**前提知识+相关的说明**

1． 目前我们学习到最后的微服务架构SpringCloud，到我这里基本人需要大家熟悉以前的学习内容和知识，也即我默认大家已经熟悉了

SpringMVC+Spring/SpringBoot+Mybatis+Maven+git……

不再重复讲解，

2． 本次Cloud的讲解的方式，由于我们只有2.5天，大概21种技术之多，只能挑选最重要最常用的技能给大家分享，俗称Cloud技术的五大神兽

public classDept{

          private Integer id;

         privateString deptName;

          …..

}

POM/XXXMapper.xml/application.yml

重点讲解和Cloud相关的新知识

3．本次课程只是Cloud的第一季

天上飞的理念，必然有落地的实现

**1．从面试题开始**

**1.1什么是微服务**

**1.2微服务之间是如何独立通讯的**

**1.3 SpringCloud和Dubbo有哪些区别**

   Dubbo通信机制基于RPC远程过程调用、SpingCloud基于RESTful API调用

**1.4 SpringBoot和SpringCloud，请你谈谈对他们的理解**

**1.5 什么是服务熔断？什么是服务降级**

**1.6 微服务的优缺分别是什么？说下你在项目开发中碰到的坑**

**1.7你所知道的微服务技术栈有哪些？请列举一二**

**1.8 eurek和zookeeper都可以提供服务注册与发现的功能，请说说两个的区别？**

**1.9……..**

**2.微服务概述**

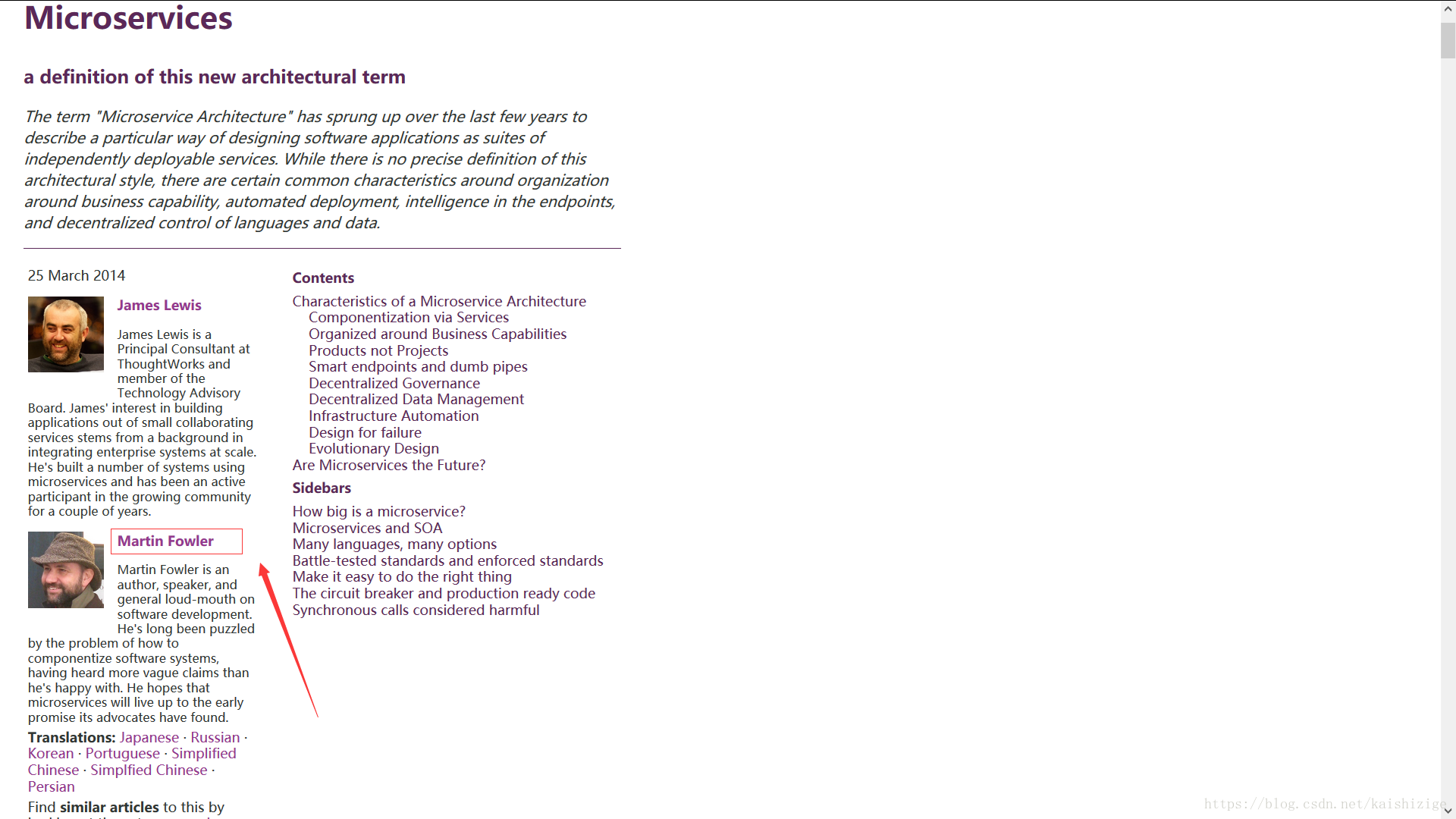
**2.1 微服务是什么**

**出处**

         业内大牛马丁.福勒（MartinFowler）这样描述微服务：

         论文网址：

<https://martinfowler.com/articles/microservices.html>’



l 就目前而言，对于微服务业界并没有一个统一的、标信的定义（Whilethere is no precise definition of this architectural syle）

In short, the microservice architecturalstyle [[1]](https://martinfowler.com/articles/microservices.html#footnote-etymology)is an approach to developing a single application as a suite of small services,each running in its own process and communicating with lightweight mechanisms,often an HTTP resource API. These services are built around businesscapabilities and independently deployable by fully automated deploymentmachinery. There is a bare minimum of centralized management of these services,which may be written in different programming languages and use different datastorage technologies.

总之，建筑风格microservice[[1]](https://martinfowler.com/articles/microservices.html" \l "footnote-etymology" \t "_blank)预算编制办法一种应用于一套小服务，每个运行在其自身和与轻量级方法机制，通常的HTTP资源API。这些服务是建立围绕业务能力和独立的完全展开机械自动化部署。有限度的这些服务的集中化管理，可以写成不同的编程语言和使用不同数据存储技术。

**总结**

l  但通常而言，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一组小的服务，每个服务运行在其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所存储。

**单机系统All       In     One**

可以看作eclipse工作空间里面只有一个工程

如：Tmall

                  com.atguigu.service

                          商品/交易/积分/订单…….

**分布式**

         专业的事交给专业的人做，尽量降低耦合度

         各个模块/服务，各自独立出来，分灶吃饭

         各自微小的一个过程，让专业的人专业的模块，来做专业的事情

         独立部署

l  拆分

l  各自独立的进程

l  拥有自己独立的数据库

         Dubbo通信机制基于RPC远程过程调用、SpingCloud基于RESTful API调用

**技术维度理解**

         微服务化的核心就是将传统的一站式应用，根据业务拆分成一个一个的服务，彻底地去耦合，每一个微服务提供单个业务功能的服务，一个服务做一件事，

       从技术角度就是一种小而独立的处理过程，类似进程的概念，能够自行单独启动或销毁，拥有自己独立的数据库

**2.2 微服务与微服务架构**

**微服务**

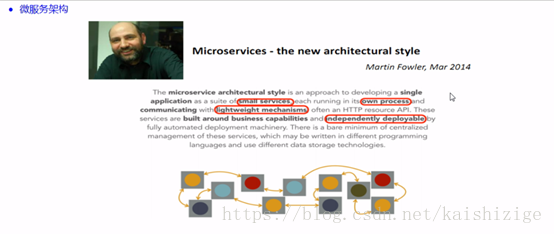
强调的是的大小，它关注的是某一个点，是具体解决某一个问题/提供落地对应服务的一个服务应用，狭义的看，可以看作Eclipse里面一个个微服务工程/或者Module

Eclipse工具里面用maven开发的一个个独立的小moudle，它具休是使用SpringBoot开发的一个小模块，专业的事情交给专业的模块来做，一个模块就做这一件事情

强调的是一个个的个体，每个个体完成一个具体的任务或者功能

医院一个个的科室

**微服务架构**



         微服务架构是一种架构模式，它提倡将单一应用程序划分成一组小的服务，服务之间相协调、互相配合，为用户提供最终价值。每个服务运行在其独立的进程中，服务与服务间采用轻量级的通信机制互相协作（通常是基于HTTP的RESTful API）。每个都围绕着具体业务进行构建，并且能够独立的部署到生产、类生产环境等。另外，应当尽量避免统一的、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。

**2.3 微服务优缺点**

**优点**

l  每个服务足够内聚，足够小，代码容易理解这样能聚集一个指定的业务功能或业务需求

l  开发简单、开发效率提高，一个服务可能就是专一的只干一件事

l  微服务能够被小团队单独开发，这个小团队是2到5人的开发人员组成

l  微服务是松耦合的，是有功能意义的服务，无论是在开发阶段或部署阶段都是独立的

l  微服务能使用不同的语言开发

l  易于和第三方集成，微服务允许容易且灵活的方式集成自动部署，通过持续集成工具，如Jenkins，Hudson，bamboo

l  微服务易于被一个开发人员理解、修改和维护，这样小团队能够更关注自己的工作成果。无需通过合作体现价值。

l  微服务允许你利用整合最新技术

l  微服务只是业务逻辑的代码，不会和HTML,CSS或其他界面组件混合

l  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

**缺点**

开发人员要处理分布式系统的复杂性

多服务运维难度，随着服务的增加，运维的压力也在增大

系统部署依赖

服务间通信成本

数据一致性

系统集成测试

性能监控…………

开发中，我们们有两种开发模式

1． 前后端分离

a)      我们Java程序员，相对而言比较幸福，why?

b)      我们只需要管理后端，给前端的H5工程师就按照约定

c)       Rest地址+输入参数格式和报文约定+输出参数

$.post{rest,jsonParameter,callBack}

可以灵活搭配，连接公共库+连接独立库？？？

2.全栈工程师

         H5+javaEE+………..

**2.4 微服务技术栈有哪些**

         微服务技术栈：多种技术的集合体

         我们再讨论一个分布式的微服务架构的话，它需要有哪些维度？？

         一个分布式的微服务架构       维度       E时代下的数字化生活

                  服务治理                                                                      手机

                  服务注册                                 电脑

                  服务调用                                充电宝

                  服务负载均衡                            路由器

                  服务监控                                                                      ……           小米科技

                          各个杂牌

                  SpringCloud

|  |  |  |
| --- | --- | --- |
| **微服务条目** | **落地技术** | **备注** |
| 服务开发 | SpringBoot、Spring、SpringMVC |  |
| 服务配置与管理 | Netflix公司的Archaius、阿里的Diamond等 |  |
| 服务注册与发现 | Eureka、Consul、Zookeeper等 |  |
| 服务调用 | Rest、RPC、GRP C |  |
| 服务熔断器 | Hystrix、Envoy等 |  |
| 负载均衡 | Ribbon、Nginx等 |  |
| 服务接口调用  （客户端调用服务的简化工具） | Feign等 |  |
| 消息队列 | Kafka、RabbitMQ、ActiveMQ等 |  |
| 服务配置中心管理 | SpringCloudConfig、Cherf等 |  |
| 服务路由（API网关） | Zuul等 |  |
| 服务监控 | Zabbix、Nagios、Metrics、Spectator等 |  |
| 全链路追踪 | Zipkin、Brave、Kubernetes等 |  |
| 服务部署 | Docker、OpenStack、Kubernetes等 |  |
| 数据流操作开发包 | SpringCloud Stream(封装与Redis、Rabbit、Kafka等发送接收消息) |  |
| 事件消息总线 | Spring Cloud Bus |  |
| …… |  |  |

**2.5 为什么选择SpringCloud作为微服务架构**

**选型依据**

整体解决方案和框架成熟度

社区热度

可维护性

学习曲线

**当前各大IT公司用的微服务架构有哪些？**

阿里Dobbo/HSF

京东JSF

新浪微博Motan

当当网DubboX

……

**各微服务框架对比**

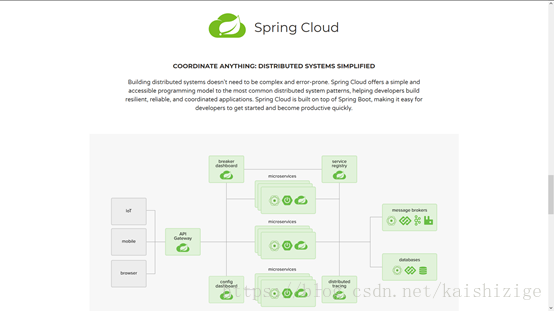
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **功能点/服务**  **框架** | **备选方案** | | | | |
| **Netflix/Spring Cloud** | **Motan** | **gRPc** | **Thrift** | **Dubbo/DubboX** |
| 功能定位 | 完整的微服务框架 | RPC框架，但整合了ZK  或Consul，实现集群环境的基本的服务注册/发现 | RPC框架 | RPC框架 | 服务框架 |
| 支持Rest | 是   Ribbon支持多种可插拔的序列化选择 | 否 | 否 | 否 | 否 |
| 支持RPC | 否 | 是（Hession2） | 是 | 是 | 是 |
| 支持多语言 | 是（Rest形式）？ | 否 | 是 | 是 | 是 |
| 服务注册/发现 | 是（Eureka）  Eureka服务注册表，Karyon服务端框架支持服务自注册和健康检查 | 是（Zookeeper/consul） | 否 | 否 | 是 |
| 负载均衡 | 是（服务端zuul+客户端Ribbon） Zuul-服务，动态路由   云端负载均衡Eureka（  针对中 间层服务器） | 是（客户端） | 否 | 否 | 是（客户端） |
| 配置服务 | Netflix Archaius   Spring Cloud Config Server集中配置 | 是（zookeeper提供） | 否 | 否 | 否 |
| 服务调用链监控 | 是（zuul）  Zuul提供边缘服务，API网关 | 否 | 否 | 否 | 否 |
| 高可用/容错 | 是（服务端Hystrix+客户端Ribbon） | 是（客户端） | 否 | 否 | 是（客户端） |
| 典型应用案例 | Netflix | Sina | Gooogle | Fackbook |  |
| 社区活跃程度 | 高 | 一般 | 高 | 一般 | 2017.8重新启动 |
| 学习难度 | 中等 | 低 | 高 | 高 | 低 |
| 文档丰富度 | 高 | 一般 | 一般 | 一般 | 高 |
| 其他 | Spring Cloud Bus为我们的应用程序带来了更多管理端点 | 支持降级 | Netflix内部在开发集成gRPC | IDL定义 | 实践的公司比较多 |

**3.SpringCloud入概述**

**3.1 是什么**

**官网说明**

<https://spring.io/>



SpringCloud，基于SpringBoot提供了一套微服务解决方案，包括服务注册与发现，配置中心，全链路监控，服务网关，负载均衡，熔断器等组件，除了基于NetFlix的开源组件做高度抽象封装之外，还有一些选型中立的开源组条件

SpringCloud利用SpringBoot的开发便利性巧妙地简化了分布式系统基础设施的开发，SpringCloud为开发人员提供了快速建立分布式系统的一些工具，包括配置管理、服务发现、断路器、路由、微代理、事件总线、全局锁、决策竞选、分布式会话等等，它们都可以用SpringBoot的开发风格做到一键启动和部署。

SpringBoot并没有重复制造轮子，它只是将目前各家公司开的比较成熟、经得起实际考验的服务框架组合起来，通过SpringBoot风格进行再封装屏蔽掉了复杂的配置和实现原理，最终给开发者留出了一套简单易懂、易部署和易维护的分布式系统开发工具包

**SpringCloud=分布式微服务架构下的一站式解决方案，是各个微服务架构落地技术的集合体，俗称微服务全家桶**

**SpringCloud和SpringBoot是什么关系**

         SpringBoot专注于快速方便的开发单个个体微服务

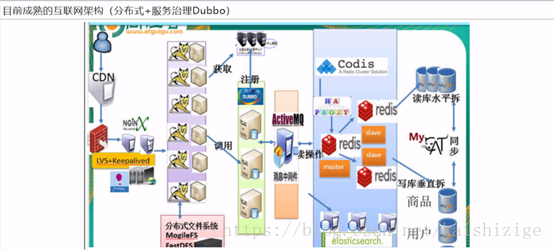
         SpringCloud是关注全局的微服务协调整理治理框架，它将SpringBoot开发的一个个单体微服务整合并管理起来，为各个微服务之间提供，配置管理、服务发现、断路器、路由、微代理、事件总线、全局锁、决策竞选、分布式会话等等集成服务

         SpringBoot可以离开SpringCloud独立使用开发项目，但是SpringCloud离不开SpringBoot，属于依赖的关系

         SpringBoot专注于快速、方便的开发单个微服务个体，SpringCloud关注于全局的服务治理框架。

**Dubbo是怎么到SpringCloud的？哪些优缺点让去技术选项**

n  **目前成熟的互联网架构（分布式+服务治理Dubbo）**



n  **我们把SpringCloud VS DUBBO进行一番对比**

u  **活跃度**

[**https://github.com/dubbo**](https://github.com/dubbo)

[**https://github.com/spring-cloud**](https://github.com/spring-cloud)

u  **对比结果**

|  |  |  |
| --- | --- | --- |
|  | **Dubbo** | **Spring Cloud** |
| 服务注册中心 | Zookeeper | Spring Cloud Netflix Eureka |
| 服务调用中心 | RPC | REST API |
| 服务监控 | Dubbo-monitor | Spring Boot Admin |
| 断路器 | 不完善 | Spring Cloud Netflix Hvstrix |
| 服务网关 | 无 | Spring Cloud Netflix |
| 分布式配置 | 无 | Spring Cloud Config |
| 服务跟踪 | 无 | Spring Cloud Sleuth |
| 消息总线 | 元 | Spring Cloud Bus |
| 数据流 | 无 | Spring Cloud Stream |
| 批量任务 | 无 | Spring Cloud Task |
| …… | …… | …… |

最大区别：SpringCloud抛弃了Dubbo的RPC通信，采用的是基于HTTP的REST方式

严格来说，这两种方式各有优劣，虽然从一定程度上来说，后者牺牲服务调用的性能，但也避免了上面提到的原生RPC带来的问题。而且REST相比RPC更为灵活，服务提供方和调用方的依赖只依靠一纸契约，不存在代码级别的强依赖，这在强调快速深化的微服务环境下，显得更加合适

品牌机与组装机的区别

很明显，Spring Cloud的功能比DUBBO更加强大，涵盖面更广，而且作为Spring的拳头项目，它也能够与Spring Framework、Spring Boot、Spring Data、Spring Batch等其他Spring项目完美整合，这些对于微服务而言是至关重要的。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。

社区支持与更新力度

最为重要的是，DUBBO停止了5年左右的更新，虽然2017.7重启了。对于技术发展的新需求，需要由开发者自行拓展升级（比如当当网弄出了DubboX）,这对于很多想要采用微服务架构的中小软件组织，显然是不太合适的，中小公司没有这么强大的技术能力去修改Dubbo源码+周边的一整套解决方案，并不是每一个公司都有阿里的大牛+真实的线上生产环境测试过。

n  **总结Cloud与Dubbo**

问题：

曾风靡国内的开源RPC服务框架Dubbo在重启维护后，令许多用户为这雀跃，但同时，也迎来一些质疑的声音。互联网技术发展迅速，Dubbo是否还能跟上时代？Dubbo与Spring Cloud相比又有何优势和差异？是否会有相关举措保证Dubbo的后续更新频率？

人物：Dubbo重启维护开发的刘军，主要负责人之一

刘军，阿里巴巴中间件高级研发工程师，主导了Dubbo重启维护以后的几个发版计划，专注于高性能RPC框架和微服务相关领域。曾负责网易考拉RPC框架的研发及指导在内部使用，参与了服务治理平台、分布式跟踪系统、分布式一致性框架等从无到有的设计与开发过程。



关于Dubbo和Spring Cloud间的关系，我们在开源中国年终盛典的Dubbo分享中也作了简单阐述，首先要明确的一点是Dubbo和Spring Cloud并不是完全的竞争关系，两者所解决的问题域并不一样：Dubbo定位始终是一款RPC框架，而Spring Cloud的目标是微服务架构下的一站式解决方案。如果非要比较的话，我觉得Dubbo可以类比Netflix OSS技术栈，而SpringCloud集成了Netflix OSS作为分布式服务治理解决方案，但除此之外Spring cloud还提供了包括config、stream、security、sleuth等等分布式问题解决方案。

当前由于RPC协议、注册中心元数据不匹配等问题，在面临微服务基础框架造型时Dubbo与Spring Cloud是只能二选一，这也是为什么大家总是拿Dubbo和Spring Cloud做对比的原因之一。Dubbo之后会积极寻求适配到SpringCloud生态，比如作为Spring Cloud的二进制通信方案来发挥Dubbo的性能优势，或者Dubbo通过模块化以及对http的支持适配到Spring Cloud。

**3.2 能干嘛**

Distributed/versioned configuration(分布式/版本控制配置)

Service registration and disconvery(服务注册与发现)

Routing(路由)

Service-to-service-calls(服务到服务的调用)

Load balancing(负载均衡配置)

Distributed messaging(分布式消息管理)

…….

**3.3 去哪下**

**官网**

http://projects.spring.io/spirng-cloud/

**参考书**

<https://springcloud.cc/spring-cloud-netflix.html>

本次开发API说明

<http://cloud.spring.io/spring-cloud-static/Dalston.SR1/>

<https://spirngcloud.cc/spring-cloud-dalston.html>

SpringCloud中国社区

[http://sprincloud.cn](http://sprincloud.cn/)

SpringCloud中文网

<https://springcloud.cc/>

**3.4 怎么玩**

服务的注册与发现（Eureka）

服务消费者（rest+Ribbon）

服务消费者（Feign）

断路器（Hystrix）

断路器监控（Hystrix Dashboard）

路由网关（Zuul）

分布式配置中心（Spring Cloud Config）

消息总线（Spring Cloud Bus）

服务链路追踪（Spring Cloud Sleuth）

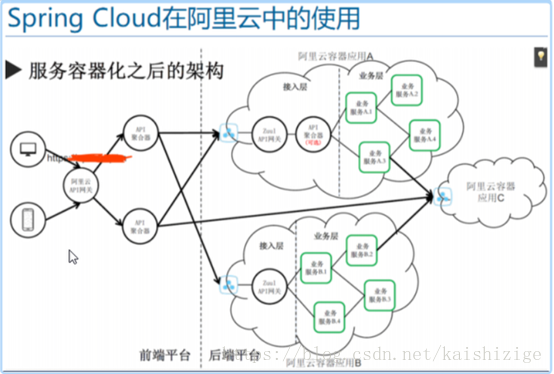
……..

All

**3.5  SpringCloud国内使用情况**

国内公司

阿里云



面试的时候是不是需要有一种

理论知识+面试时候的谈资！！！！！！

**4.Rest微服务构建案例工程模块**

**4.1 总体介绍**

l  承接着我们的spirngmvc+mybatis+mysql初级高级课程，以Dept部门模块做一个微服务通用案例Consumer消费者（Client）通过REST调用Provider提供者（Server）提供的服务

n  已经学习过的知识springmvc

n  已经学习过的知识mybaits

l  Maven的分包分模块架构复习

一个简单的Maven模块结构是这样的：

----- app-parent 一个父项目（app-parent）聚合很多子项目（app-util,app-dao,app-service,app-web）

         |----- pom.xml(pom)

         |

         |---------- app-util

||---------- pom.xml(jar)

|

         |---------- app-dao

||---------- pom.xml(jar)

|

         |---------- app-service

||---------- pom.xml(jar)

|

         |---------- app-web

||---------- pom.xml(war)

l  直接动手干

n  Maven的分包分模块架构复习

一个Project带着多个Module子模块

MicroServiceCloud父工程（Project）下次次带着3个子模块（Module）

microservicecloud-api

         封装的整体entity/接口/公共配置等

microservicecloud-provider-dept-8001

         微服务落地的服务提供

microservicecloud-consumer-dept-80

         微服务调用的客户端使用

         80端口

80端口是为HTTP（HyperText Transport Protocol）即超传输协议开放的

此为上网冲浪使用次数最多的协议，主要用于WWW(World Wide Web)即万维网传输信息的协议

可以通过HTTP地址（即常说的“网址”）加“：80”来访问网站，

因为浏览网页服务默认的端口号都是80，因此只需输入网址即可，不用输入“：80”了。