# SpringCloud问题整理

# 问题整理

## 1.SpringCloud微服务与Dubbo的区别

|  |
| --- |
| **dubbo的服务是通过RPC实现的，接口强依赖（RPC远程过程调用）通过TCP等一大堆复杂的过程**  **SpringCloud是使用REST API ,REST 相比RPC更为轻量化，服务方和调用方的依赖只是靠一纸契约，不存在代码级别的强依赖。SpringCloud组合了大量的组件**  **通过上面再几个环节上的分析，相信大家对Dubbo和Spring Cloud有了一个初步的了解。就我个人对这两个框架的使用经验和理解，打个不恰当的比喻：使用Dubbo 构建的微服务架构就像组装电脑，各环节我们的选择自由度很高，但是最终结果很有可能因为一条内存质量不行就点不亮了，总是让人不怎么放心，但是如果你是一 名高手，那这些都不是问题；而Spring Cloud就像品牌机，在Spring Source的整合下，做了大量的兼容性测试，保证了机器拥有更高的稳定性，但是如果要在使用非 原装组件外的东西，就需要对其基础有足够的了解。** |

## 2.SpringCloud与SpringBoot 区别

|  |
| --- |
| **SpringBoot 是Spring快速配置脚手架，可以基于SpringBoot快速开发单个微服务，SpringBoot基于Spring，是Spring的学习和使用便可快速，不仅适合原有工程结构，更适合微服务开发**  **SpringCloud基于SpringBoot 为微服务体系开发中的架构问题，提供了一整套的解决方案，微服务注册与发现，服务消费，服务保护与熔断，网关，分布式调用追踪，分布式配置管理等。SpringCloud依赖于SpringBoot** |

## 3.

|  |
| --- |
|  |

# 配置管理

2.1 Eureka 配置

|  |
| --- |
| **pom.xml**  **<dependencies>**  **<!-- eureka -server服务端 -->**  **<dependency>**  **<groupId>org.springframework.cloud</groupId>**  **<artifactId>spring-cloud-starter-eureka-server</artifactId>**  **</dependency>**  **<dependency>**  **<groupId>org.springframework.boot</groupId>**  **<artifactId>spring-boot-devtools</artifactId>**  **</dependency>**  **<dependency>**  **<groupId>org.springframework</groupId>**  **<artifactId>springloaded</artifactId>**  **</dependency>**  **</dependencies>** |
| **application.yml**  **server:**  **port: 7001**    **eureka:**  **instance:**  **hostname: eureka7001.com #eureka服务端的实例名称 在host中配置**  **client:**  **register-with-eureka: false #false表示不向注册中心注册自己**  **fetch-registry: false #false表示自己是注册中心，我的职责就是维护服务实例，并不需要去检索服务**  **service-url:**  **#设置与eurekaServer交互地址查询服务和注册服务**  **#defaultZone : http://${eureka.instance.hostname}:${server.port}/eureka/**  **defaultZone: http://eureka7002.com:7002/eureka/,http://eureka7003.com:7003/eureka/ #Eureka集群** |