# 微服务问题应答

## Spring Cloud微服务如何设计异常处理机制

面向外部提供的服务接口，会通过服务网关（如使用Zuul提供的apiGateway）面向公网提供服务，如给App客户端提供的用户登陆、注册等服务接口。在编写面向外部的服务接口时，服务端所有的异常处理我们都要进行相应地捕获，并在controller层映射成相应地错误码和错误信息，返回给调用方。

而面向内部的服务接口，则是在进行微服务拆分后由于各个微服务系统的边界划定问题所导致的功能逻辑分散。在基于Spring Cloud的微服务体系中，微服务提供方会提供相应的客户端SDK代码，而客户端SDK代码则是通过FeignClient的方式进行服务调用，Feign内部会集成像Ribbon和Hystrix这样的框架来实现客户端服务调用的负载均衡和服务熔断功能。

**最佳实践设计**

首先，无论是内部还是外部的微服务，在服务端我们都应该设计一个全局异常处理类，用来统一封装系统在抛出异常时面向调用方的返回信息。而实现这样一个机制，我们可以利用Spring提供的注解@ControllerAdvice来实现异常的全局拦截和统一处理功能。从最佳实践上考虑，我们一般会为内部和外部接口分别设计一个统一面向调用方的异常对象。

FeignClient为我们提供了异常解码机制

## 原生Spring-Cloud-Config的配置中心的缺点

**Git的权限控制是个坑**：Git的权限管理是说控制用户能不能Push或者Delete分支，或者能不能Push代码，而不是能不能访问某个目录的文件。

**粒度问题**：你每次对一条配置发生crud的操作，其带来的影响是整个文件发生变动。当时我们最理想的存储介质就是数据库。

**Spring-Cloud-Config的刷新机制是个坑**：因为一个配置中心应该要能够做到，配置发生改动的时候，项目能够自动感知，自动更新配置才对。在Spring-Cloud-Config中，这套机制是借助一些代码仓库（SVN、Github等）提供的Webhook机制加上Spring-Cloud-Bus来实现的。

那么问题又来了！配置数据放在数据库中，数据库里没有Webhook这种东西啊，怎么做到实时刷新？

**长轮询，短轮询**

在页面上要实时显示后台的库存数量！比如库存减少了，用户不需要进行刷新，页面上的数字自己会变化。

如果采取短轮询就是在客户端(js)中不断访问后台，后台接到请求马上返回最新的库存数，然后刷新到这个页面当中。

长轮询是客户端(js)依然是不断的去请求。但是呢，服务端不是马上返回。而是等待库存数量变化了再返回。

怎么实现：我们在项目中采用Spring的DeferredResult来实现。

**最后一个问题:如何有效快速的监听出配置表的数据发生了变动？**

因为我们用的是mysql。这里有一个Mysql的自定义函数叫mysql-udf-http。具有http\_get()、http\_post()、http\_put()、http\_delete()四个函数，可以在MySQL数据库中利用HTTP协议进行REST相关操作。

然后再和mysql的触发器结合起来用，可以实现在配置表发生变动的时候，主动通知我们的配置中心服务端。让服务端明白配置发生了变动！