SC笔记

# 网站架构演变过程

## 传统架构(单点应用)

ssh或ssm项目，会将整个业务模块放在一个项目中，分为mvc架构、逻辑层、数据库访问层、控制层。

优点：适合单人或小团队开发。

缺点：耦合度太高，一个模块的失败，可能导致全项目出错，例如一个模块将数据库连接占完，则其它模块就不能访问了。

## 分布式架构

基于传统架构演变过来的。将多个模块分开为多个项目，比如登陆、积分模块分成两个项目，每个项目有自己的数据库。每个项目之间没有关系。

项目：包含业务逻辑层和视图层，即包括前台项目和后台项目。

服务：只包含义务逻辑层，没有视图层。

优点：耦合度降低、粒度更细。

## SOA架构

面向服务架构，俗称服务化，将共同的业务代码抽取出来，提供给其它接口调用，服务之间采用rpc远程调用。

服务：将共同的业务逻辑进行拆分，拆分成独立一个项目进行部署。比如前台服务调用后台服务。

特点：底层基于SOAP(简单对象协议http+xml混合物)或者ESB(消息总线)实现，底层使用HTTP或https协议 + 重量级XML数据交换格式进行通讯。在后面微服务中，以json格式替代xml。

缺点：

* 依赖中心化服务发现机制。
* SOAP的xml占用宽带，xml报文冗余大，高并发时不太好。
* 服务管理混乱，缺少服务管理和治理设施不完善。

rpc远程调用技术框架(核心底层socket技术或netty实现)：

* httpclient
* springcloud
* dubbo
* grpc

## 微服务架构

基于SOA架构演变过来，粒度比SOA更加精细，目的是为了提高效率。每个服务之间互不影响，且必须独立部署(独立数据库、独立redis等)，更加轻量级，采用restful风格提供API，也就是Http协议和Json格式，更加轻巧，适合敏捷开发。

SOA架构与微服务架构的区别：

* 微服务基于SOA，但是去除了SOA中的ESB消息总线，采用http+json(restful)进行传输。
* 微服务架构粒度更加精细，服务之间互不影响，每个微服务独立部署，更加轻巧。
* SOA架构中数据库存储会发生共享，微服务强调每个服务都是单独数据库。
* 微服务更适合敏捷开发，版本迭代快。

# SpringCloud简介

## 为什么使用SC

* 是目前一套比较完整的微服务解决方案框架。
* 解决了一套问题，而不是某个问题。

## 定义

sc是基于sb基础上开发的微服务框架，包括服务治理、注册中心、配置管理、断路器、智能路由、微代理、控制总线、全局锁、分布式会话。

# Eureka

## 相关概念

### 注册中心

存放服务地址相关信息(接口地址)

### 服务注册

将服务信息注册到注册中心上。

### 服务发现

从注册中心上获取服务信息。

### 服务提供者

提供服务接口方。

### 服务消费者

调用别人接口进行使用。一个服务可以既是消费者又是提供者。

## SC支持的注册中心

Eureka、Consule、Zookeeper

## 服务注册原理

* 启动注册中心。
* 启动A服务。
* A服务将服务地址和端口以别名方式注册到注册中心(serviced: Aservice value: 127.0.0.1:8080)，一般会有多个一模一样的A服务，用于分担压力。所以在注册中心会显示一个serviceId对应多个ip。
* B消费者调用接口时，使用服务别名，即serviceId去注册中心获取rpc远程调用地址，对于SC微服务来说，B会将serivceId对应的所有ip和port缓存到jvm(每30s刷新一次)，使用本地负载均衡决定调用哪个微服务。
* 决定后使用HttpClient实现调用。