Declarative client: Feign client

The restTemplate is being used for making calls to other services but when used with loadbalancing API and CircuitBreaker APIs , it becomes cumbersome . NetFlix provides an easier-to-use client called Feign . Feign automatically integrates with load balancing and circuitBreaker APIs to provide load balancing and fallback mechanism .

RestTemplate drawback:

a.Need to be aware of the various method of the RestTemplate API to use it . b. A separate bean for load balancing . c . A separate service for circuit breaker is needed. d .The header details of a request from Zull (gateway API) are not forwarded to the other microservices using Rest Template

FEIGN Client:

Feign is a declarative client from NetFlix . We as developer declare the APIs for contacting other microservices . We define the rules in the form of our own interface . At runtime, Feign will create an implementation for our interface automatically .Thus with minimul code and self made interfaces , we can have greater control over how one microservices communicate with the other .

Feign client automatically uses circuit breaker and fallback for calls without the need for a separate service class by the help of appropriate dependency and configuration .

Q What is Sleuth ? sliuth

Sleuth is **a tool from Spring cloud family**. It is used to generate the trace id, span id and add these information to the service calls in the headers and MDC, so that It can be used by tools like Zipkin and ELK etc. to store, index and process log files.

Eg. If customer send request to banking application to see all the report like balance, history, credit status >> then one microservice is dependent on other microservice to fetch all the data >> so during this period to know about how many micro services request travels how long it will take to each single microservices . Request travel through microservice can be traced by traceId (this id is same for every microservces request travels) and staying time in microservice is SpanId , SpanId is different for each microservices that request travels

Q. What is ZipKin ?

Ans : Zipkin is an application that monitors and manages the Spring Cloud Sleuth logs of your Spring Boot application. To build a Zipkin server, we need to add the Zipkin UI and Zipkin Server dependencies in our build configuration file.

By adding appropriate dependency , we make sleuth send all its details to Zipkin server . The Zipkin server has an in memory DB where the details are stored . Zipkin has powerful UI application, which allows us to analyze the logs and take appropriate action .

Steps: to make ZipKin serve >> 1st : create one zipkin server application with required dependency 2nd : add required dependency in each client application 3rd: add properties in common application.properties file which is shared by all application ( see central configuration in git , I have make it’s document already )>> the third steps will help all application to connect to Zipkin server .