Spring - Mar 2

System

→ dont want everyone can access DB to do CRUD

→ server to provide API (permission)

→ multiple servers make net become bigger and bigger, like a graph

→ run out of control

→ cuz dependences become so complicated. When server shuts down, it will impact lots of other servers who depend on its service(APIs)

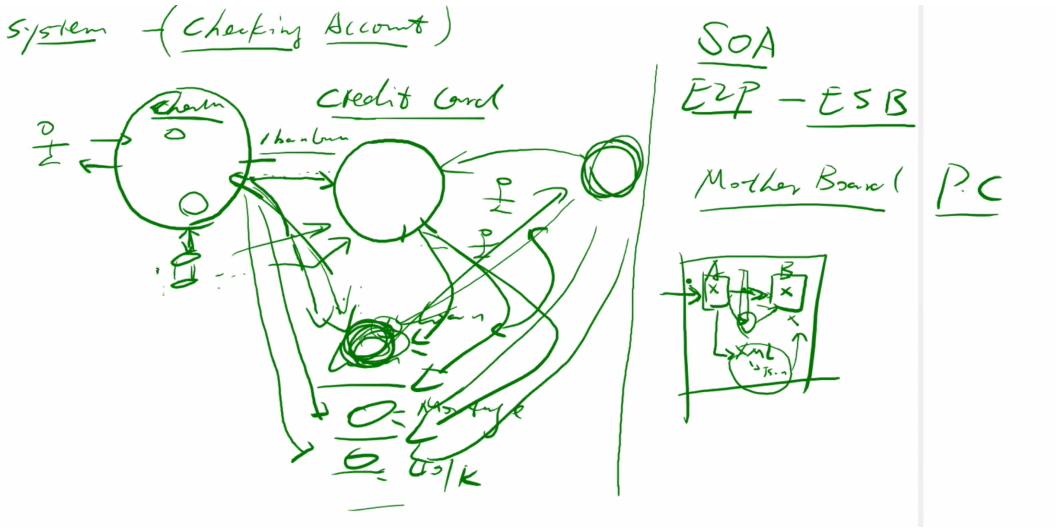
SOA(Service Oriented Architectures)

- we provide the services, as a basic unit
- manage the services

EIP (enterprise integration pattern) → ESB(Enterprise Service Bus)

ESB: it dont need nodes know each other (lose couple)

> instead, ESB looks like a controller, and it links every part together and every part dont know each other
>
> e.g
>
> ESB seems like a mother board of PC
>
> nodes dont care about where output data is going to
>
> ESB controls data flows
>
> When we want to change a data flow, we dont need change A or B node. Just to change ESB



MicroServices vs. SOA

Q: What is MicroServices?

A: We create many small PCIs, we call them services, they can be developed, deployed, tested, separately, individually, but they need to work together in order to achieve business logic, and this architecture should be robust, we call it resilient.

Resilient: Resilient means that like at any moment, you shut down some servers or you add some new servers, this whole system can handle it can tolerate those failures and some issues.

MicroServices

focus on split a very big service into servlet small services to MAKE whole system become more elastic

SOA

at very beginning, designed the services, designed input/output that they can talk each other. All the services together, they can do something

SOA dont care about the services are light or heavy

MicroServices

- **scalability**

- **flexibility → development**

  We have services, they could be very small(one/two api), which will be easy to add or delete. When the service become very busy, we can easily add more server to handle the increasing data flow

One solution → MicroServices

Spring Cloud

- Distributed/versioned configuration
- Service registration and discovery
- Routing API gateway

https://spring.io/projects/spring-cloud

---

the Two Sum project

zipkin is not supported by spring ?

Netflix provides a very good MicroServices solution with lots frameworks.

- Eureka(Service Discovery server/service)

Service discovery
is the process of automatically detecting devices and services on a network. Service discovery protocol (SDP) is a networking standard that accomplishes detection of networks by identifying resources.

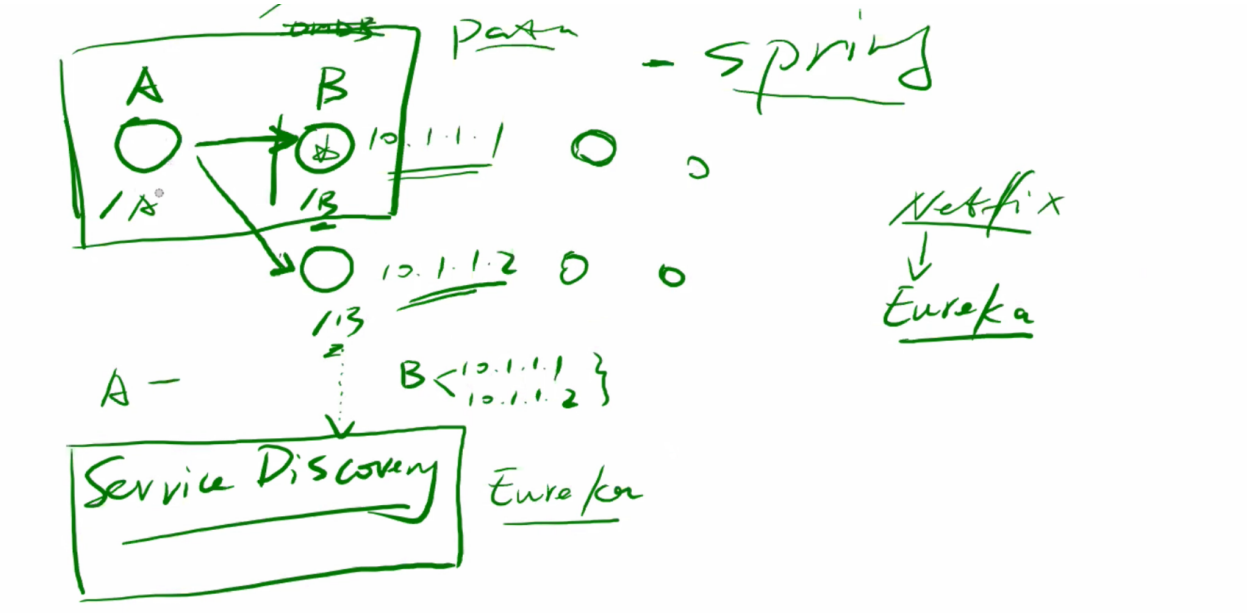
framework: e.g. consul

What's the Eureka do?

Just like a registry: every time a new microservice run, then it will come to Eureka to register itself

- record the multiple ip address/domain name for the same service

- health check: if a server/service does not have response for check, it will tag “not work” status



In this case:

Goal: A need to send request to B

Solution: instead of writing hard codes of IP address(B), directly on the URL say “B/APIs”. The “B” will be translated into one of IP addresses → framework called Ribbon

Ribbon: client side load balancer/balancing

When user to access a web, he/she does not know which accurate server they will hit. They hit load balancer first and transfer to certain server.

- of course, there will be lots of load balancers to be hit(套娃)

How to work?

Goal: Ribbon sends request to B

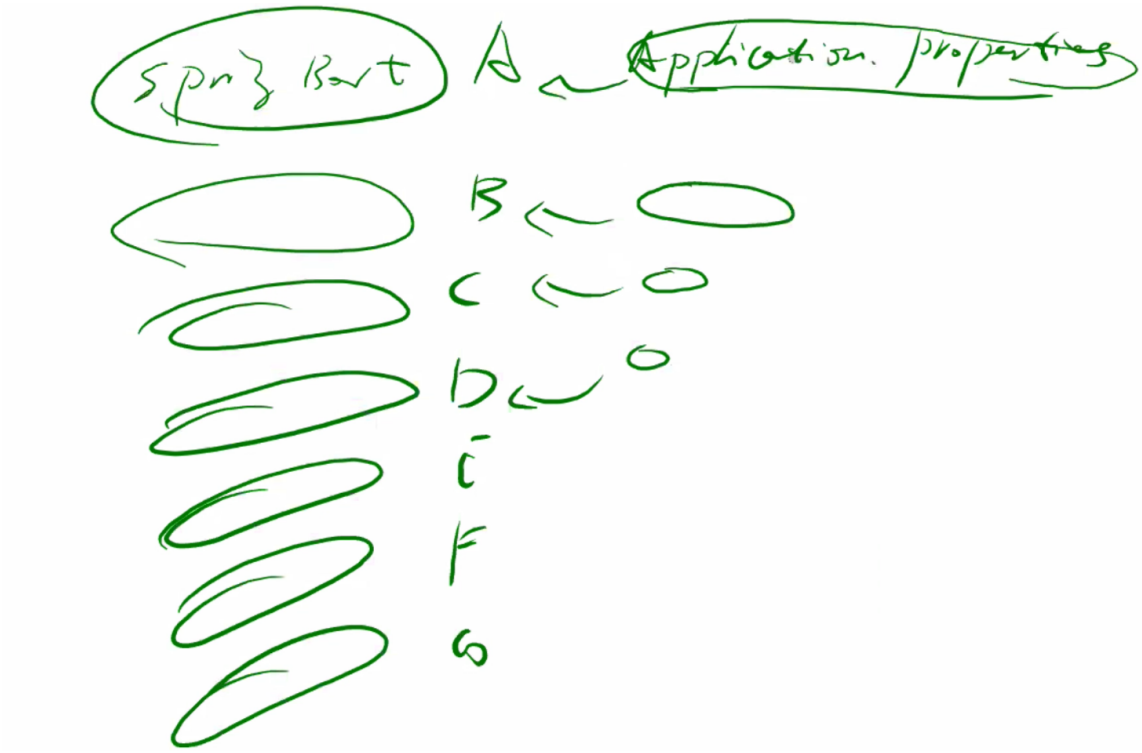
- before Ribbon sends request to B, it will ask Eureka (Service Discovery) to get all the IPs of B. and then Robbin will decide which IP it will use(round-robin pattern, choose one).

All the thing will dynamic by using Spring Cloud Netflix

- we just need the annotations, one or two annotations and all those functionalities will be applied to the code.

Configure server

Distributed/versioned configuration



In the case:

> There are lots of Spring boot projects on the left. Every project has an application.properties file. When we need to change running environments, it's tedious that changing every application.properties file one by one.

Spring Cloud Config Server

- host all the configuration files in it, it will be downloaded to certain projects separately

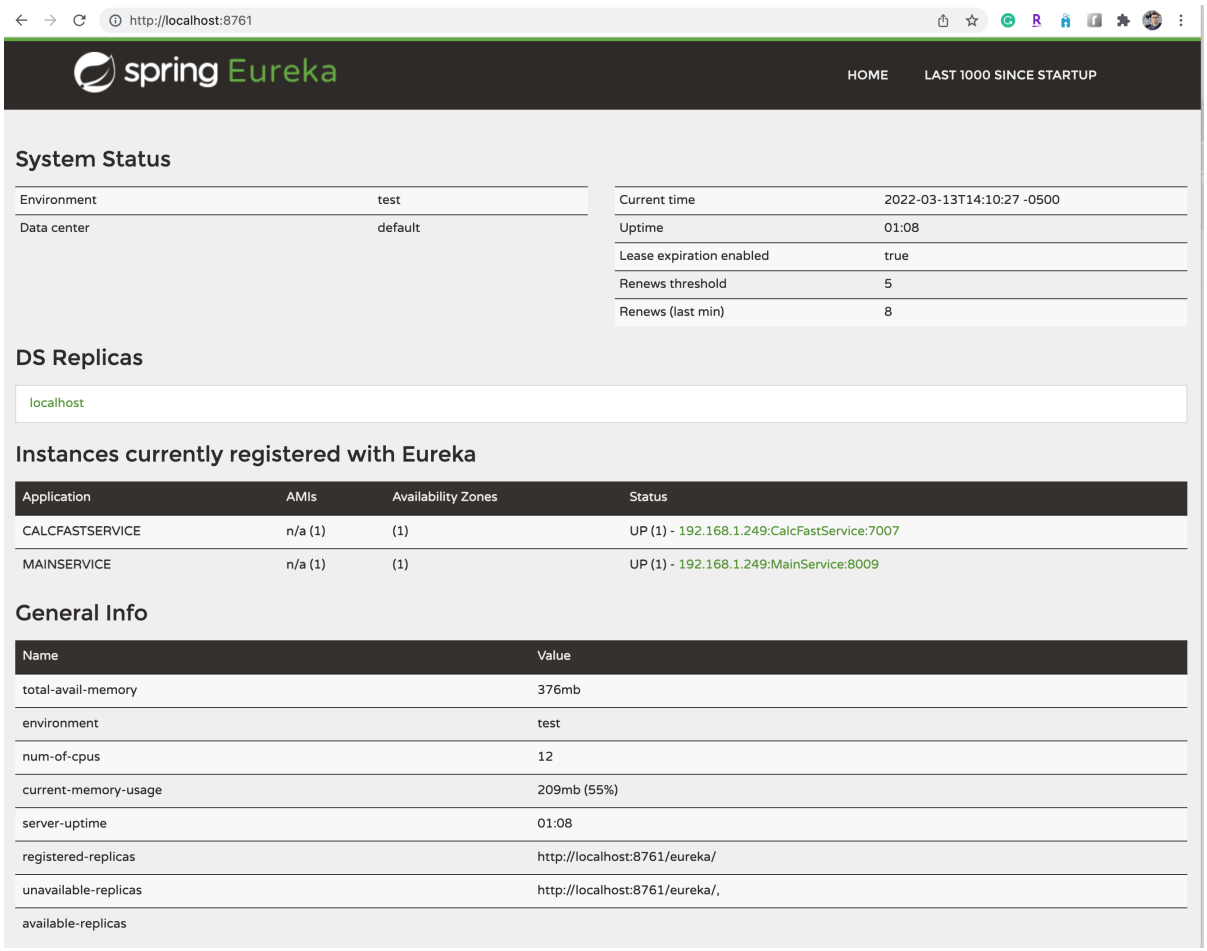
In configure Server, it has, for example, “A.properties”, “B.properties” and so on.

When A start running, it will go to Configure Server to download its own configuration and start service.

- spring-cloud-config-server dependency

Service_discovery → resource/bootstrap.properties

- spring.application.name=DiscoveryService (service name)
- spring.cloud.config.uri=http://localhost:8888 (it will download config from this server)
  - if not detected config server(could be not started/errors), the server will do Tomcat default start-process. (Running on 8080)



Instances currently: current servers register in Eureka

- the instance is equivalent to a service or node

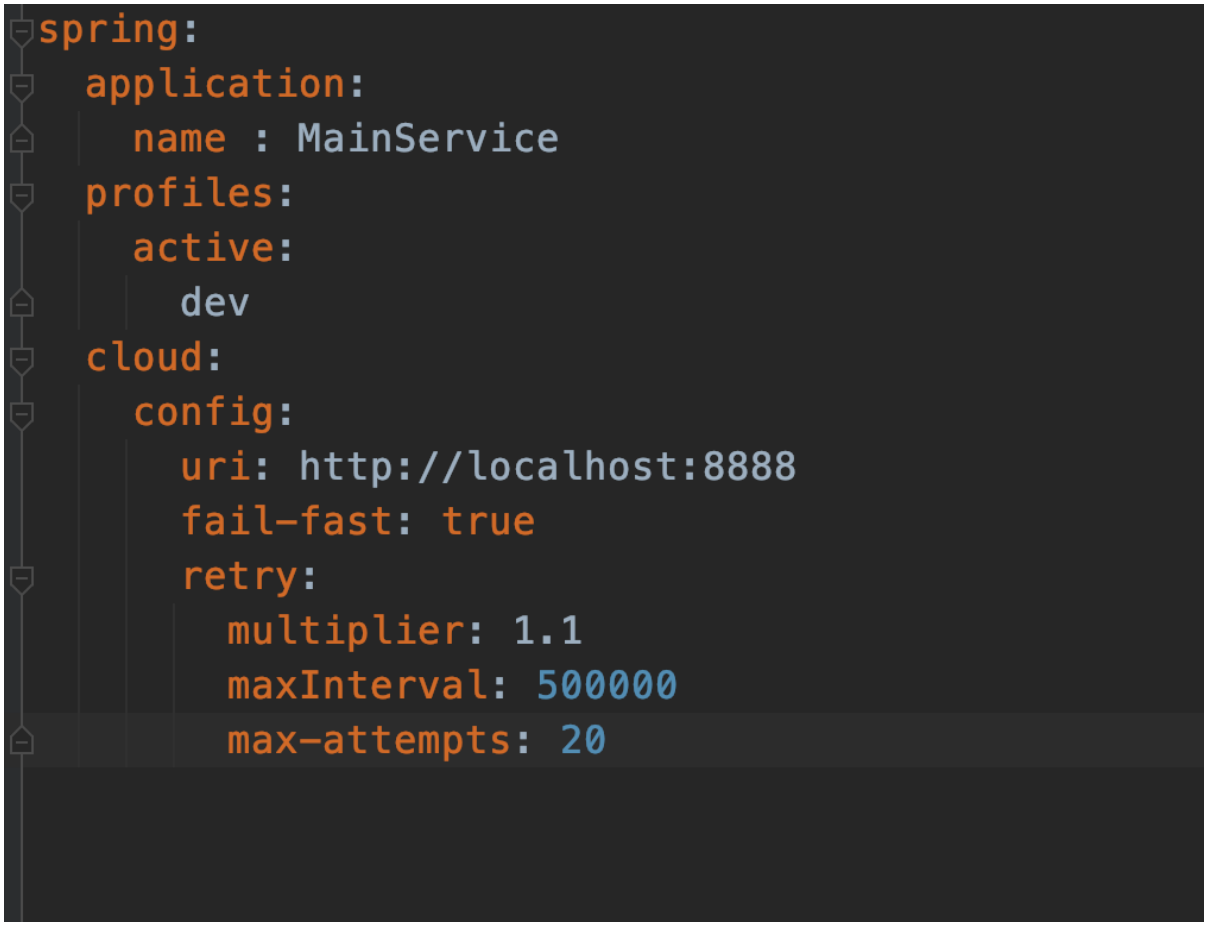
Dependency:

- `<artifactId>spring-cloud-starter-config</artifactId>` // download config data
- `<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>` // register itself to Eureka
- `<artifactId>spring-cloud-starter-netflix-ribbon</artifactId>` // load balance from A to B (A is itself, B is consumer)

.yml (yamo) and .properties are same

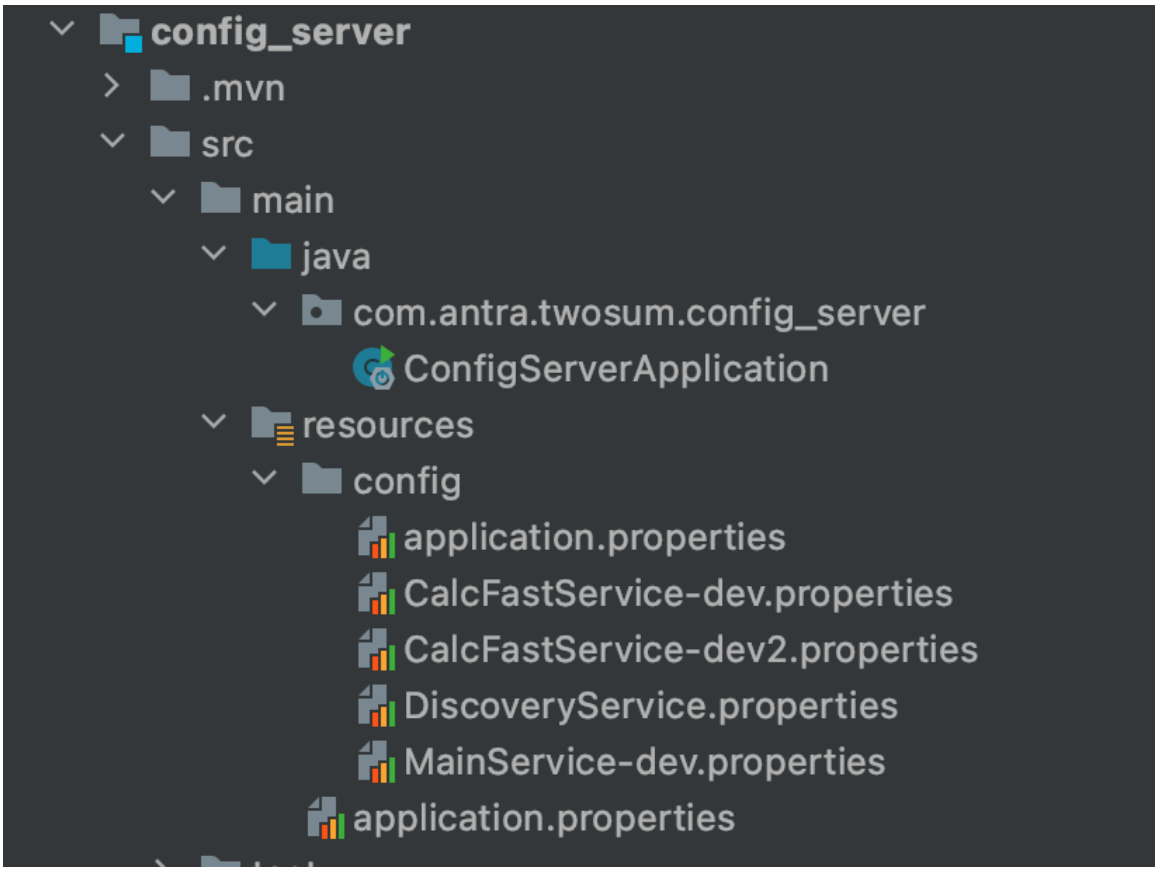
Spring boot can use both of them

.yml use tree or say indentation to declare them



In this case:

this service called MainService hyphen dev(Mainservice-dev)



if we have another environment, just use another environment name here

And it can be overrode by using command line

Eureka default port is 8761 (when we don't to set it)

@LoadBalanced

RestTemplate

```java
@LoadBalanced
@Bean
RestTemplate restTemplate() {
    LOGGER.info(globalProp);
    return new RestTemplate();
}
```

`import org.springframework.cloud.client.loadbalancer.LoadBalanced;`

Load Balanced resTemplate will understand the url: `"http://CalcFastService/calc"`

CalcFastService is not a domain name, which's just listed in bootstrap.yml config file, the declaration name of the service

```yaml
spring:
  application:
    name : CalcFastService
  profiles:
    active:
      dev
```

and also in instances list of Eureka server

@LoadBalanced Ribbon in the Service will rewrite the "CalcFastService" part (round robin)

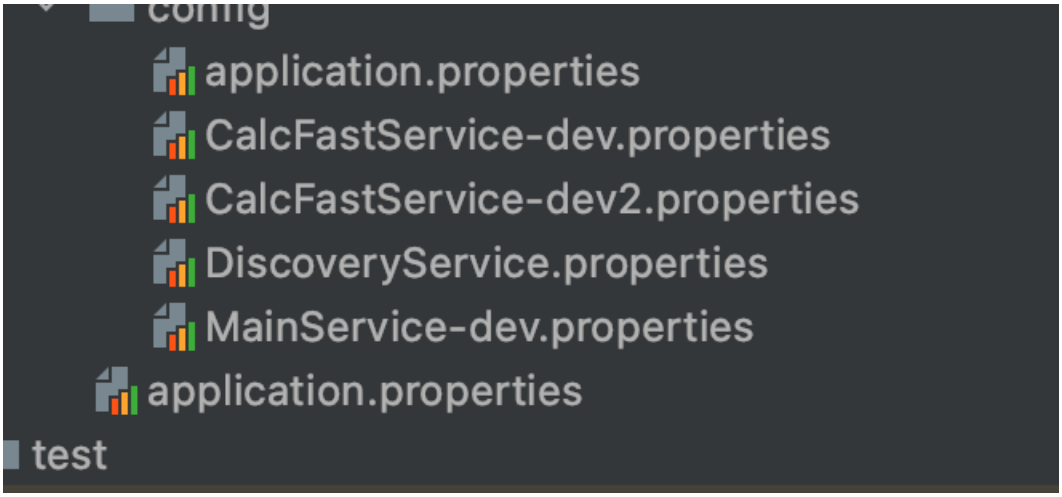
Application	AMIs	Availability Zones	Status
CALCFASTSERVICE	n/a (1)	(1)	UP (1) - 192.168.1.249:CalcFastService:7007
MAINSERVICE	n/a (1)	(1)	UP (1) - 192.168.1.249:MainService:8009

Now there is one A(MainService) and one B(CalcFastService).

Start one more B:

1. start B service again

2. use a different port



- use “CalcFastService-dev2” config file to start second B service with a different port
- change B’s “active” in config file with “dev2”

```
spring:
  application:
    name : CalcFastService
  profiles:
    active:
      dev2
```

PS: Cuz IntelliJ can not allow to run twice of a project

- use command line: mvn spring-boot:run
  - advantage is you don’t need to build it
- another:
  - mvn package → .jar/.war
  - java -jar xxx.jar

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
CALCFASTSERVICE	n/a (2)	(2)	UP (2) - 192.168.1.249:CalcFastService:7007 , 192.168.1.249:CalcFastService:7006
MAINSERVICE	n/a (1)	(1)	UP (1) - 192.168.1.249:MainService:8009

Attention:

- When a server cut off, Ribbon can not know immediately(it still cache its IP).
- When several times later, it will refresh its cache to delete the server.

So another feature:

- resilience to solve Robbin cashe sync problem