链滴

java 深入之微服务设计原则

作者：wgl530

原文链接：https://ld246.com/article/1577950089772

来源网站：链滴

微服务设计原则

和数据库设计中的N范式一样,微服务也有一定的设计原则,这些原则指导我们更加合理地架构微服务。

单一职责原则

单一职责原则指的是一个单元(类、方法或者服务等)只应关注整个系统功能中单独、有界限的一部分单一职责原则可以帮助我们更优雅地开发、更敏捷地交付。单一职责原则是SOLD原则之一。有兴趣读者可前往htp:e. wikipedia. org/wiki/

SOLD((object-oriented_design进行扩展阅读。

服务自治原则

服务自治是指每个微服务应具备独立的业务能力、依赖与运行环境。在微服务架构中,服务是独立的业单元,应该与其他服务高度解耦。每个微服务从开发、测试、构建、部署,都应当可以独立运行,而不应依赖其他的服务。

轻量级通信机制

微服务之间应该通过轻量级的通信机制进行交互。小编认为,轻量级的通信机制应具备两点:首先是它体量较轻;其次是它应该是跨语言、跨平台的。例如我们所熟悉的REST协议,就是一个典型的“轻量级信机制”;而例如java的RMI则协议就不大符合轻量级通信机制的要求,因为它绑定了ava语言。微服务构中,常用的协议有REST、AMQP、 STOMP、MQTT等

微服务粒度

微服务的粒度是难点,也常常是争论的焦点。应当使用合理的粒度划分微服务,而不是一味地把服务做。代码量的多少不能作为微服务划分的依据,因为不同的微服务本身的业务复杂性不同,代码量也不同。

在微服务的设计阶段,就应确定其边界。微服务之间应相对独立并保持松耦合。小编认为,领域驱动设计(Domain Driven Design,简称DDD)中的“界限上下文(BoundedContext)”可作为划分微服务边界确定微服务粒度的重要依据。限于篇幅,DDD的内容无法展开讲解,对DDD感兴趣的读者朋友们可阅读《Domain Driven Design Quickly》快速入门,也可阅读DDD开山鼻祖 Eric evans的《领域驱动设计:件核心复杂性应对之道》深入理解。同时,在划分微服务的过程中,还应综合考量团队的现状。康威定相信大家已经很熟悉了。微服务架构的演进是一个循序渐进的过程。在演进过程种，常常会根据业务变化，对微服务进行重构，甚至是重新划分，从而让架构更加合理。最终，当微服务的开发、部署、试一级运维的效率很高，并且成本很低时，一个好的微服务架构就形成了