1. 服务消费者（ribbon+rest）
2. 定义

Spring cloud有两种服务调用方式，一种是ribbon+restTemplate，另一种是feign。

1. ribbon简介

ribbon是一个负载均衡客户端，可以很好的控制htt和tcp的一些行为。Feign默认集成了ribbon。

* 1. ribbon 已经默认实现了这些配置bean：

1. IClientConfig ribbonClientConfig: DefaultClientConfigImpl
2. IRule ribbonRule: ZoneAvoidanceRule
3. IPing ribbonPing: NoOpPing
4. ServerList ribbonServerList: ConfigurationBasedServerList
5. ServerListFilter ribbonServerListFilter: ZonePreferenceServerListFilter
6. ILoadBalancer ribbonLoadBalancer: ZoneAwareLoadBalancer
7. 新建一个服务消费者工程（consumer）
   1. 导入jar包

<dependencies>

<dependency>

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

<artifactId>spring-cloud-starter-eureka</artifactId>

</dependency>

<dependency>

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

<artifactId>spring-cloud-starter-ribbon</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

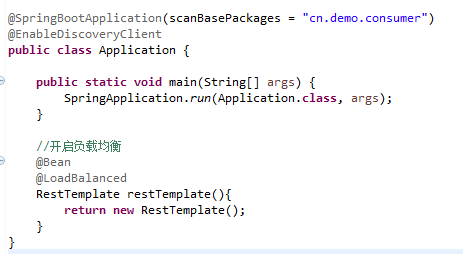
</dependency>

</dependencies>

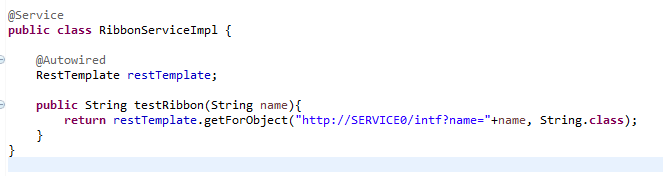
* 1. 配置文件application.yml（同其他）
  2. 在Spring Cloud注册同一服务多个实例（关闭热部署）

修改端口后，多次启动service。

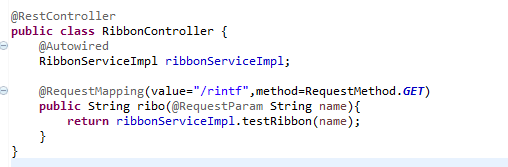
* 1. 注解@EnableDiscoveryClient，指定消费者向服务中心注册



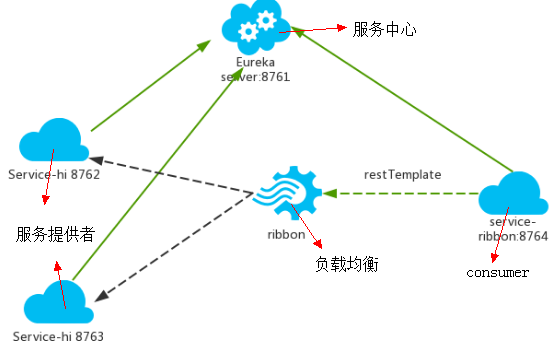
* 1. Service调用服务的方法（重要）



* 1. 封装consumer地址



* 1. 工作流程



1. 一个服务注册中心，eureka server,端口为8080
2. service0工程跑了两个实例，端口分别为8081,8082，分别向服务注册中心注册
3. consumer端口为8764,向服务注册中心注册
4. 当consumer通过restTemplate调用service0的intf接口时，因为用ribbon进行了负载均衡，会轮流的调用service0：8081和8082 两个端口的intf接口；
5. 服务消费者（Feign）
6. 定义

Feign是一个声明式的伪Http客户端，使用Feign只需要创建一个接口并注解。它具有可插拔的注解特性，可使用Feign 注解和JAX-RS注解。Feign支持可插拔的编码器和解码器。Feign默认集成了Ribbon，并和Eureka结合，默认实现了负载均衡的效果。

1. Feign 采用的是基于接口的注解
2. Feign 整合了ribbon
3. 创建一个服务消费者（Feign）
   1. 导入jar包

<dependencies>

<dependency>

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

<artifactId>spring-cloud-starter-feign</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

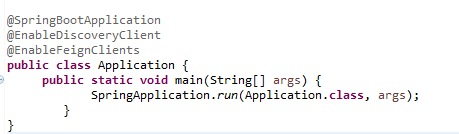
<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

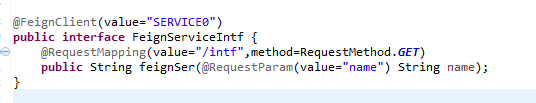
</dependency>

</dependencies>

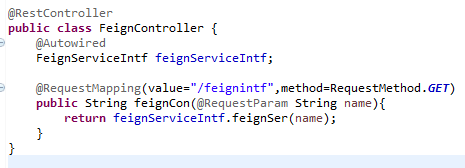
* 1. 配置文件application.yml（同上）
  2. 注解@EnableDiscoveryClient @EnableFeignClients，向服务中心注册



* 1. Servie层消费者调用服务的方法（基于接口的注解）



* 1. Controller层暴露消费者服务地址



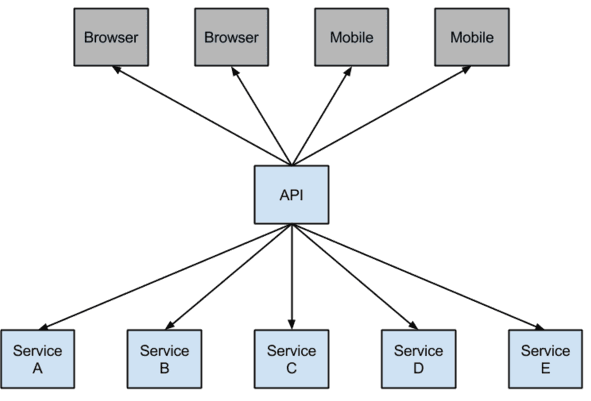
1. 断路器
2. 断路器背景-雪崩效应

在微服务架构中，根据业务来拆分成一个个的服务，服务与服务之间可以相互调用（RPC），在Spring Cloud可以用RestTemplate+Ribbon和Feign来调用。为了保证其高可用，单个服务通常会集群部署。由于网络原因或者自身的原因，服务并不能保证100%可用，如果单个服务出现问题，调用这个服务就会出现线程阻塞，此时若有大量的请求涌入，Servlet容器的线程资源会被消耗完毕，导致服务瘫痪。服务与服务之间的依赖性，故障会传播，会对整个微服务系统造成灾难性的严重后果，这就是服务故障的“雪崩”效应。

1. 断路器简介

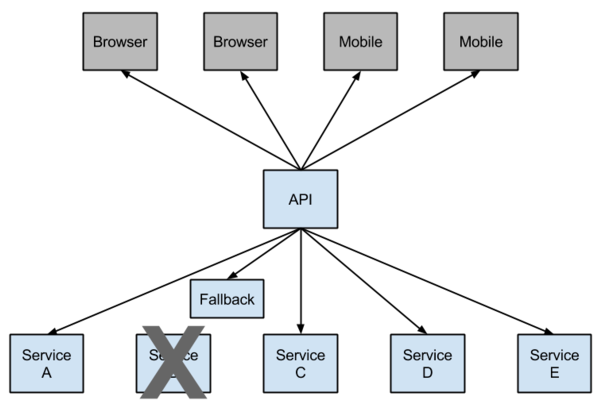
Netflix开源了Hystrix组件，实现了断路器模式，SpringCloud对这一组件进行了整合。

* 1. 微服务架构，一个请求调用多个服务。



* 1. 雪崩效应

较底层的服务如果出现故障，会导致连锁故障。当对特定的服务的调用的不可用达到一个阀值（Hystric 是5秒20次） 断路器将会被打开。



1. 在Ribbon使用断路器
   1. 在pom.xml中引入jar包

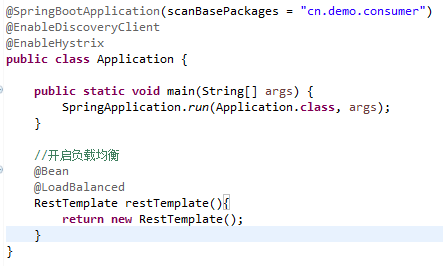
<dependency>

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

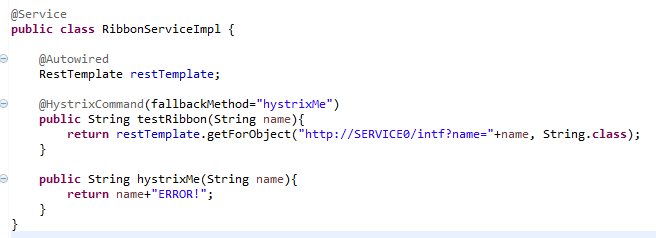
<artifactId>spring-cloud-starter-hystrix</artifactId>

</dependency>

* 1. 在启动类中加入注解@EnableHystrix，开启Hystrix



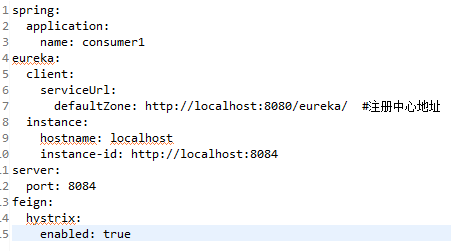
* 1. 在Service类中加上@HystrixCommand注解。该注解对该方法创建了熔断器的功能，并指定了fallbackMethod熔断方法。



1. 在Feign中使用断路器

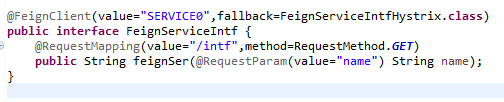
4.1 修改配置文件（格式控制十分严格）

Feign是自带断路器的，在D版本的Spring Cloud中，它没有默认打开。需要在配置文件中配置打开它，在配置文件加以下代码：

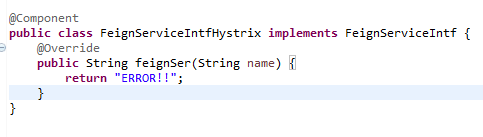


4.2 修改Service接口类

只需要在FeignClient的SchedualServiceHi接口的注解中加上fallback的指定类就行



4.3 新建一个hystrix类，实现feign接口类



1. Hystrix Dashboard （断路器：Hystrix 仪表盘）
   1. 添加jar包

基于service-ribbon 改造，Feign的改造和这一样。

首选在pom.xml引入spring-cloud-starter-hystrix-dashboard的起步依赖

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

<dependency>

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

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

</dependency>

5.2修改启动类，加入注解@EnableHystrixDashboard，开启hystrixDashboard

* 1. <http://localhost:8084/hystrix>访问界面

教程地址：<http://blog.csdn.net/forezp/article/details/70148833>

微服务架构工具总览：https://www.jianshu.com/p/e940b04157ff