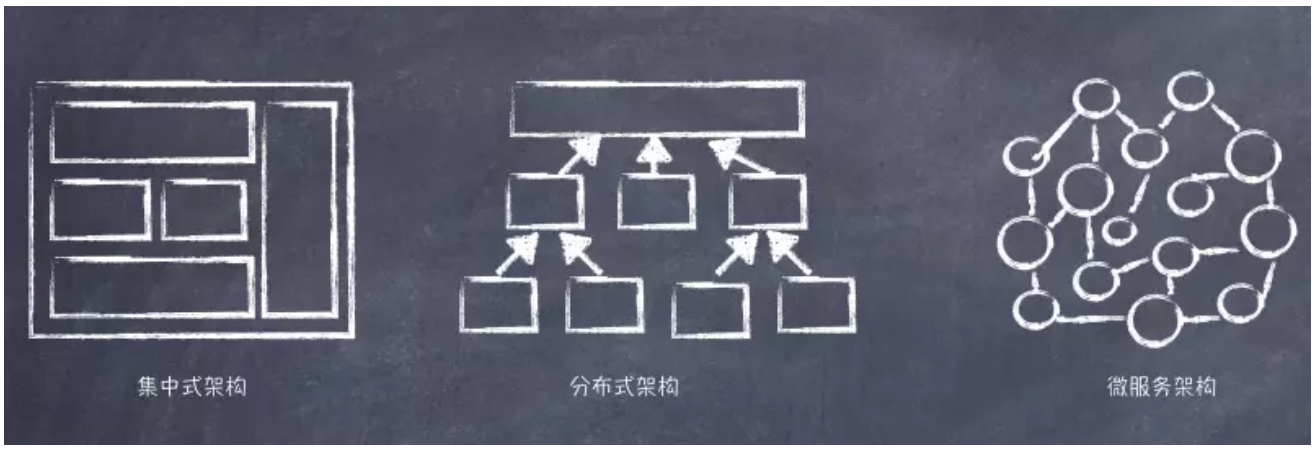
所以微服务，一般来说，是有一套和外部通讯的标准接口的，譬如REST API。

名字带了一个"微"字，说明提供的功能很小，或者很弱。但是一个非常小，或者非常弱的功能，是无法构成一个系统的，因此，他们之间，必须是能够相互组合的。

软件领域，一般把它理解成一种新的架构设计模式。可以和我们通常所熟知的软件架构做类比，譬如集中式架构，分布式架构。这里用一幅抽象的图来表示。




# 微服务架构的特征-Martin Fowler

Martin Fowler 在他的文章中总结了Micro Service的特点

1. 围绕业务功能进行组织 Organized around Business Capabilities

- 不再是以前的纵向切分, 而改为按业务功能横向划分, 一个微服务最好由一个小团队针对一个业务单元来构建

1. 做产品而非做项目 Products not Projects

- 不再是做一个个项目, 交付后就完事了, 而是做产品, 从设计编码到产品运维全过程掌控和负责 you build it, you run it

1. 智能终端加简单通道 Smart endpoints and dumb pipes

- 基于resource的API,大量逻辑放在客户端, 服务器端着重于提供资源 Be of the web, not behind the web

1. 去中心化管理 Decentralized Governance

- 自行其是, 自我管理, 不必在局限在一个系统里, 围绕着一个中心

1. 去中心化数据管理 Decentralized Data Management

- 各人自扫门前雪, 自己管理和维护自己的数据, 各自之间互不直接访问彼此的数据, 只通过 API 来存取数据

1. 基础设施自动化 Infrastructure Automation

- 每个微服务应该关注于自己的业务功能实现, 基础设施应该尽量自动化, 构建自动化,测试自动化, 部署自动化, 监控自动化

1. 为应对失败而设计 Design for failure

    - 设计之初就要考虑高可靠性High Availablity 和灾难恢复 Disaster Recover, 并考虑在错误监测和错误诊断方面如何着手
2. 演进式设计 Evolutionary Design
    - 没有完美的架构, 唯一不变的是变化, 要善于应对变化,容易改变其设计和实现, 因为其小,故而易变

# 微服务架构的特征和风格

## 特征

一个微服务的架构应该具有以下特征

1. 容易被替换和升级 比如以前用 Ruby 快速开发的原型可以由 Java 实现的微服务代替, 反正服务接口没变, 所以也没有什么影响
2. 按功能单元组织服务 职责最好相对独立和完整, 以避免对其他服务过多的依赖和交互
3. 微服务可选择最适合自己的技术方案 服务性质的不同影响技术的选型, 比如帐户的注册登录, 完成可以由 ruby on rails, python django 这些脚本框架来实现. 但是, 对于音视频流的编解码和处理, 最好用 C/C++ 甚至汇编语言来写. 其他的诸如数据库的选型, ORM 或 MVC 框架的选择, 都可以随机应变, 按照业务和技术的具体需求, 根据团队的技术栈和人员现状选择最适合的方案
4. 架构由层次化转向扁平化 服务内部可以进行适当的分层, 服务之间尽量扁平化, 不要引入过多的层次

## 风格

1. 短小精悍, 独立自治 只做一个业务并专注做好它
2. 自动化部署和测试 相比大而全的单个服务, 微服务会有更多的进程, 更多的服务接口, 更多不同的配置, 如果不能将部署和测试自动化, 微服务所带来的好处会大大逊色
3. 尽量减少运维的负担 微服务的增多可能会导致运维成 本的增加, 监控和故障诊断也可能会更困难, 所以要未雨绸缪, 在一开始的设计阶段, 就要充分考虑到如何及时的发现问题和解决问题
4. 拥抱失效与故障 微服务的高可靠性设计和防错性设计是与生俱来的, 分布在不同的机器,地域上的服务的硬件, 网络等随时有可能出问题, 而这些问题要对服务质量没有任何影响
5. 每个服务都是灵活易变, 可伸缩,可扩展, 可组合的 微服务为应对变化提供了更多的可能, 就象乐高积木, 可以随意增减组合, 堆积出来不同的产品