 3 micro services are there running on 3 different 2 different each instances of the, to two things like.

→ inventory-magment-service 2-instances it is running.

raw-material-management service., it also have two-two instances of the services.

In this way two-two instances of the services are running.

→ How does the client know which service is running on which node of the cluster? so we need eureka server (or) Discovery server

→ Now Eureka Discovery server is there, so all the micro services are discovered through micro services. Eureka discovery server (registered and discovery through the help of Eureka discovery server)

→ Now the client inorder to use this services how is going to know? → client side API will be there (Discovery client API).

→ client has to use discovery client API in order to access the microservices if we apply the API Gate way we no need to discover client API?

→ client trying to access directly to the micro service we need what Discovery client API.

It is directly accessing it dont know which

microservice running on which nodes of the cluster so the client must and should require discovery client API.

→ (Discovery client API is talk to discovery/Eureka server to identify where is my microservice service is running on the cluster.)

But if we are using API Gate way now, API Gate way access/acts what entry point of microservices of application.

Now client always talk to API Gate way, he dont need to talk to discovery server directly.

→ How many url-patterns for client client side for all the micro services i.e. only one url talk to the api gateway i.e. sufficient.

→ We try to access inventory-management service. Send the request to whoom API Gate way, raw-material management service also send the request whoom API Gate way. All services send the request to API Gateway.

why → API Gate way acts as common entry point.

Dont say front controller even though similar to front controller not used front controller here. Its will be called as API Gateway.

→ what is the purpose of API Gate Way?

It apply routing, transformation, security data aggregation, logging, monitoring, Caching every thing managed to whoom API gateway level.

→ A client send the request → (upon receiving request) → to API ~~to API Gate way~~ (→ who will takes care of forwarding corresponding micro services for our application now ?) ? API Gate way

→ Now How does API Gate way knows which microservice running on which nodes of the cluster ?

→ API Gate way will talks to the Eureka server and once identify inventory-management-service running on 2 nodes who will takes care of load-balancing the request on to the multiple nodes on that particular micro services over the cluster ? → API Gate way completly.

→ Additional responsible of API Gate way what → discovery and load balancing also. → the few more additional responsibilities by default will taking care. by whoom api Gate way.

→ So client side we need discovery client API → no client will never need to talk to Eureka server because if we talks the API Gate way → it takes care of routing the request to the corresponding microservices.

So client side load bancing not required here.

→ So client side load balancing and client side discovery API also not required. client will just talk to the API Gateway (we need client Application (client Program) to call the API Gateway)

→ How can we enforce

→ The server side what is the guaranty that the client is going to implement load balancer ? if the client is not implementing load balancer the load

the multiple instance of the microservices will not be distributed? How we take care of this?

Ans Using API Gate way.

How do we ensure that API Gate way is High Available?

Ans there is a chance in API Gate way might be down? How do we ensure API Gate way is Available? we need to run multiple instances of the API Gate way. So we need HA for API Gate way.

→ If we are having multiple API Gate ways now, How we need to distribute traffic across the multiple API Gate way. (Again we need Eureka server need or not?) what is the purpose of Eureka server discovering of the micro services ~~not API Gate way should be~~ not API Gate way discovering. ~~Only~~ ~~one~~ (do discover API Gate way) (API Gate way is fixed) i.e. that means ~~discovery~~ Eureka Server (or) we need lbr only. we no need at the client side (or) API Gate way we need lbr at the server side.

→ Eureka server purpose is what (which microservice is running on which nodes of the cluster) we dont know so Eureka server is required, so API Gate way already discovery through Eureka server. Why do we need to discovery Eureka server? Either we have client side load balanced or server side lbr.

→ lbr purpose is what if we give 2 nodes of 2 ip address now, first request send to first IP address, second request send to second IP address i.e. we want right. (Either one of the API Gate way should be able to receive the request) how API Gate way identify

~~Often~~ For routing the traffic between the API Gate ways now what we need Server side lbr ~~URL~~.

→ Either we have client side load balancer (or) we can have server side load balancer. server side we can have call lbr.

For routing the traffic between the API Gate way what we need → server side lbr url.

→ lbr purpose is what if we give 2 nodes into 2 ip addresses now first request will send first ip address and second request will send second ip address i.e. what we want either one of the API Gate way is able to receive the request and discover through the Eureka server and forward the request to the microservices.

→ Now even client side load balancer are not required. only server side lbr url we need to give to the client application. client simply access the microservices, event client dont know even he is is accessing the API Gateway.

→ why would learn ribbon (lbr), discovery server, discovery client API, feign client API?

→ There are internal applications are there which are organization for consuming microservices. These people wont go API Gate way.

→ External partner applications we need to going through API Gate way. But internal web application dont need to route through an api gate way, there is directly send the request to whom

reka server then these requires what if the directly sending request to the Eureka server what would be need discovery client and client side lbs also. so these are clitecture internal web application we called (both are required)

→ External partnerd application to consume our microservices we need the apigate way Architecture. (no lbs, no discovery client) directly send the request through the server side lbs, i.e. that takes care of distributing the traffic across multiple API Gate way instances. So that API Gateways will takes care of discovery which microservices with the help of Eureka sever and forward the request through the corresponding micro services.

→ so there are lot of 3rd party vendor libraries working with apigate way.

① Zall ②

in addition to spring provides cloud gateway which is implemented on reactive api. It supports 3 features.

① routing. ② transformation ③ security through spring security mode.

Architecture of API Gateway

→ If directly expose micro services to the client ⑧ there is no security, transformation, routing. like that, there are version

① mobile version 1.1 ② Browser version 1.2

we need routing.

→ when ever the client/external application sending request i.e. HTTP protocol based request (http rest end point) using rest end point only the request will send

→ when ever the client has send to the request ~~is send in~~ based on http protocol now → The incoming request is will be received by the → api ~~gateway~~ like dispatcher servlet ~~is there~~ (dispatcher servlet in web application here same way the ~~reactive~~ reactive dispatcher servlet will be there internally it will not expose directly it will work with spring boot)

→ The request will be routed ~~there~~ to whoom

Gateway Handler Mapping will be there → (How we have handler mapping MVC like)

→ The incoming request will be ~~routing~~ reactive to reactive servlet and it will forward the request to whoom → Gateway handler mapping. now the Gate way handler mapping send forward the request to whoom → (like controller)

now ~~the here~~ Here Gateway web Handler.

→ How many microservices are there on the server site →

→ lot of microservices are there in server side.

→ who has to take care of identifying for this request url which micro service has to be invoked

→ Gateway Handler.

→ Here Crate way handler mapping will calls the Crate way web handler (why does the name web handler because it is working based on HTTP Protocol

→ so (Crate way Handler mapping) will be received request (this way)

now → we try to apply routing routes what are those ① what is the incoming request url

for example /product → forward request to whom INVENTORY-MGMT-SERVICE, so for which all the requests

→ All incoming request of spring cloud crate way who will know → crate way Handler will know.

How does crate way Handler mapping know → we will configure → /product request coming means forward to incoming request to Inventory-MGMT service

→ If the the incoming request is /raw-material forward the request to RAW_MGMT_SERVICE.

→ where does the mapping between the request and corresponding microservice to be invoked will be there in → crate way Handler mapping.

→ upon the receiving request ↓ now → crate way handler mapping verifies weather the request url pattern with which request is send by the client is valid request or not. If it is a valid request means → it identify the corresponding service that has been invoked and forward the request to whom → crate way web handler.

→ The routing will be happening at the crate way handler mapping → (Now the crate way Handler upon on the receiving the request)

→ Now the Gateway web handler upon the receiving request will see what is that service is concrete url (or) its a load balancer url (i.e. discovery client url) → If it is

① discovery client url means → it talks to Eureka server passes what inventory-HTTP-service name can we tell me the endpoint names where is the service is running. ② then it will load balance it

~~③ It is load balanced~~ ;

→ what is Gateway web Handler doing routing

→ After routing we have to forward the request to whom corresponding microservices. before that can apply transformations, we can modify the http header, attached url, we can modified url, modify request body, response body every this will be possible at Gateway level. so that is where a series of filters will be applied

→ filters will be executed during the response time algo ↔ they can be multiple filters that can be attached between the microservice invocation and Gateway web handler.

→ Now all the filters will be executed in the sequential order (chain of filters) (request handler filter ⇄ response handler filters. During the time of filter the filter might receive the request change the url pattern (or) modify the header parameter, modify cookie format like and → modify the request body and forward the request to whom → corresponding microservice.

→ micro service after processing request will send the response back to whoom → filters →
response filters sends to gateway web handler
→ which will sends the response to the client application.

→ while mapping the request to the corresponding micro service now we can use n no of predicates are there.

→ Different types of predicates are there.
what are the predicates we can apply (gateway)
(after predicate, before predicate, between predicate, cookie predicate, query param predicate) → there no of predicate we need to apply as part of routing.

→ based on predicate conditions are being evaluated through appropriate routing of the request will be taken care by whoom → Gate way web handler.

this the architecture of the API Gateway (How does the api gate way will work

→ there are 3 major components are the in api gate way.
① routing ② predicates ③ filters.

→ transformation
→ the client will sending xml → where we convert xml to json logic filter
server understand json

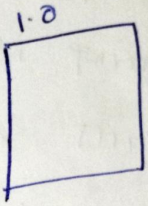
Core API gateway
we use Spring boot → all the things are auto configure. page no (180)

Gateway Handler mapping, Gateway web Handler

Autoconfigure.
→ we need to populate the Gateway Handler mapping
→ what routing rules with predicates/conditions.
i.e. if the request url so and so and forward the request to so and so. (page no (183))

we have inventory management services are there
1.0, and 2.0.

→ If the client is a mobile client
1.0 → micro services is



STOCKAPiSavice
-List

2.0 → we added location also 2.0.

→ mobile application is old 1.0
web application → modified version 2.0.

→ If the client is trying to access microservices
we need discovery Server. if it is mobile application
How does the client knows to forward the request through the
1.0 and if web application forward to the request through
the 2.0 microservice now. (don't know).
~~don't know~~.

→ Now here we registered url pattern of microservice
is INVENTORY-MGMT-SERVICE/STOCK/1.0/*
version no (url pattern appends)

secondary → INVENTORY-MGMT-SERVICE/STOCK/1.1/*

In eureka server both microservices will be regi
(1.0 an
→ so the client dont know which one ~~will be~~ they have to hit i.e. where we need to write what API gateway

→ only one url pattern in api gate way

/stock , (If the request is coming url pattern /stock receive the request then load balancer the request to whoom → Discovery client ~~gothrough~~ goes to eureka will identify micro services load balance to one micro service.

But we need to tell if the request url with which the request is coming is /stock forward request to whoom

- INVENTORY-MGMT-SERVICE

i.e. /stock = INVENTORY-MGMT-SERVICE

and apply the predicates.

→ query parameter if the request is we need verify

① platform = mobile → forward request to 1.0 url.

② platform = web → forward the request to 2.0 url.

→ where we write the predicate code is (route with predicate) → API gate way. so that acording API gate way route the request.

we want implement use Gateway.

Page no 180

2 ways we working with api gate way

① through configuration approach
② Programmatic API.

@Bean — (OP) Page no (180, 181)

```
public RouteLocator routeLocator(RouteLocatorBuilder builder)
{
    return builder.routes().route("stock list",
        r -> r.path("/stock/**").uri("lb://INVENTORY-MGMT-SERVICE")
            .build());
}
```

→ one of the route we are added → for every route one of the id will be added → what is the route id → stock list, route object r(a expression)

r.path("/stock/**").

→ Get the incoming request received by whoom api Gateway,
→ forward the request to whoom which uri

lb://INVENTORY-MGMT-SERVICE.build()

→ if the incoming request uri has /stock/** means → forward the request to the inventory management service (after it goes to discovery server → identified no of instances are running, fixed one of the instance → load balance it.

→ if we wont write lb: ____ (It will not understand the uri pattern we have to populated as discovery server uri → must and should use lb://here.

→ If it is direct uri means we ~~need to~~ can start with http://

→ Gateway web Handler, Gateway handler mapping who is populating routing → Auto configuration.

6x

→ the auto configuration class is going to create a class called RouteLocatorBuilder object → using RouteLocatorBuilder object we are going to populate

① routes (mandatory.
  ⊢ predicates (optional) } for the route matching with predicate apply
  ⊢ filter (optional) } this filter.

Here predicate and filter are optional.

routes are mandatory.

(1:∞:8)

inventory-mgmt-service

(1) tatamotors-eurekaserver.

(1) cd inventory-mgmt-service

(2) cd tatamotors-eurekaserver } cd cmd. → rename

(3) cd tatamotors-configserver

(1) java -jar target/tatamotors-eurekaserver-1.0.jar.
↳(rename → eurekaserver)

(2) java -jar target/tatamotors-configserver-1.0.jar →(rename configserver)

(3) java -jar target/inventory-mgmt-service-1.0.jar
↳ packaging application.

cd /inventory-mgmt/service >

(1) java -jar -Dserver.port=8088

target/inventory-mgmt-service-1.0.jar

→ two instances of inventory mgmt service is running

→ configserver, eurekaserver

→ client should not directly access through micro services we need API gateway.

→ we build api gateway

→ create new project tatamotors-cloudgateway

(StockAPI controller)

StockAPIgateway

(API gateway url)

//http://localhost:9900/inventory/stock

/television/available.

service url: http://localhost:8088/inventory

tata motors - cloud gateway (client side)

application.yml

server
  port: 9900

eureka
  client:
    register-with-eureka: false
    fetch-registry: true
    service-url:
      defaultZone: http://localhost:8761/eureka

spring:
  cloud:
    gateway:
      routes:
        - id: stocklist
          uri: lb://INVENTORY-MANAGEMENT-SERVICE
          predicates: (mandatory)
            - path: /inventory/**
          (or) http://localhost/8088/

→ circuitbreaker dependency

→ spring gateway reactive supports server

netty → web server → it does not work on tomcat
spring cloud gateway works on spring-webflux →