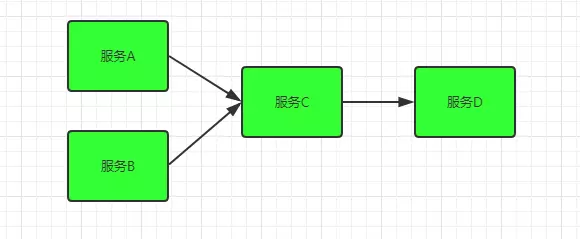
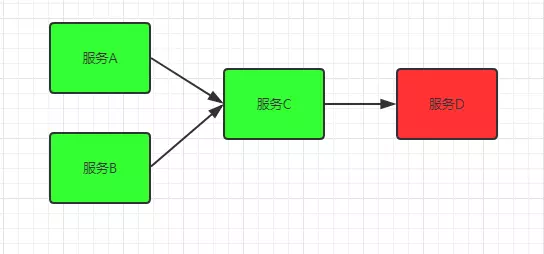
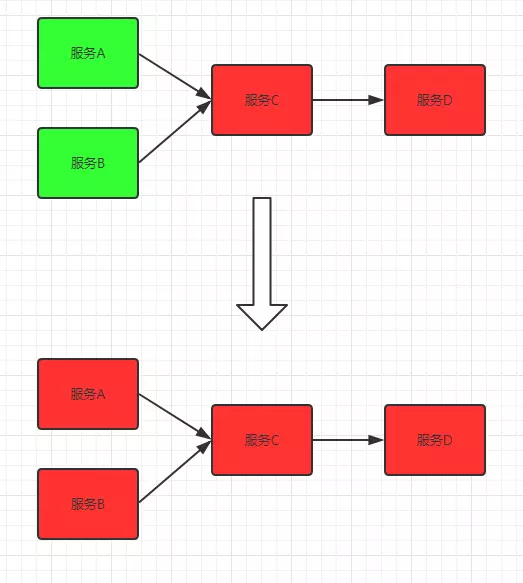
# 级联错误导致微服务集群瘫痪



一组简单的服务依赖关系A，B服务同时依赖于基础服务C，基础服务C又调用了服务D



服务D是一个辅助类型服务，整个业务不依赖于D服务，某天D服务突然响应时间变长，导致核心服务C响应时间变长，其上请求越积越多，C服务也出现响应变慢的情况，由于A，B强依赖于服务C，故而一个无关紧要的服务却影响整个系统的可用。



hystrix熔断器就是一种处理集群中这样问题的组件;

核心思路是:快速错误,牺牲局部保存全局.

# hystrix熔断机制

断路器存在保证系统不会产生大量服务请求压力积压

服务降级提升了系统的有效返回数据的效率

### 断路器

当开启断路器机制时,调用后端微服务的过程就被断路器管理了,根据响应的效果,实现断路器的3中状态的切换

|  |
| --- |
| https://img2018.cnblogs.com/blog/1296956/201905/1296956-20190508222427289-101360919.png |

* + - **打开（全开）**:打开断路器时,对后端微服务提供者不在继续访问
* **关闭**:关闭断路器时,对后端微服务提供者正常访问
* **半开**:半开目的是检测后端这个曾经故障的微服务提供者是否恢复，通过发怂一部分请求访问这个提供者,根据响应判断是否正常，然后决定切换到全开还是关闭.。

如果请求响应都正常,说明已经恢复了,切换到关闭状态

如果请求响应不正常,说明还没恢复,切换到全开状态

### 服务降级

无论断路器是如何工作的,总有一部分数据请求没有获取正确响应.能否实现后端没有正确响应返回时,提供一些可用的响应,至少不是error页面.

服务降级---退而求其次的解决思路.

# JAVA中利用ribbon实现hystrix的使用

### 导入依赖

|  |
| --- |
| ***<*dependency*>  <*groupId*>***org.springframework.cloud***</*groupId*>  <*artifactId*>***spring-cloud-starter-hystrix***</*artifactId*> </*dependency*>*** |

### 启动类添加注解

|  |
| --- |
| @EnableCircuitBreaker |

### 编写代码

|  |
| --- |
| @Service **public class** HelloService **{** @Autowired  **private** RestTemplate **template**;   /\*sayHi方法是发起服务调用的方法,添加服务降级机制  当这个方法调用service-hi 出现问题,断路器就会打开,同时任何一次失败响应  都会进入到服务降级的方法 名称error\*/  @HystrixCommand**(**fallbackMethod = **"error")  public** String sayHi**(**String name**) {** //想办法调用8091 8092 8093负载均衡  //通过template对象,发起请求调用service-hi  String url=**"http://service-hi/client/hello?name="**+name;  String responseBody = **template**.getForObject**(**url, String.**class)**;  //hello name ,I am from 8091/8092/8093  **return** responseBody;  **}** //服务降级的方法error,必须和sayHi结构是一致的  **public** String error**(**String name**){  return "sorry,error happend"**;  **}  }** |

# JAVA中利用feign实现hystrix的使用

feign使用hystrix只需要配置一个属性

|  |
| --- |
| hystrix.metrics.enabled=true |

# 面试题

题目:如果一个微服务集群中登录节点全部失效,如何处理?

思路:

可以采用熔断机制来解决这个问题

什么熔断机制:牺牲局部保存全局

什么是断路器

什么是服务降级

通过服务降级，在多个登录节点全部失效时,可以采用备用登录节点处理这个问题.