MicroServices Intra Communication

Limitation with Discovery Client, LoadBalancerClient

=> They can find and get producer MS ServiceInstance and other details from Eureka Server .. But they can not give http calls to interact with target/Producer MS

For that we need to use RestTemplate seperately

write

=> we need to code manually to find and get Target MS Service Instance

To overcome

these problems use Feign Client as Client Comp/Client Type comp

Feign Client /Open Feign (It is called abstract client)

==============

=> It is called abstract client becoz we just provide interface with method declaration and adding annotation.. But entire logic will be generated in the InMemory dynamic proxy class (Proxy pattern)

with

Using Feign Client

=>It is Combination Client i.e it takes the giveaget/producer Ms service Instance from Eureka server and also takes care of interacting target/Producer Ms by generating http calls that to with out wrting code.

The client comps of MS intra communication => a) DiscoveryCLient b) LoadBalancerClient c) FeignClient

No need of using RestTemplate separately here

=> It is internally Load Balancer Client i.e it gets Target/Producer Ms service Instance from the List of seerviceInstances which having Less Load Factor.

=> This mode of Client Comp development improves the productivity of the App.

=> Here we do not develop helper

client comp class for Consumer RestController rather

we develop Interface having @FeignClient("ServiceId") and the generated InMeory DyamicProxy class object will be injected to Consumer RestController.

=> While worokign with FeignClient we need to add 2 annotation/special annotations along with

regualar annotations in the Consumer App

a) @EnableFeignClients on the top main class along with @EnableEurekaClient /@EnableDiscoveryClient

b) @FeingClient on the top of interface for which dynamic Inmemory proxy class

will be generated.

Feign Client Inteface

must match with

======= Target MS service Id

@FeignClient("Billing-Service")

public interface IBillingServiceRestConsmer{

@GetMapping("/billing/info")

public String getBillingInfo();

>This interface can have

any name

Must match with the request path

Target MS

==========

ServiceInstance nanme:: Billing-Service

@RequestMapping("/billing") Provider/Producer/Server MS

|-->@GetMapping("/info")

}

गु

}

this signature

This method

(including global path) of Target MS method/opration)

must match with

can have any name

public String showPaymentOptions(){

target Ms method

signature

Example App Using Feign Client

============

step1) Develop Project as Eureka Server

=>dependencies :: Eureka service

of

=> add @EnableEurekaServer on the top main class => application.properties

server.port=8761

eureka.client.register-with-eureka=false eureka.client.fetch-registry=false

step2) Develop Project as Producer MicroService

=>dependencies :: web, EurekaDiscoveryClient.

=> add @EnableEurekaclient on the top of main class

=> application.properties

#Ms Properties

#Port number

server.port=9900

#Service id

spring.application.name=Billing-Service

#Eureka server publishing info

=> While working with maven based spring boot starter Project we can add just new starters on top of existing starters with out distrubing already added starters by using project popup menu (right click menu) .. But same is not possible while working with gradle based spring boot starter Project i.e while adding new starters in the middle of the project development we should add both old and new starters.

eureka.client.service-url.default-zone=http://localhost:8761/eureka

Provide serviceId:random number as the instance id

eureka.instance.instance-id=${spring.application.name}:${random.value}

=> Develop RestController having producer methods

package com.nt.controller;

import org.springframework.beans.factory.annotation.Value; import org.springframework.web.bind.annotation.GetMapping; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("/billing/api")

public class BillingServiceController {

@Value("${server.port}")

private int port;

@Value("${eureka.instance.instance-id}")

private String instanceId;

@GetMapping("/info")

public String getBillingInfo() {

return "we accept Card Payment, UPI Payment, NetBaking Payment, COD---> port::"+port+"----InstanceId::"+instanceId;

step3)

step5:

}

}

Develop Cosumer MS Project adding spring web, Eureka DiscoveryClient,open Feign

=> Add @EnableEurekaClient, @enableFeignClient annotations on top of main class

@SpringBootApplication

@EnableEurekaClient

@EnableFeignClients

public class SpringBootMsProj04ShoppingServiceConsumerApplication {

public static void main(String[] args) {

}

}

SpringApplication.run(SpringBootMsProj04ShoppingServiceConsumerApplication.class, args);

@EnableFeignClients

Scans for interfaces that declare they are feign clients (via org.springframework.cloud.openfeign. FeignClient @FeignClient). Configures component scanning directives for use with

org.springframework.context.annotation.Configuration

@Configuration classes.

=> add the following entries in application.properties

#Ms Properties

#Port number

server.port=6600

#Service id

spring.application.name=Shopping-Service

#Eureka server publishing info

eureka.client.service-url.default-zone=http://localhost:8761/eureka

=> Take an interface supporting FeignClient code as InMemory Dynamic Proxy class

package com.nt.client;

import org.springframework.cloud.openfeign.FeignClient; import org.springframework.web.bind.annotation.GetMapping;

@FeignClient("Billing-Service")

target MS

service ID

public interface IBillingServiceRestConsumer {

}

@GetMapping("/billing/api/info")

complete request path

of target MS service/operation method

public String fetchBillDetails();

can be any name.. No standards to fallow

=>Develop the Cosumer RestrController Injecting FeignCleint related Proxy object to

consume the target/Producer Ms services.

//RestController

package com.nt.controller;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.web.bind.annotation.GetMapping; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RestController;

import com.nt.client.IBillingServiceRestConsumer;

@RestController

@RequestMapping("/shopping/api")

public class ShoppingController {

@Autowired

private IBillingServiceRestConsumer consumer;

@GetMapping("details")

public String displayShopping Details() {

System.out.println("ShoppingController:: client comp class name::"+consumer.getClass());

return "Pongal Shopping for Family ...."+consumer.fetchBillDetails();

}

}

Execute the application

=====================

=>*Run Eureka server app*

=> *Run the Producer MS application for 2 or 3 times with different port numbers*

changed in the application.properties file

=> *Run the Cosumer Ms application*

=> Go to Eureka Server Home page and modify the Cosumer Service url

to as shown below

←**>http://192.168:1:236:6060/shopping/api/details (or) http://localhost:6060/shopping/api/details**

Pongal Shopping for Family ....we accept Card Payment, UPI Payment, NetBaking Payment, COD--->port::9902----InstanceId::Billing-Service:43c6cbdef49aebccda0ab868294114f4

Different pratices to develop Feign Client Interfaces

===================

==========

Producer Ms/Target Ms

======================

a)

ServiceId/applicaiton -name:: Vendor-Service

@RestController

@RequestMapping("/vendor")

public class VendorServiceController{

@GetMapping("/all")

public ResponseEntity<List<Product>>

getAllProducts(){

Feign Client Interface at Cosumer MS

a) @FeignClient("Vendor-Service")

public interface IVendorServiceConsumer{

@GetMapping("/vendor/all")

public ResponseEntity<List<Product>>fetchAllProducts(); (or)

@GetMapping("/vendor/all")

public List<Product>fetchAllProducts();

}

b) Service Id/app name :: Payment-Service

@RestController

@RequstMapping("/payment")

public class PaymentServiceController{

service id

@FeignClient("Payment-Service")

public interface IPaymentService RestConsumer{

@PostMapping("/payment/save")

@PostMapping("/save")

:

public String addCard(@RequestBody CardDetails details);

Assuming the front end app of Consumer MS is giving JSON inputs

public String saveCard(@RequestBody CarDetails details){

}

model class

}

@DeleteMapping("/payment/delete/{cardNo}")

public String removeCard(@PathVariable Integer cardNo); Assume Front End app

is not giving json data..

to this Cosumer MS

@DeleteMapping("/delete/{cardNo}")

public String removeCard

(@PathVariable Integer cardNo){

What is Spring webClient ?

Ans) if u r using spring webflux module (supports Rective Programming)

to develop the cosumer MS or Client app in Restfull env.. then we need to

use this WebClient instead of RestTemplate (Spring Rest module)

What is the diffrence b/w Normal java class and InMemory Proxy class?

Normal java class

HDD --->Hard Disk Drive

from HDD

.java (HDD) ----------> .class (HDD) ---------> JVM of JRE loads this .class file for execution [In the JVM Memory of RAM)

Proxy class (InMemory class)

compiled code

JVM of JRE loads this

for execution

(In the JVM Memory of RAM)

[In the JVM Memory of RAM)

Normal JavaApp/ ------> generates Proxy class source code----------> Proxy class compilation Container execution (In the JVM Memory of RAM)

(In JVM Memory of

the RAM)

Enable

While working with FeignClient comp based MS Intra Communication why we need to place @DiscoveryClient and @EnableFeignClients on the top of main class?

Ans) Basically our Consumer MS is client to Eureka server and must be registered with Eureka Server, For this we need to place @EnableDiscoveryClient annotation.. To make the Spring cloud module searching for @FeignClient interfaces for generating dynamic InMemory Proxy classeswe need to use @EnableFeignClients annotation