## Spring Cloud项目

### 建立主体工程；maven，打包方式pom；配置pom.xml

1. 配置properties；java版本、spring-boot版本、字符集等。
2. 配置dependencyManagement，通过${spring-boot.version}获取properties配置，统一管理依赖版本。
3. 配置dependencies，导入依赖包：spring-boot，spring-cloud，swagger。版本通过${...}获取。

### **建立子类工程；module，打包方式jar；导包。**

选择simple project，否则可能结构不全。

#### 注册中心

导入spring-cloud-starter-eureka-server；启动类添加@EnableEurekaServer；

配置application.yml：

server:

port: 8761

eureka:

instance:

hostname: localhost

client:

registerWithEureka: false

fetchRegistry: false

serviceUrl:

defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/

#### **服务提供者**

导入spring-cloud-starter-eureka；启动类添加@EnableEurekaClient

配置application.yml：

eureka:

client:

serviceUrl: defaultZone: http://localhost:8761/eureka/

server:

port: 8762

spring:

application:

name: service-hi

#### **服务消费者**

**Ribbon方式**

导入spring-cloud-starter-eureka、spring-cloud-starter-ribbon；

启动类添加@EnableDiscoveryClient，注入bean: restTemplate；并通过

@LoadBalanced注解表明这个restRemplate开启负载均衡的功能

配置application.yml：

eureka:

client:

serviceUrl:

defaultZone: http://localhost:8761/eureka/

server:

port: 8764

spring:

application:

name: service-ribbon

**Feign方式**

导入spring-cloud-starter-eureka、spring-cloud-starter-feign；

启动类添加@EnableFeignClients、@EnableDiscoveryClient，集成了ribbon。

添加接口，接口添加@Feign(value=”服务名”)；方法添加@RequestMapping

配置application.yml：

eureka:

client:

serviceUrl:

defaultZone: http://localhost:8761/eureka/

server:

port: 8765

spring:

application:

name: service-feign

#### **断路器**

服务故障时，打开断路器，调用fallback()。

断路器打开后，负载均衡不再访问故障服务？？？

服务恢复时，不自动断开断路器？？？

浏览器缓存原因；Hystric默认阀值5秒20次

**Ribbon项目使用断路器**

导入spring-cloud-starter-hystrix；启动类添加@EnableHystrix

Service方法加@HystrixCommand(fallbackMethod=”a”)；实现方法a。

**Feign中使用断路器**

修改配置文件，打开自带熔断器，

feign.hystrix.enabled=true

修改注解：@FeignClient(value = "service-hi",fallback = S.class)

创建S类实现接口，注入容器(@Component)；实现方法即为断路器调用方法。

**Hystrix Dashboard (断路器：Hystrix 仪表盘)**

导入spring-cloud-starter-hystrix-dashboard；

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-hystrix-dashboard</artifactId> </dependency>

启动类添加@EnableHystrixDashboard

#### **路由网关zuul**

Zuul的主要功能是路由转发和过滤器。zuul默认和Ribbon结合实现了负载均衡的功能

**路由**

导入spring-cloud-starter-eureka、spring-cloud-starter-feign；

启动类添加@EnableZuulProxy

配置application.yml：

eureka:

client:

serviceUrl:

defaultZone: http://localhost:8761/eureka/

server:

port: 8769

spring:

application:

name: service-zuul

zuul:

routes:

api-a:

path: /api-a/\*\*

serviceId: service-ribbon

api-b:

path: /api-b/\*\*

serviceId: service-feign

**服务过滤**

继承抽象类ZuulFilter，重写方法。

filterType()：返回一个字符串代表过滤器的类型，在zuul中定义了四种不同生命周期的过滤器类型，具体如下：pre--路由之前，routing--路由之时，post--路由之后，error--发送错误调用

filterOrder()：过滤的顺序

shouldFilter()：这里可以写逻辑判断，是否要过滤，本文true,永远过滤。

Run()：过滤器的具体逻辑。可用很复杂，包括查sql，nosql去判断该请求到底有没有权限访问

#### **配置中心。**

在Spring Cloud中，有分布式配置中心组件spring cloud config ，它支持配置服务放在配置服务的内存中（即本地），也支持放在远程Git仓库中。在spring cloud config 组件中，分两个角色，一是config server，二是config client。

**构建Config Server**

导入spring-cloud-starter-eureka，spring-cloud-config-server

启动类添加@EnableConfigServer

配置application.yml：

server:

port: 8888

spring:

application:

name: config-server

cloud:

config:

label: master

server:

git:

password: your password

searchPaths: respo

uri: https://github.com/forezp/SpringcloudConfig/

username: your username

http请求地址和资源文件映射如下:

/{application}/{profile}[/{label}]

/{application}-{profile}.yml

/{label}/{application}-{profile}.yml

/{application}-{profile}.properties

/{label}/{application}-{profile}.properties

文件模板何处配置？？？

**构建config client**

导入spring-cloud-starter-eureka，spring-cloud-starter-config

启动类添加@EnableConfigServer；@Value("${foo}")，获取foo配置值

配置bootstrap.properties：

spring.application.name=config-client

spring.cloud.config.label=master 远程仓库的分支

spring.cloud.config.profile=dev

dev开发环境配置文件，test测试环境，pro正式环境

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

eureka.client.serviceUrl.defaultZone=http://localhost:8889/eureka/

spring.cloud.config.discovery.enabled=true

spring.cloud.config.discovery.serviceId=config-server

server.port=8881

#### **消息总线**

Spring Cloud Bus 将分布式的节点用轻量的消息代理连接起来。它可以用于广播配置文件的更改或者服务之间的通讯，也可以用于监控。本文要讲述的是用Spring Cloud Bus实现通知微服务架构的配置文件的更改。

**改造config client**

导包spring-retry，spring-boot-starter-aop，

spring-cloud-starter-bus-amqp，spring-boot-starter-actuator。

修改配置文件

spring.rabbitmq.host=localhost

spring.rabbitmq.port=5672

# spring.rabbitmq.username=

# spring.rabbitmq.password=

发送post请求：[http://localhost:8881/bus/refresh](http://localhost:8881/bus/refresh" \t "http://blog.csdn.net/forezp/article/details/_blank)，重新读取配置文件

/bus/refresh接口可以指定服务，即使用”destination”参数，比如 “/bus/refresh?destination=customers:\*\*” 即刷新服务名为customers的所有服务，不管ip。

#### **[服务链路追踪(Spring Cloud Sleuth)](http://blog.csdn.net/forezp/article/details/70162074)**

Spring Cloud Sleuth集成了服务追踪组件zipkin。主要功能就是在分布式系统中提供追踪解决方案，并且兼容支持了 zipkin，你只需要在pom文件中引入相应的依赖即可。

**构建server-zipkin**

导包zipkin-server，zipkin-autoconfigure-ui

启动类添加@EnableZipkinServer，开启ZipkinServer的功能

配置文件

server.port=9411

**修改服务**

导包spring-cloud-starter-zipkin

启动类添加@EnableZipkinServer，开启ZipkinServer的功能

配置文件

server.port=8988

spring.zipkin.base-url=http://localhost:9411 spring.application.name=service-hi

服务中调用其他服务；zipkin会自动记录。

#### **[高可用的服务注册中心](http://blog.csdn.net/forezp/article/details/70183572)**

修改服务注册中心，添加多个服务注册中心，实现Eureka Server集群化。

添加两个配置文件，application-peer1.yml，application-peer2.yml

server:

port: 8761

spring:

profiles: peer1

eureka:

instance:

hostname: peer1

client:

serviceUrl:

defaultZone: http://peer2:8769/eureka/

修改hosts文件

127.0.0.1 peer1

127.0.0.1 peer2

更改服务注册url

eureka:

client:

serviceUrl:

defaultZone: http://peer1:8761/eureka/

server:

port: 8762

spring:

application:

name: service-hi

eureka.instance.preferIpAddress=true是通过设置ip让eureka让其他服务注册它。

Eureka-eserver peer1 8761,Eureka-eserver peer2 8769相互感应，当有服务注册时，两个Eureka-eserver是对等的，它们都存有相同的信息。

#### **[docker部署spring cloud项目](http://blog.csdn.net/forezp/article/details/70198649)**

#### **[断路器监控(Hystrix Dashboard)](http://blog.csdn.net/forezp/article/details/70217283)**

改造服务，添加Hystrix Dashboard

导包spring-boot-starter-actuator，spring-cloud-starter-hystrix，

spring-cloud-starter-hystrix-dashboard。

启动类添加@EnableHystrix、@EnableHystrixDashboard；

程序中声明断路点HystrixCommand。

#### **[断路器聚合监控(Hystrix Turbine)](http://blog.csdn.net/forezp/article/details/70233227)**

导包spring-cloud-starter-turbine，spring-cloud-netflix-turbine，

spring-boot-starter-actuator

启动类添加@EnableTurbine，开启turbine，@EnableTurbine注解包含了

@EnableDiscoveryClient注解，即开启了注册服务。

配置文件

spring:

application.name: service-turbine

server:

port: 8769

security.basic.enabled: false

turbine:

aggregator:

clusterConfig: default # 指定聚合哪些集群，多个使用","分割，默认为default。可使用http://.../turbine.stream?cluster={clusterConfig之一}访问

appConfig: service-hi,service-lucy ### 配置Eureka中的serviceId列表，表明监控哪些服务

clusterNameExpression: new String("default")

# 1. clusterNameExpression指定集群名称，默认表达式appName；此时：turbine.aggregator.clusterConfig需要配置想要监控的应用名称

# 2. 当clusterNameExpression: default时，turbine.aggregator.clusterConfig可以不写，因为默认就是default

# 3. 当clusterNameExpression: metadata['cluster']时，假设想要监控的应用配置了eureka.instance.metadata-map.cluster: ABC，则需要配置，同时turbine.aggregator.clusterConfig: ABC

eureka:

client:

serviceUrl:

defaultZone: http://localhost:8761/eureka/

#### **[服务注册(consul)](http://blog.csdn.net/forezp/article/details/70245644)**

spring cloud consul，是一个提供服务发现和配置的工具。具有分布式、高可用、高扩展性。

**Linux平台安装consul**

mkdir -p $GOPATH/src/github.com/hashicorp && cd $!

git clone https://github.com/hashicorp/consul.git

cd consul

make bootstrap

make bootstrap

**Windows平台安装consul**

官网下载：[https://www.consul.io/downloads.html](https://www.consul.io/downloads.html" \t "http://blog.csdn.net/forezp/article/details/_blank)

安装后，在环境变量path下加上：E:\programfiles\consul；

cmd启动：consul agent -dev

**构建工程**

导包spring-cloud-starter-consul-discovery

启动类添加@EnableDiscoveryClient，开启服务发现

配置文件

spring:

cloud:

consul:

host: localhost

port: 8500

discovery:

healthCheckPath: ${management.contextPath}/health

healthCheckInterval: 15s

instance-id: consul-miya

application:

name: consul-miya

server:

port: 8502

#### **编写逻辑，测试。**