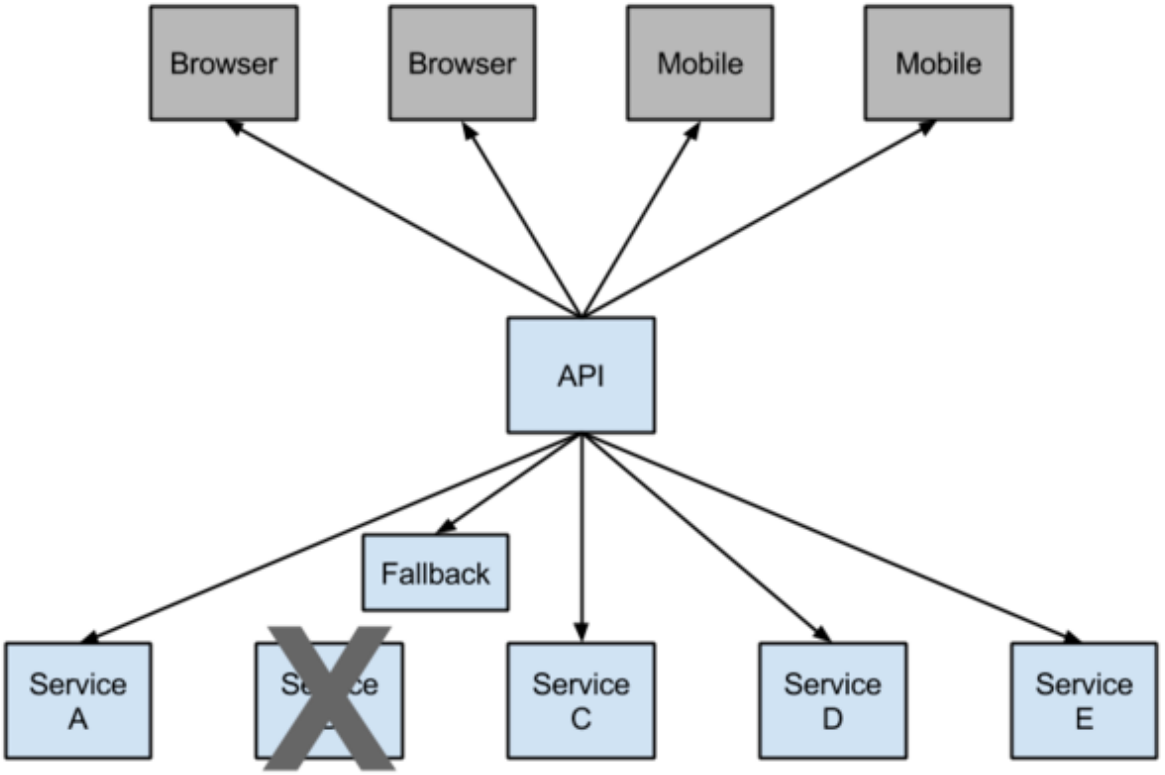
Hystrix简介

在微服务架构中，服务与服务之间可以互相调用（RPC），在SpringCloud可以用RestTemplate+Ribbon和Feign来调用。但是服务并不能保证100%可用，如果单个服务出现问题，其他服务调用这个服务会出现线程阻塞，当线程资源被消耗完时，会导致服务瘫痪，服务与服务之间故障传播，对整个微服务框架造成严重后果，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。

如下图所示，当对特定的服务的调用的不可用达到一个阀值（Hystrix是5秒20次），断路器就会被打开。



断路被打开后，可用避免连锁故障，fallback方法可以直接返回一个固定值。

准备工作

1. 启动eureka-server工程；
2. 启动eureka-client1工程，端口为8762；

一、在ribbon使用断路器

在pom.xml文件中加入首先在pox.xml文件中加入spring-cloud-starter-netflix-hystrix的起步依赖：

```
<dependency>
 <groupId>org.springframework.cloud</groupId>
 <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
</dependency>
```

在程序的启动类ServiceRibbonApplication 加@EnableHystrix注解开启Hystrix：

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.client.loadbalancer.LoadBalanced;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import org.springframework.cloud.netflix.hystrix.EnableHystrix;
import org.springframework.context.annotation.Bean;
import org.springframework.web.client.RestTemplate;

@SpringBootApplication
@EnableEurekaClient
@EnableDiscoveryClient
@EnableHystrix // 断路器
public class ServiceRibbonApplication {

 public static void main(String[] args) {
 SpringApplication.run(ServiceRibbonApplication.class, args);
 }

 @Bean
 @LoadBalanced // 通过@LoadBalanced注解表明这个restRemplate开启负载均衡的功能
 RestTemplate restTemplate() {
 return new RestTemplate();
 }
}
```

改造HelloService类，在hiService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，字符串为"hi,"+name+",sorry,error!"，代码如下：

```
import com.netflix.hystrix.contrib.javanica.annotation.HystrixCommand;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;
```

```
5
6  /**
7   * @author: JunYaoWang
8   * @create: 2018-12-05 11:21
9   **/
10 @Service
11 public class HelloService {
12
13  @Autowired
14  private RestTemplate restTemplate;
15
16  /**
17  * @HystrixCommand 断路器，出现错误时调用hiError方法
18  * @param name
19  * @return
20  */
21  @HystrixCommand(fallbackMethod = "hiError")
22  public String hiService(String name) {
23  // 直接用的程序名替代了具体的url地址，在ribbon中它会根据服务名来选择具体的服务实例，
24  // 根据服务实例在请求的时候会用具体的url替换掉服务名
25  return restTemplate.getForObject("http://eureka-client1/hi?name="+name,String.class);
26  }
27
28  /**
29  * eureka-client1服务调用失败时调用此方法
30  * @param name
31  * @return
32  */
33  public String hiError(String name){
34  return "hi," + name + ",sorry,error!";
35  }
36 }
```

启动：service-ribbon 工程，当我们访问http://localhost:8764/hi?name=gewdata,浏览器显示：

```
1 hi gewdata,i am from port:8762
```

此时关闭 service-hi 工程，当我们再访问http://localhost:8764/hi?name=forezp，浏览器会显示：

```
1 hi ,forezp,orry,error!
```

二、Feign中使用断路器

Feign是自带断路器的，在D版本的Spring Cloud之后，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：

```
# 自带断路器，默认是关闭
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：

```
import com.gewdata.service.impl.SchedualServiceHiHystric;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

/**
 * @author: JunYaoWang
 * @create: 2018-12-05 15:26
 *
 * 定义一个feign接口，通过@ FeignClient（“服务名”），
 * 来指定调用哪个服务。比如在代码中调用了eureka-client1服务的“/hi”接口
 * fallback 断路发生时指定类
 **/
@FeignClient(value = "eureka-client1",fallback = SchedualServiceHiHystric.class)
public interface SchedualServiceHi {

 @RequestMapping(value = "/hi",method = RequestMethod.GET)
 String sayHiFromClientOne(@RequestParam(value = "name") String name);
```

SchedualServiceHiHystric需要实现SchedualServiceHi 接口，并注入到Ioc容器中，代码如下：

```
import com.gewdata.service.SchedualServiceHi;
import org.springframework.stereotype.Component;

/**
 * SchedualServiceHiHystric需要实现SchedualServiceHi 接口注入到OC容器中
 * @author: JunYaoWang
```

```
 * @create: 2018-12-11 09:19
 **/
@Component
public class SchedualServiceHiHystric implements SchedualServiceHi {
 @Override
 public String sayHiFromClientOne(String name) {
 return "sorry,error!" + name;
 }
}
```

启动servcie-feign工程，浏览器打开http://localhost:8765/hi?name=gewdata,注意此时service-hi工程没有启动，网页显示：

```
sorry,error!gewdata
```

打开service-hi工程，再次访问，浏览器显示：

```
hi gewdata,i am from port:8762
```