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。

为了解决这个问题，业界提出了断路器模型。

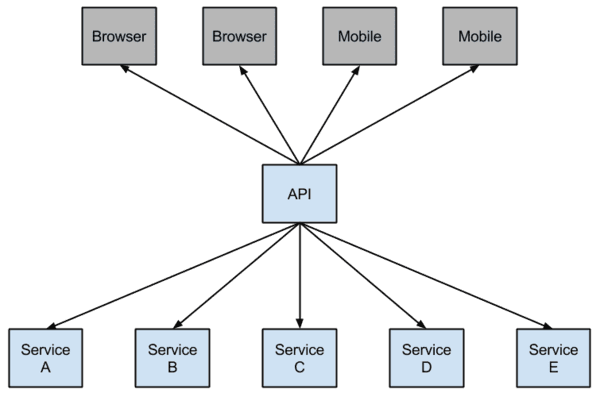
本文档基于[服务消费者(rest+ribbon)](onenote:" \l "服务消费者(rest+ribbon)&section-id={B46804C8-2079-4343-A399-0DFB75448724}&page-id={7CC8375C-9516-46DE-96FA-103BBCB8EEBC}&end&base-path=https://d.docs.live.net/6599c42f24631913/文档/总结/SpringCloud.one)和[服务消费者(feign)](onenote:" \l "服务消费者(feign)&section-id={B46804C8-2079-4343-A399-0DFB75448724}&page-id={820C7550-E7C9-47D5-8E64-528C9C01D210}&end&base-path=https://d.docs.live.net/6599c42f24631913/文档/总结/SpringCloud.one)。

**1.断路器简介**

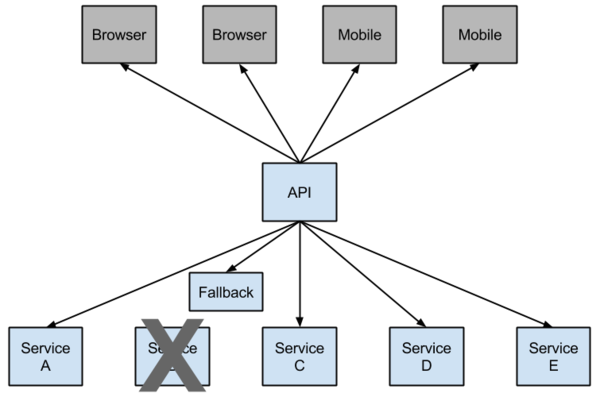
Netflix has created a library called Hystrix that implements the circuit breaker pattern. In a microservice architecture it is common to have multiple layers of service calls.

. —-摘自官网

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。 在微服务架构中，一个请求需要调用多个服务是非常常见的，如下图：



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。



断路打开后，可用避免连锁故障，fallback方法可以直接返回一个固定值。

**2.ribbon中使用断路器**

方式一：创建如下工程

New Module 
Dependencies 
Core 
Web 
Template Engines 
SQL 
NoSQL 
Integration 
Cloud Core 
Cloud Config 
Cloud Discovery 
Cloud Routing 
Spring Boot 2.0.1 
Hystrix Dashboard 
Turbine 
Turbine Stream 
Selected Dependencies 
Cloud Routing 
Ribbon 
Cloud Circuit Breaker 
Hystrix 
Cloud Tracing 
Cloud Messaging 
Cloud AWS 
Cloud Contract 
Pivotal Cloud Foundry 
Azure 
Spring Cloud GCP 
Ops 
Hystrix 
Circuit breaker with spring-cloud-netflix Hystrix 
Circuit Breaker 
previous 
Next 
Cancel 
Help 

方式二：在[服务消费者(rest+ribbon)](onenote:" \l "服务消费者(rest+ribbon)&section-id={B46804C8-2079-4343-A399-0DFB75448724}&page-id={7CC8375C-9516-46DE-96FA-103BBCB8EEBC}&end&base-path=https://d.docs.live.net/6599c42f24631913/文档/总结/SpringCloud.one)创建的工程中加入依赖

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

在程序的启动类中加@EnableHystrix注解开启Hystrix：

ßSpr i ngBootApp1 i cation 
gEnab1 eDi scaveryC1 i ent 
:ßEnab1eHystrix 
public class RibbonApp1ication { 
public static void main (StringC] args) 
{ SpringApp1 ication. run(RibbonApp1 ication. class, 
args) 
ßBean 
ßLoadBa1anced 
RestTemp1ate restTemp1ate() 
[ return new RestTemp1ate() 

改造TestService类，在testService方法上加上@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，熔断方法直接返回了一个字符串。

gService 
public class TestService 
private RestTemplate rest Template; 
"testErrnr") 
gHystrixCommand (fal IbackMethod 
public String testService (String name) { 
return rest Template. getFor0bject( url: "http: ; ; SERVICE-DEMO; String. class) 
public String tesrError (String name) 
, sorry, error 
return "hi, " 
+ name * 

熔断方法中的参数应和方法本身的参数一致。

开启8762，8763两个client

启动工程，访问http://localhost:8764/test?name=齐涛

浏览器显示

hi 齐涛,i am from port:8762

hi 齐涛,i am from port:8763

关闭8763服务，会出现

hi,齐涛,sorry,error!

但是频率远远低于

hi 齐涛,i am from port:8762

**3.feign中使用断路器**

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：

feign.hystrix.enabled=true

基于service-feign工程进行改造，只需要在FeignClient的SchedualServiceHi接口的注解中加上fallback的指定类就行了：

ßFeignC1ient (value — 
ßService 
service-demo , 
fallback 
= FeignServiceTestHystrix. class) 
public interface FeignServiceTest { 
ßRequestMapping (value — 
, method = RequestMethod. GED 
String sayHiFromC1ient0ne (ßRequestParam(va1ue - 
"name") String name) 

FeignServiceTestHystrix需要实现FeignServiceTest 接口，并注入到Ioc容器中，代码如下：

gComponent 
public class FeignServiceTestHystrix implements FeignServiceTest { 
gOverride 
public String sayHiFromClient0ne (String name) 
return "sorry 

启动工程，访问http://localhost:8765/test?name=齐涛

浏览器显示

hi齐涛, i am from 8762

hi齐涛, i am from 8763

关闭8763服务，会出现

sorry 齐涛

hi 齐涛,i am from port:8762

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

基于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.cloud</groupId>

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

</dependency>

或

New Module 
Dependencies 
Core 
web 
Template Engines 
SQL 
NOSQL 
Integration 
Cloud Core 
Cloud Config 
Cloud Discovery 
Cloud Routing 
'loud Circuit Breaker 
Cloud Tracing 
Cloud Messaging 
Cloud AWS 
Cloud Contract 
Pivotal Cloud Foundry 
Azure 
Spring Cloud GCP 
I/O 
Ops 
spring Boot 
Hystrix 
Hystrix Dashboard 
Turbine Stream 
Hystrix Dashboard 
Circuit breaker dashboard with spring-cloud-netflix 
Selected Dependencies 
Cloud Circuit Breaker 
Hystrix Dashboard X 
Hystrix 
Previous 
Next 
Cancel 
Help 

在主程序启动类中加入@EnableHystrixDashboard注解，开启hystrixDashboard：

gSpringBootAppl i cation 
gEnableDi scoveryC1 i ent 
gEnab1eHystrix 
-gEnab1eHystrixDashboard 
public class RibbonApplication { 
public sta •c void main (StringC] args) { 
SpringApp1ication. run(RibbonApp1ication. class, 
CLoadBa1anced 
args) , 
RestTemplate restTemp1ate() ( return new RestTemp1ate() 

J@SpringBootApplication 
@EnableDiscoveryClient 
@Enab1eFeignC1ients 
l@EnableHystrixDashboard 
public class ServiceFeignApplication ( 
public static void main (StringCJ args) ( SpringApplication. run(ServiceFeignApplication. class, 
args) 

打开浏览器：访问<http://localhost:8764/hystrix>,界面如下

C O localhost 
:8764/hystrix 
Hystrix Dashboard 
http://hostname:port/turbine}turbine.stream 
Cluster via Turbine (default cluster): http://turbine-hostname:port/turbine.stream 
Cluster via Turbine (custom cluster): http://turbine-hostname:port/turbine.stream?cluster=[clusterName] 
Sing/e Hystrix App: http://hystrix-app:port/hystrix.stream 
ms Title: 
Example Hystrix App 
Monitor Stream 

不知道什么意思