Atitit 微服务的原理与实现方式

目录

[第一章 什么是微服务？ 1](#_Toc31863)

[第二章 核心要素 2](#_Toc7148)

[足够微小， 2](#_Toc22300)

[服务化 进程隔离 2](#_Toc978)

[第三章 实现模式 2](#_Toc8717)

[多个app启动文件（推荐轻量级） 2](#_Toc23731)

[多项目启动（繁琐） 2](#_Toc9581)

[综合 2](#_Toc30506)

[第四章 通讯方式 3](#_Toc31845)

[Ws rest rpc 3](#_Toc17102)

[mq socket 3](#_Toc23803)

[Cmd 3](#_Toc14125)

# ****什么是微服务？****

微服务 Microservices 之父，马丁.福勒，对微服务大概的概述如下：

就目前而言，对于微服务业界并没有一个统一的、标准的定义（While there is no precise definition of this architectural style ) 。

但通常在其而言，微服务架构是一种架构模式或者说是一种架构风格，它提倡将单一应用程序划分成一组小的服务，每个服务运行独立的自己的进程中，服务之间互相协调、互相配合，为用户提供最终价值。

服务之间采用轻量级的通信机制互相沟通（通常是基于 HTTP 的 RESTful API ) 。每个服务都围绕着具体业务进行构建，并且能够被独立地部署到生产环境、类生产环境等。

另外，应尽量避免统一的、集中式的服务管理机制，对具体的一个服务而言，应根据业务上下文，选择合适的语言、工具对其进行构建，可以有一个非常轻量级的集中式管理来协调这些服务。可以使用不同的语言来编写服务，也可以使用不同的数据存储。

****①小服务****

小服务，没有特定的标准或者规范，但他在总体规范上一定是小的。

****②进程独立****

每一组服务都是独立运行的，可能我这个服务运行在 Tomcat 容器，而另一个服务运行在 Jetty 上。可以通过进程方式，不断的横向扩展整个服务。

****③通信****

过去的协议都是很重的，就像 ESB，就像 SOAP，轻通信，这意味着相比过去更智能更轻量的服务相互调用，就所谓 smart endpoints and dumb pipes。

****④部署****

不止业务要独立，部署也要独立。不过这也意味着，传统的开发流程会出现一定程度的改变，开发的适合也要有一定的运维职责。

****⑤管理 多语言 技术栈 综合 php js等各有优缺点**** 能使用不同的语言开发。

传统的企业级 SOA 服务往往很大，不易于管理，耦合性高，团队开发成本比较大。

微服务，可以让团队各思其政的选择技术实现，不同的 Service 可以根据各自的需要选择不同的技术栈来实现其业务逻辑

* 每个微服务都有自己的存储能力，可以有自己的数据库，也可以有统一数据库。

# 核心要素

## 足够微小，

## 服务化 进程隔离

# 实现模式

## 多个app启动文件（推荐轻量级）

每个服务不同的端口号。。

## 多项目启动（繁琐）

## 综合

# 通讯方式

## Ws rest rpc

## mq socket

## Cmd