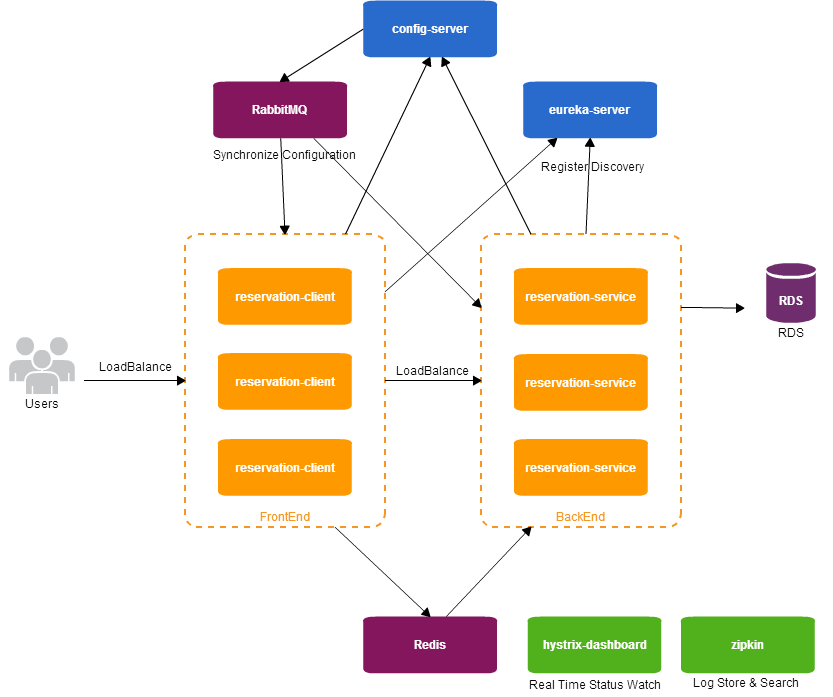
[**Microservice with SpringCloud**](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://samchu.logdown.com/posts/313414-microservice-with-springcloud&usg=ALkJrhjfdd3iGQMpTfaByl7dSRFq-xuT3g)

Microservice actually not very good management, you can imagine there are many routes, configuration, monitoring and other issues to engage in, but if you are using Java team, basically SpringCloud provide a very multi-component lets you use some simple profile Annotation can handle with Discovery, Synchronize Settings, Proxy, LoadBalance, Realtime Dashboards, LogAnalyzer and other mechanisms, such as the following figure.



Select the component, then to the side [http://start.spring.io/](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://start.spring.io/&usg=ALkJrhiikODQn3epmMh5s9nvEL_ACr8Unw) components checked that you need to download the project to choose to use this exercise SpringBoot1.3 & Gradle

[Use SpringBoot establish RestAPI](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://samchu.logdown.com/posts/313414-microservice-with-springcloud&usg=ALkJrhjfdd3iGQMpTfaByl7dSRFq-xuT3g#step1)   
[To establish a unified management of the Config Server](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://samchu.logdown.com/posts/313414-microservice-with-springcloud&usg=ALkJrhjfdd3iGQMpTfaByl7dSRFq-xuT3g#step2)   
[New Autodiscover Service](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://samchu.logdown.com/posts/313414-microservice-with-springcloud&usg=ALkJrhjfdd3iGQMpTfaByl7dSRFq-xuT3g#step3)   
[New Proxy Mechanism](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://samchu.logdown.com/posts/313414-microservice-with-springcloud&usg=ALkJrhjfdd3iGQMpTfaByl7dSRFq-xuT3g#step4)   
[Increase LoadBalance mechanism](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://samchu.logdown.com/posts/313414-microservice-with-springcloud&usg=ALkJrhjfdd3iGQMpTfaByl7dSRFq-xuT3g#step5)   
[To forward the request through redis](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://samchu.logdown.com/posts/313414-microservice-with-springcloud&usg=ALkJrhjfdd3iGQMpTfaByl7dSRFq-xuT3g#step6)   
[Increase reply message when a service interruption](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://samchu.logdown.com/posts/313414-microservice-with-springcloud&usg=ALkJrhjfdd3iGQMpTfaByl7dSRFq-xuT3g#step7)   
[Real-time monitoring](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://samchu.logdown.com/posts/313414-microservice-with-springcloud&usg=ALkJrhjfdd3iGQMpTfaByl7dSRFq-xuT3g#step8)   
[Log collection](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://samchu.logdown.com/posts/313414-microservice-with-springcloud&usg=ALkJrhjfdd3iGQMpTfaByl7dSRFq-xuT3g#step9)   
[References](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://samchu.logdown.com/posts/313414-microservice-with-springcloud&usg=ALkJrhjfdd3iGQMpTfaByl7dSRFq-xuT3g#step10)

**Use SpringBoot establish RestAPI**

First establish a **reservation-service** project to use the following components

| **Web** | **Data** | **Cloud Config** | **Cloud Discovery** | **Cloud Tracing** | **Cloud Messaging** | **Database** | **Ops** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Web | JPA | Config Client | Eureka Discovery | Zipkin | Stream Redis | H2 | Actuator |
| Rest Repositories | - | - | - | - | - | - | - |

First, put the following up these dependency notes, because being less than

build.gradle

dependencies {

/ \*

compile ( 'org.springframework.cloud:spring-cloud-starter-config')

compile ( 'org.springframework.cloud:spring-cloud-starter-eureka')

compile ( 'org.springframework.cloud:spring-cloud-starter-zipkin')

compile ( 'org.springframework.cloud:spring-cloud-starter-stream-redis')

compile ( 'org.springframework.boot: spring-boot-starter-actuator')

\* /

compile ( 'org.springframework.boot: spring-boot-starter-data-jpa')

compile ( 'org.springframework.boot: spring-boot-starter-data-rest')

compile ( 'org.springframework.boot: spring-boot-starter-web')

runtime ( 'com.h2database: h2')

testCompile ( 'org.springframework.boot: spring-boot-starter-test')

}

application.properties

server.port = 8025

ReservationServiceApplication.java

. package com example; import java.util.Arrays; import java.util.Collection; import javax.persistence.Entity; import javax.persistence.GeneratedValue; import javax.persistence.Id; import org.springframework.boot.CommandLineRunner; import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigure.SpringBootApplication; import org.springframework.context.annotation.Bean; import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.data. repository.query.Param; import org.springframework.data.rest.core.annotation.RepositoryRestResource; import org.springframework.data.rest.core.annotation.RestResource;SpringBootApplication public class ReservationServiceApplication {// start when pre-test plug informationBean CommandLineRunner runner (ReservationRepository rr) {return args -> {Arrays asList ( "Dr. rod, Dr Syer, Juergen, ALL THE COMMUNITY, Josh." split ( ",").) forEach (x -..> rr save (new Reservation (x))) ;; rr findAll () forEach (System out :: println);....};..} public static void main (String [] args) {SpringApplication run (ReservationServiceApplication class , args);}} // ​​this is to put your annotations directly into the Repository RESTful APIRepositoryRestResource interface ReservationRepository extends JpaRepository <Reservation, Long> {RestResource (path = "by-name") Collection <Reservation> findByReservationName (Param ( "rn") String rn);}Entity class Reservation {IdGeneratedValue private Long id; private String reservationName; public Reservation () {} public Reservation (String reservationName) {this reservationName = reservationName;.} public Long getId () {return id;} public String getReservationName () {return reservationName;}Override public String toString () {StringBuilder sb = new StringBuilder ( "Reservation {");.. sb append ( "id =") append (id );. sb append ( ", reservationName = '") append (reservationName) append (.. "'}"); return sb toString ();.}}

This is to help us build the generator test template

ReservationServiceApplicationTests.java

. package com example;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.springframework.test.context.web.WebAppConfiguration;

import org.springframework.boot.test.SpringApplicationConfiguration;

import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

RunWith (SpringJUnit4ClassRunner. Class)

SpringApplicationConfiguration (Classes = ReservationServiceApplication. Class)

WebAppConfiguration

public class ReservationServiceApplicationTests {

Test

public void contextLoads () {

}

}

Then use the GET to [http: // localhost: 8025 / reservations](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://localhost:8025/reservations&usg=ALkJrhiJSwWu7yJQesTmM14hQYKqMDbc9Q) to obtain information, to obtain this result

{

"\_embedded": {

"reservations": [

{

"reservationName": "Dr. rod",

"\_links": {

"self": {

"href": "http: // localhost: 8025 / reservations / 1"

},

"reservation": {

"href": "http: // localhost: 8025 / reservations / 1"

}

}

},

{

"reservationName": "Dr. Syer",

"\_links": {

"self": {

"href": "http: // localhost: 8025 / reservations / 2"

},

"reservation": {

"href": "http: // localhost: 8025 / reservations / 2"

}

}

},

{

"reservationName": "Juergen",

"\_links": {

"self": {

"href": "http: // localhost: 8025 / reservations / 3"

},

"reservation": {

"href": "http: // localhost: 8025 / reservations / 3"

}

}

},

{

"reservationName": "ALL THE COMMUNITY",

"\_links": {

"self": {

"href": "http: // localhost: 8025 / reservations / 4"

},

"reservation": {

"href": "http: // localhost: 8025 / reservations / 4"

}

}

},

{

"reservationName": "Josh",

"\_links": {

"self": {

"href": "http: // localhost: 8025 / reservations / 5"

},

"reservation": {

"href": "http: // localhost: 8025 / reservations / 5"

}

}

}

]

},

"\_links": {

"self": {

"href": "http: // localhost: 8025 / reservations"

},

"profile": {

"href": "http: // localhost: 8025 / profile / reservations"

},

"search": {

"href": "http: // localhost: 8025 / reservations / search"

}

},

"page": {

"size": 20,

"totalElements": 5,

"totalPages": 1,

"number": 0

}

}

A few simple lines can be converted into the database RESTful API rely mainly on theRepositoryRestResource

**To establish a unified management of the Config Server**

The establishment of a **config-server** project

Use the following components

| **Cloud Config** |
| --- |
| Config Server |

Mainly dependent on the spring-cloud-config-server

build.gradle

dependencies {

compile ( 'org.springframework.cloud:spring-cloud-config-server')

testCompile ( 'org.springframework.boot: spring-boot-starter-test')

}

application.properties

spring.cloud.config.server.git.uri = D: / springcloud / config-repo

server.port = 8888

* When the application is set spring.cloud.config.server.git.uri start reading from where profiles from Github, but also from local Git file

D: / springcloud / config-repo folder to put the files

application.yml

server.port: $ {PORT: $ {SERVER\_PORT: 0}}

info.id: $ {spring.application.name}

debug: true

spring.sleuth.log.json.enabled: true

logging.pattern.console: "% clr (% d {yyyy-MM-dd HH: mm: ss.SSS}) {faint} % clr (% 5p) % clr ($ {PID: - }) {magenta} % clr (---) {faint} % clr ([trace =% X {X-Trace-Id: -}, span =% X {X-Span-Id: -}]) {yellow} % clr ([% 15.15t]) {faint} % clr (% - 40.40logger {39}) {cyan} % clr (:) {faint} % m% n% wex "

reservation-service.properties

server.port = $ {PORT: 8000}

message = HELLO world!

spring.cloud.stream.bindings.input = reservations

Start Program

ConfigServerApplication.java

. package com example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.config.server.EnableConfigServer;

EnableConfigServer

SpringBootApplication

public class ConfigServerApplication {

public static void main (String [] args) {

. SpringApplication run (ConfigServerApplication class, args.);

}

}

Remember to includeEnableConfigServer can start ConfigServer features a

After starting Config-Server can access [http: // localhost: 8888 / reservation-service / master](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://localhost:8888/reservation-service/master&usg=ALkJrhjaeHG2jwJX1nq7Gfhu4z59XfYW9Q) can get the relevant information set

{

"name": "reservation-service",

"profiles": [

"master"

],

"label": null,

"version": "b017cbcb47700df4ffd7e824614532dd18128040",

"propertySources": [

{

"name": "D: /springcloud/config-repo/reservation-service.properties",

"source": {

"server.port": "$ {PORT: 8000}",

"spring.cloud.stream.bindings.input": "reservations",

"message": "HELLO world!"

}

},

{

"name": "D: /springcloud/config-repo/application.yml",

"source": {

"server.port": "$ {PORT: $ {SERVER\_PORT: 0}}",

"info.id": "$ {spring.application.name}",

"debug": true,

"spring.sleuth.log.json.enabled": true,

"logging.pattern.console": "% clr (% d {yyyy-MM-dd HH: mm: ss.SSS}) {faint}% clr (% 5p)% clr ($ {PID: -}) {magenta}% clr (---) {faint}% clr ([trace =% X {X-Trace-Id: -}, span =% X {X-Span-Id: -}]) {yellow}% clr ([% 15.15t]) {faint }% clr (% - 40.40logger {39}) {cyan}% clr (:) {faint}% m% n% wex "

}

}

]

}

Modify **reservation-service**

The original profile was changed to rely **Config-Server** provided

The original comment out depend added back spring-cloud-starter-config with spring-boot-starter-actuator

build.gradle

dependencies {

compile ( 'org.springframework.cloud:spring-cloud-starter-config')

/ \*

compile ( 'org.springframework.cloud:spring-cloud-starter-eureka')

compile ( 'org.springframework.cloud:spring-cloud-starter-zipkin')

compile ( 'org.springframework.cloud:spring-cloud-starter-stream-redis')

\* /

compile ( 'org.springframework.boot: spring-boot-starter-actuator')

compile ( 'org.springframework.boot: spring-boot-starter-data-jpa')

compile ( 'org.springframework.boot: spring-boot-starter-data-rest')

compile ( 'org.springframework.boot: spring-boot-starter-web')

runtime ( 'com.h2database: h2')

testCompile ( 'org.springframework.boot: spring-boot-starter-test')

}

Remember to update the look-dependent

The original application.properties renamed and changed to the following bootstrap.properties

bootstrap.properties

spring.application.name = reservation-service

spring.cloud.config.uri = http: // localhost: 8888

* spring.application.name use their own names, that time can be seen from the interface must correspond to the profile name
* The location spring.cloud.config.uri Config-Server

Increase from the controller can display information received Condif-Server

RefreshScope

RestController

class MessageRestControler {

Value ( "$ {Message}")

private String message;

RequestMapping ( "/ Message")

String message () {

. return this message;

}

}

As long as you can start [HTTP: // localhost: 8,000 / message](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://localhost:8000/message&usg=ALkJrhifNa49P7FtkuMN83YSJMd4AhdR0Q) ! HELLO world get information note Port changed Oh, since the outset acquire content reservation-service.properties from Config-Server, also made ​​message = HELLO world ! content to render.

PlusRefreshScope profile intention when there is a change, you can through the URL to trigger an update

curl -X POST 'http: // localhost: 8000 / refresh'

But how to keep in sync a study See Also   
[Synchronize all nodes using SpringCloud settings](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://samchu.logdown.com/posts/292630-use-springcloud-to-synchronize-all-nodes-configuration&usg=ALkJrhhRoAuOMuQUQkJmJapkg03_eIuMzg)

**New Autodiscover Service**

Establish a **eureka-server** project

Use the following components

| **Cloud Config** | **Cloud Discovery** |
| --- | --- |
| Config Client | Eureka Server |

New bootstrap.properties then delete unneeded application.properties, because we use Config Server configuration file provided

bootstrap.properties

spring.application.name = eureka-service

spring.cloud.config.uri = http: // localhost: 8888

Under the new eureka-service.properties to **Config-Server** profile path

eureka-service.properties

server.port = $ {PORT: 8761}

eureka.client.register-with- eureka = false

eureka.client.fetch- registry = false

# eureka.client.enabled = false

eureka.instance.hostname = localhost

# Eureka.client.serviceUrl.defaultZone = http: // $ {eureka.instance.hostname}: $ {server.port} / eureka /

In fact, hostname theoretically can not be added, but the following error message will appear, but it does not matter on the web is to say there is no mistake Client

2015-11-09 14: 53: 30.685 ERROR 21880 --- [trace =, span =] [nio-8761-exec-1] cneureka.resources.StatusResource: Could not determine if the replica is available java.lang.NullPointerException : null at com.netflix.eureka.resources.StatusResource.isReplicaAvailable (StatusResource.java:87) at com.netflix.eureka.resources.StatusResource.getStatusInfo (StatusResource.java:67) at org.springframework.cloud.netflix.eureka .server.EurekaController.status (EurekaController.java:68)

Part of the main program

EurekaServerApplication.java

. package com example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

EnableEurekaServer

SpringBootApplication

public class EurekaServerApplication {

public static void main (String [] args) {

. SpringApplication run (EurekaServerApplication class, args.);

}

}

After starting from [http: // localhost: 8761 /](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://localhost:8761/&usg=ALkJrhj_G7RruRLOQ9OPzX4Ric6kmWzUIw) services which are currently observed

Modify reservation-service automatic discovery of new clients

The spring-cloud-starter-eureka depend added back

build.gradle

dependencies {compile ( 'org.springframework.cloud:spring-cloud-starter-config') compile ( 'org.springframework.cloud:spring-cloud-starter-eureka') / \* compile ( 'org.springframework.cloud:spring -cloud-starter-zipkin ') compile (' org.springframework.cloud:spring-cloud-starter-stream-redis') \* / compile ( 'org.springframework.boot: spring-boot-starter-actuator') compile ( 'org.springframework.boot: spring-boot-starter-data-jpa') compile ( 'org.springframework.boot: spring-boot-starter-data-rest') compile ( 'org.springframework.boot: spring-boot -starter-web ') runtime (' com.h2database: h2 ') testCompile (' org.springframework.boot: spring-boot-starter-test ')}

Remember after the update starting dependence category plusEnableDiscoveryClient

EnableDiscoveryClient

SpringBootApplication

public class ReservationServiceApplication {

// Start when pre-test data plug

Bean

CommandLineRunner runner (ReservationRepository rr) {

return args -> {

Arrays. AsList ( "Dr. rod, Dr. Syer, Juergen, ALL THE COMMUNITY, Josh". Split ( ","))

. ForEach (x -.> Rr save (new Reservation (x))) ;;

. rr findAll () forEach (System out :: println.).;

};

}

public static void main (String [] args) {

. SpringApplication run (ReservationServiceApplication class, args.);

}

}

And then go back and start [http: // localhost: 8761 /](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://localhost:8761/&usg=ALkJrhj_G7RruRLOQ9OPzX4Ric6kmWzUIw) observation, you can find registered with Eureka list one more RESERVATION-SERVICE app name in DS Replicas Instances currently

**New Proxy Mechanism**

The establishment of a **reservation-client** project

Use the following components

| **Web** | **Cloud Config** | **Cloud Discovery** | **Cloud Routing** | **Cloud Circuit Breaker** | **Cloud Tracing** | **Cloud Messaging** | **Ops** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| HATEOAS | Config Client | Eureka Discovery | Zuul | Hystrix | Zipkin | Stream Redis | Actuator |

First temporarily zipkin up with hateoas annotation practice Proxy Mechanism

build.gradle

dependencies {compile ( 'org.springframework.boot: spring-boot-starter-actuator') compile ( 'org.springframework.cloud:spring-cloud-starter-config') compile ( 'org.springframework.cloud:spring-cloud -starter-eureka ') compile (' org.springframework.cloud:spring-cloud-starter-hystrix ') / \* compile (' org.springframework.boot: spring-boot-starter-hateoas ') compile (' org.springframework .cloud: spring-cloud-starter-zipkin ') \* / compile (' org.springframework.cloud:spring-cloud-starter-stream-redis ') compile (' org.springframework.cloud:spring-cloud-starter-zuul ') testCompile (' org.springframework.boot: spring-boot-starter-test ')}

The examples application.properties delete and add bootstrap.properties

bootstrap.properties

spring.application.name = reservation-client

spring.cloud.config.uri = http: // localhost: 8888

Start Program

ReservationClientApplication.java

. package com example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

import org.springframework.cloud.netflix.zuul.EnableZuulProxy;

EnableZuulProxy

EnableDiscoveryClient

SpringBootApplication

public class ReservationClientApplication {

public static void main (String [] args) {

. SpringApplication run (ReservationClientApplication class, args.);

}

}

EnableZuulProxy WithEnableDiscoveryClient increase in the original sample program   
After starting to [http: // localhost: 8050 / reservation-service / reservations](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://localhost:8050/reservation-service/reservations&usg=ALkJrhjQHRoKDamwgIReIMH_2CojlaOJTw) original reservation-service access to information on the following, you can obviously get out more than one reservation-service agent path

{

"\_embedded": {

"reservations": [

{

"reservationName": "Dr. rod",

"\_links": {

"self": {

"href": "http: // localhost: 8050 / reservation-service / reservations / 1"

},

"reservation": {

"href": "http: // localhost: 8050 / reservation-service / reservations / 1"

}

}

},

{

"reservationName": "Dr. Syer",

"\_links": {

"self": {

"href": "http: // localhost: 8050 / reservation-service / reservations / 2"

},

"reservation": {

"href": "http: // localhost: 8050 / reservation-service / reservations / 2"

}

}

},

{

"reservationName": "Juergen",

"\_links": {

"self": {

"href": "http: // localhost: 8050 / reservation-service / reservations / 3"

},

"reservation": {

"href": "http: // localhost: 8050 / reservation-service / reservations / 3"

}

}

},

{

"reservationName": "ALL THE COMMUNITY",

"\_links": {

"self": {

"href": "http: // localhost: 8050 / reservation-service / reservations / 4"

},

"reservation": {

"href": "http: // localhost: 8050 / reservation-service / reservations / 4"

}

}

},

{

"reservationName": "Josh",

"\_links": {

"self": {

"href": "http: // localhost: 8050 / reservation-service / reservations / 5"

},

"reservation": {

"href": "http: // localhost: 8050 / reservation-service / reservations / 5"

}

}

}

]

},

"\_links": {

"self": {

"href": "http: // localhost: 8050 / reservation-service / reservations"

},

"profile": {

"href": "http: // localhost: 8050 / reservation-service / profile / reservations"

},

"search": {

"href": "http: // localhost: 8050 / reservation-service / reservations / search"

}

},

"page": {

"size": 20,

"totalElements": 5,

"totalPages": 1,

"number": 0

}

}

**Increase LoadBalance mechanism**

Modify **reservation-client** project

First dependent hateoas added back

build.gradle

dependencies {compile ( 'org.springframework.boot: spring-boot-starter-actuator') compile ( 'org.springframework.cloud:spring-cloud-starter-config') compile ( 'org.springframework.cloud:spring-cloud -starter-eureka ') compile (' org.springframework.cloud:spring-cloud-starter-hystrix ') compile (' org.springframework.boot: spring-boot-starter-hateoas ') / \* compile (' org.springframework .cloud: spring-cloud-starter-zipkin ') \* / compile (' org.springframework.cloud:spring-cloud-starter-stream-redis ') compile (' org.springframework.cloud:spring-cloud-starter-zuul ') testCompile (' org.springframework.boot: spring-boot-starter-test ')}

Then add a controller

RestController

RequestMapping ( "/ Reservations")

class ReservationApiGatewayRestController {

Autowired

LoadBalanced

private RestTemplate restTemplate;

RequestMapping ( "Names")

public Collection <String> getReservationNames () {

ParameterizedTypeReference <Resources <Reservation >> ptr =

new ParameterizedTypeReference <Resources <Reservation >> () {};

ResponseEntity <Resources <Reservation >> responseEntity =

.. this restTemplate exchange ( "http: // reservation-service / reservations",

. HttpMethod GET, null, ptr);

Collection <String> nameList = responseEntity

. GetBody ()

. GetContent ()

. Stream ()

. Map (Reservation:: getReservationName)

. Collect (Collectors toList ().);

return nameList;

}

}

In fact, here you can guess how it is used, then the following section is Java8 syntax Oh.

After the restart from [http: // localhost: 8050 / reservations / names](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://localhost:8050/reservations/names&usg=ALkJrhj55YNT7tplHY4LgxfZ4kg7EerOpg) tried to crawl data, you can achieve the desired result is a list of names

[

"Dr. rod",

"Dr. Syer",

"Juergen",

"ALL THE COMMUNITY",

"Josh"

]

**To forward the request through redis**

Modify **reservation-client** project

To modify build.gradle the spring-cloud-starter-stream-redis add to the mix

build.gradle

dependencies {

compile ( 'org.springframework.cloud:spring-cloud-starter-config')

compile ( 'org.springframework.cloud:spring-cloud-starter-eureka')

/ \*

compile ( 'org.springframework.cloud:spring-cloud-starter-zipkin')

\* /

compile ( 'org.springframework.cloud:spring-cloud-starter-stream-redis')

compile ( 'org.springframework.boot: spring-boot-starter-actuator')

compile ( 'org.springframework.boot: spring-boot-starter-data-jpa')

compile ( 'org.springframework.boot: spring-boot-starter-data-rest')

compile ( 'org.springframework.boot: spring-boot-starter-web')

runtime ( 'com.h2database: h2')

testCompile ( 'org.springframework.boot: spring-boot-starter-test')

}

Top of the main program to addEnableBinding (Source.class)

EnableZuulProxy

EnableBinding (Source. Class)

EnableDiscoveryClient

SpringBootApplication

public class ReservationClientApplication {

public static void main (String [] args) {

. SpringApplication run (ReservationClientApplication class, args.);

}

}

Plus the Controller

Autowired

Output (Source. OUTPUT)

private MessageChannel messageChannel;

RequestMapping (Method = RequestMethod. POST)

public void write (RequestBody Reservation r) {

this messageChannel send (MessageBuilder withPayload (r getReservationName ()) build ()...)..;

}

Modify **reservation-service**

Modify build.gradle the spring-cloud-starter-stream-redis add to the mix

build.gradle

dependencies {

compile ( 'org.springframework.cloud:spring-cloud-starter-config')

compile ( 'org.springframework.cloud:spring-cloud-starter-eureka')

/ \*

compile ( 'org.springframework.cloud:spring-cloud-starter-zipkin')

\* /

compile ( 'org.springframework.cloud:spring-cloud-starter-stream-redis')

compile ( 'org.springframework.boot: spring-boot-starter-actuator')

compile ( 'org.springframework.boot: spring-boot-starter-data-jpa')

compile ( 'org.springframework.boot: spring-boot-starter-data-rest')

compile ( 'org.springframework.boot: spring-boot-starter-web')

runtime ( 'com.h2database: h2')

testCompile ( 'org.springframework.boot: spring-boot-starter-test')

}

Then in the main program plusEnableBinding (Sink.class)

EnableBinding (. Sink class)EnableDiscoveryClientSpringBootApplication public class ReservationServiceApplication {// start when pre-plug test dataBean CommandLineRunner runner (ReservationRepository rr) {return args -.> {Arrays asList ( "Dr. rod, Dr Syer. , Juergen, ALL THE COMMUNITY, Josh "(,")) forEach split. ".... (x -.> rr save (new Reservation (x))) ;; rr findAll () forEach (System out :: println );};..} public static void main (String [] args) {SpringApplication run (ReservationServiceApplication class, args);}}

Add a message to the access point

MessageEndpoint

class MessageReservationReceiver {

Autowired

private ReservationRepository reservationRepository;

ServiceActivator (InputChannel = Sink. INPUT)

public void acceptReservation (String rn) {

. this reservationRepository save (new Reservation (rn)).;

}

}

Then back to the profile data Config-Server folder plus spring.redis.host, because it is to send and receive via redis so of course is to give the position to use

application.yml

spring.redis.host: "localhost"

Here testing it, but also put on a redis too big a fuss, directly Docker to Run

Vagrantfile.proxy

VAGRANTFILE\_API\_VERSION = "2"

Vagrant.configure (VAGRANTFILE\_API\_VERSION) do | config |

config.vm.box = "ubuntu / trusty64"

config.vm.provision "docker"

config.vm.provision "shell", inline:

"Ps aux | grep 'sshd:' | awk '{print $ 2}' | xargs kill"

config.vm.provider: virtualbox do | vb |

vb.name = "redis"

vb.gui = $ vm\_gui

vb.memory = $ vm\_memory

vb.cpus = $ vm\_cpus

end

config.vm.network: forwarded\_port, guest: 6379, host: 6379

config.ssh.username = "vagrant"

config.ssh.password = "vagrant"

end

Vagrantfile

# - \* - Mode: ruby ​​- \* -

# Vi: set ft = ruby:

# Specify Vagrant version and Vagrant API version

# Vagrant.require\_version "> = 1.6.0"

VAGRANTFILE\_API\_VERSION = "2"

ENV [ 'VAGRANT\_DEFAULT\_PROVIDER'] = 'docker'

$ Vm\_gui = false

$ Vm\_memory = 2048

$ Vm\_cpus = 2

Vagrant.configure (VAGRANTFILE\_API\_VERSION) do | config |

"." Config.vm.synced\_folder, "/ vagrant", disabled: true

config.vm.define "redis" do | v |

v.vm.provider "docker" do | d |

d.name = "redis"

d.image = "redis"

d.ports = [ "6379: 6379"]

d.vagrant\_vagrantfile = "./Vagrantfile.proxy"

end

end

end

After allowing two files then perform the same folder

vagrant up redis --provider = docker

Well, the program with the environment are good, then call up the program, followed by POST, new data from the reservation-client - write database> reservation-service -> redis

curl -X POST -H "Accept: application / json" -H "Content-Type: application / json" -d '{ "reservationName": "Red Johnson"}' 'http: // localhost: 8050 / reservations'

Re-view [http: // localhost: 8050 / reservations / names](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://localhost:8050/reservations/names&usg=ALkJrhj55YNT7tplHY4LgxfZ4kg7EerOpg) you can see more Red Johnson's name

Why do this? I think for the amount of sudden, sometimes even if your application can expand horizontally, but no time extension is GG ...

**Increase reply message when a service interruption**

Sometimes there will still be accidents, but there is a problem if you ran a strange mistake possible for the front-end surprise, or a little camouflage look, you can make no sense of abnormal client

Modify **reservation-client** project

Start program increaseEnableCircuitBreaker, then add on the need to accomplish thisHystrixCommand (fallbackMethod = "getReservationNamesFallback") when he failed to specify methods you will use to reply to the tip getReservationNamesFallback

EnableZuulProxyEnableBinding (. Source class)EnableCircuitBreakerEnableDiscoveryClientSpringBootApplication public class ReservationClientApplication {public static void main (String [] args) {SpringApplication run (ReservationClientApplication class, args.);.}}RestControllerRequestMapping ( "/ reservations ") class ReservationApiGatewayRestController {AutowiredLoadBalanced private RestTemplate restTemplate;.AutowiredOutput (Source OUTPUT) private MessageChannel messageChannel;..RequestMapping (method = RequestMethod POST) public void write (RequestBody Reservation r) {this messageChannel. send (MessageBuilder withPayload (r getReservationName ()) build ()...);} public Collection <String> getReservationNamesFallback () {return Collections emptyList ();.}HystrixCommand (fallbackMethod = "getReservationNamesFallback")RequestMapping ( "names ") public Collection <String> getReservationNames () {ParameterizedTypeReference <Resources <Reservation >> ptr = new ParameterizedTypeReference <Resources <Reservation >> () {};.. ResponseEntity <Resources <Reservation >> responseEntity = this restTemplate exchange (" http .: // reservation-service / reservations ", HttpMethod GET, null, ptr); Collection <String> nameList = responseEntity getBody () getContent () stream () map (Reservation:..: getReservationName) collect (Collectors... . toList ()); return nameList;}}

After the reservation-client start the reservation-service turned off that way usually occurs 500 apps like abnormal return, but you can try to call [HTTP: // localhost: 8050 / reservations / names](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://localhost:8050/reservations/names&usg=ALkJrhj55YNT7tplHY4LgxfZ4kg7EerOpg) , you can find that you just get an empty set, the front end does not affect your program

[]

**Real-time monitoring**

Establish a **hystrix-dashboard** project

Use the following components

| **Cloud Config** | **Cloud Discovery** | **Cloud Circuit Breaker** |
| --- | --- | --- |
| Config Client | Eureka Discovery | Hystrix Dashboard |

Like removing application.properties, because we are now mainly rely on setting provides Config-Server, and then add bootstrap.properties

bootstrap.properties

spring.application.name = hystrix-dashboard spring.cloud.config.uri = http: // localhost: 8888

Then the main program enabledEnableHystrixDashboard

HystrixDashboardApplication.java

. package com example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.netflix.hystrix.dashboard.EnableHystrixDashboard;

EnableHystrixDashboard

SpringBootApplication

public class HystrixDashboardApplication {

public static void main (String [] args) {

. SpringApplication run (HystrixDashboardApplication class, args.);

}

}

Then add in the Config-Server profile folder

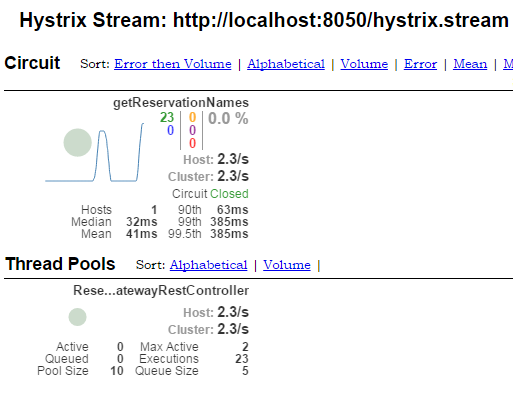
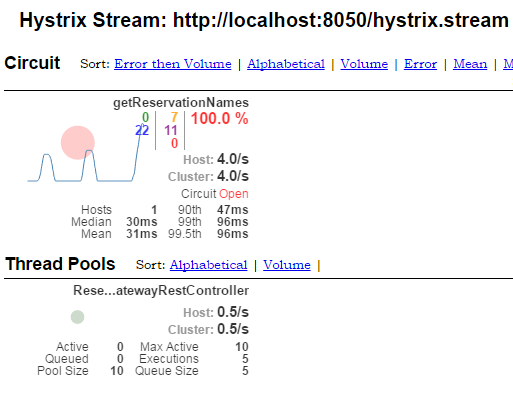
hystrix-dashboard.properties

server.port = $ {PORT: 8010}

Then start hystrix-dashboard, then access

[http: // localhost: 8050 / hystrix.stream](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://localhost:8050/hystrix.stream&usg=ALkJrhg-YNvKbqk2BeG6BDlk0k7o3UjuWw)   
You can see that we've been spit reservation-client data

Then paste the URL in the top middle of the page the following fields   
[http: // localhost: 8010 / hystrix.html](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=http://localhost:8010/hystrix.html&usg=ALkJrhiQoB7Y0i7HzyIPYjDbXxx5hJDDMg)

Then press the Monitor Stream button, you can see a monitor interface when backend success or failure you can instantly found to   
  


**Log collection**

Sometimes know which link in the wrong, but no record is still very difficult to find the problem, spring-cloud-starter-zipkin can record data transfer between each service, the official website [https://twitter.github.io/zipkin/ Quickstart.html](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=https://twitter.github.io/zipkin/Quickstart.html&usg=ALkJrhg9cYwCIICJuoeuwYi9-48aNba6qg)

This twitter thing to do, it should also be installed very annoying, try to see another day, and then look at how to use Docker Top about   
<https://github.com/joshlong/cloud-native-workshop/blob/master/bin/zipkin/docker-compose.yml>

Said first part of the program reservation-service, reservation-client previously commented out dependencies spring-cloud-starter-zipkin then added back within two projects registeredBean, the program is complete end

Bean

AlwaysSampler alwaysSampler () {

return new AlwaysSampler ();

}

zipkin study this thing to see it again.

To be completed

[https://github.com/openzipkin/zipkin](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=https://github.com/openzipkin/zipkin&usg=ALkJrhhiax5tfI76r4uQx4AjWZxr4uxx7Q)

References:   
[Getting started with Spring Cloud by Josh Long](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=https://www.youtube.com/watch%3Fv%3DSFDYdslOvu8&usg=ALkJrhiVqMCg1O8ewkzTiuoJXgjFF_Homg)

This is references Lab [YouTube Getting Started with Cloud Spring](https://translate.googleusercontent.com/translate_c?depth=1&hl=it&rurl=translate.google.it&sl=zh-CN&tl=en&u=https://www.youtube.com/watch%3Fv%3DcCEvFDhe3os&usg=ALkJrhhfEHkq7KKQfaj0Lz6NXwxuGwC9Xg) , Thank you Dr. Dave Syer, Josh Long.