知识回顾：

1. 微服务必由之路

单体架构 > 业务垂直拆分（水平拆分）> 微服务

单点故障，分拆开，部署到不同服务上，同一套系统部署多份。微服务和业务垂直拆分思路一致，微服务拆分粒度比业务垂直拆分更加细致。

微服务不能万能，粒度过细。开发n多项目，项目复杂度也增加，维护难度增加。微服务结构远程调用，本来单体内部调用，远程调用过程中，内部是java对象互相调用。远程调用，不能直接调用，必须对java对象序列化。先要序列化，拿到后反序列化，两次序列化转换，都需要耗时。增加了程序调用时间。

总结：实际开发中，微服务大头+业务垂直拆分小头。

1. SpringBoot

面试角度

套路：

1. why?

开发工具越来越多，开发结构内容越来越，开发效率变低。业界提倡敏捷开发。

SpringBoot怎么实现敏捷开发，开发方式快速。对业务无关开发工作量要减少。

Maven工程缺点：jar管理不同意，jar版本升级，jar版本冲突。三大框架整合：配置文件太多，配置项太多记不住，配置项xml文件，没有语法检查，增加bug几率。启动方式过于繁琐。

SpringBoot通过注解+约定大于配置，大量节约开发代码量。

通过SpringBoot自己出品父工程spring-boot-starter-parent，统一管理公用大多的jar的版本。基于parent和starter-web整合ssm框架，通过注解和约定实现三大框架配置文件都不用配置，web.xml不用。不是万能，还需要个性化配置时，例如：datasource，额外配置文件application.yml文件。无需配置tomcat，连插件也不需要，只需要写一个启动类，通过main执行。增加2行代码，一行SpringAppliaction.run()，一个注解搞定。

1. 定义、概念、用途
2. 同类产品比较（博学、选型）

基于Maven实现SpringBoot敏捷开发

Eclipse支持不够好，spring eclipse sts/idea

1. 在你做过项目中如何应用（跟咱们结合着说）（项目经验）

创建SpringBoot的项目，继承springboot出品父项目，依赖starter-web jar。基于ssm三大框架，@RestController，都是spring一家子，spring开发时“完全”兼容springmvc注解。@RequestMapping，@GetMapping，@PostMapping。使用新的main启动方式。

1. 挖陷阱（掌握面试主动权，由于精心准备，面试效果极其好，高薪）

我们项目中全面使用微服务框架，基于SpringBoot来构建项目，使用SpringCloud微服务框来开发我们业务。Eureka完成注册中心，Ribbon完成前端负载均衡，Feign封装REST方式支持。Hystrix完成断路器，Zuul完成API网关，Sidecar完成异构开发语言调用，NodeJS。SpringCouldConfig配置中心+Git版本控制。

SpringCloud

Why?

Dubbo

SOA 面向架构编程

N面向

1. 面向过程（生活中思考方式）
2. 面向对象 OOP
3. 面向切面 AOP
4. 面向服务、架构 SOA

ESB 企业级服务总线（热插拔）太重（所有项目）

延伸，提供服务，微服务dubbo。Dubbo SOA最佳实践。

Dubbo两件最重要的事情。服务动态发现Zookeeper（热插拔）。Dubbo RPC（netty） hession序列化，反序列化。

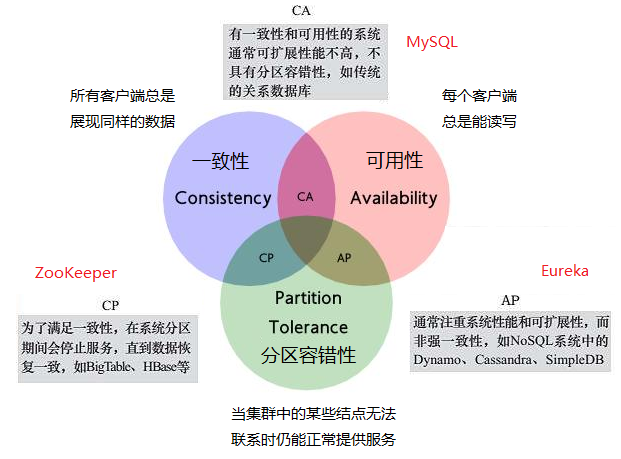
SpringCloud 底层httpclient+json(REST)

Eureka注册中心，异构开发语言直接支持

SpringCloud在每个方面都做到极致！注册中心

ZooKeeper、Eureka注册中心

CAP定理



注册中心业务，CP好呢？还是AP好呢？

注册中心特点，注册到注册中心信息不回频繁发生变化？不频繁

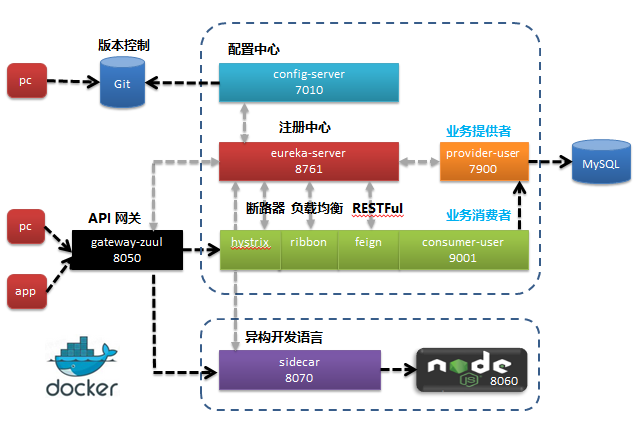
分布式系统设计定理，P分区容错性必须实现，实现方式AP侧重可用性，CP侧重一致性

Zookeeper支持集群，主从结构（leader,follows）zk节点如果有半数节点宕机，此时zk集群不让用。Zk如果在选举过程中，也不能使用。

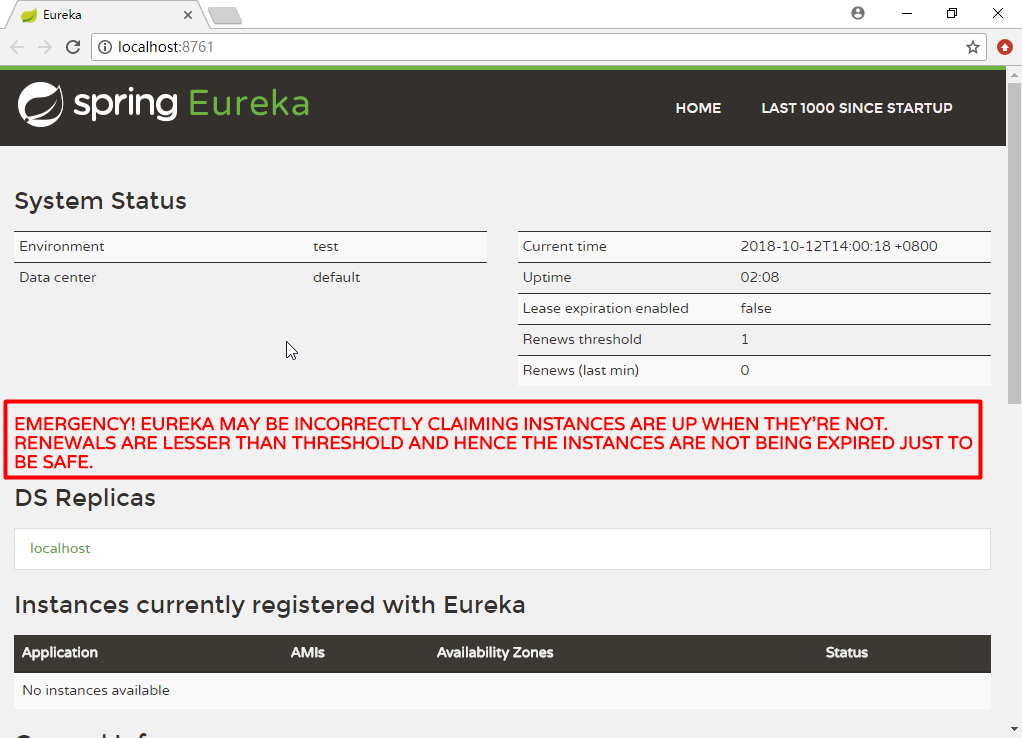
Eureka也支持集群，点对点，每两个点互为主从。哪怕剩下最后一个节点，仍然能继续访问。最后这个节点宕机，照样用。Server和client，client本地有缓存！Eureka宁可错误，也要试试。

主从中心Eureka设计更加合适！

阿里，为什么注册中心不是有ZooKeeper。



EurekaServer



超过阀值就会进入保护模式，动态列表不动态，如果调用失败服务列表仍然存在，不删除了，保证最大限度可用性，宁可连接错误的，也要试试。（设计目标）

目的：网络涌动异常，不会造成Eureka服务终端。

不是错误，在真实场景很少出现。

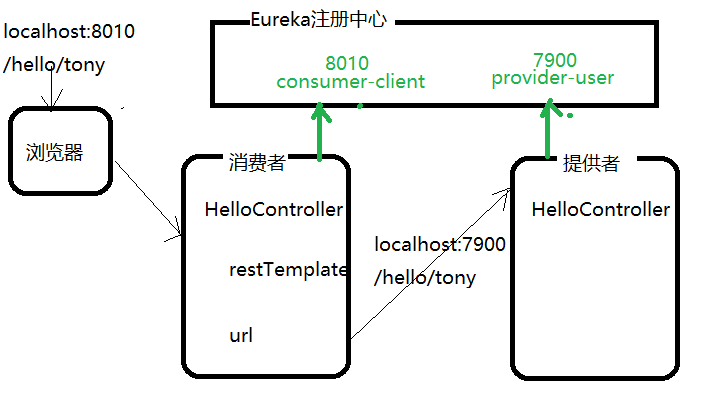
注册服务：

1. 服务提供者（服务业务）
2. 服务消费者（）

高并发，动态！！！整个架构松耦合。

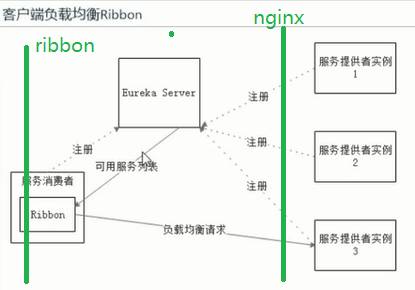
小结：

调用过程？



负载均衡Ribbon

nginx



怎么体现使用Eureka动态列表？

1. 在增加一个服务provider-user2，一样业务
2. 过程中也体现负载均衡
3. 故障自动恢复

Feign

如果使用RestTemplate学习成本比较高，不好

Dubbo对外暴露接口

Feign解决api太繁琐，而且版本升级api不同

WebService。SOA ESB后端，WebService前端结构

Apache CXF xml 封装HTTP，形成新的协议SAOP简单对象访问协议

webService支持异构开发语言。.net<>java 转换效率极其低。

txt > xml结构化 > json [{x,y},{}]

httpClient+json(REST)

J2EE jws规范jrs

Feign is a declarative web service client. 声明式

It makes writing web service clients easier. 写webservice程序非常容易

To use Feign create an interface and annotate it. 接口+注解

It has pluggable annotation support including Feign annotations and JAX-RS annotations. Feign also supports pluggable encoders and decoders. Get请求中文乱码

Spring Cloud adds support for Spring MVC annotations and

for using the same HttpMessageConverters used by default in Spring Web.

Spring Cloud integrates Ribbon and Eureka to provide a load balanced

http client when using Feign.

Feign底层会给我们写的接口来创建实例



使用jdk动态代理。

小结：

SpringCloud全家桶

微服务有两个解决方案：

Servlet茅草房，SSM三大框架 一间房，dubbo一套高级套房三室两厅，springcloud别墅

Dubbo阿里，3.0支持stream。将来我只是springcloud一个子集，只专注于RPC解决方案

1. Dubbo 基于RPC，hession序列化算法，压缩，二进制，长链接
2. springCloud 基于REST+json，httpclient

性能：dubbo远远超过springCloud

异构开发语言支持：springcloud使用json

SpringCloud

1. Eureka注册中心
2. Ribbon前端负载均衡
3. Feign 声明式封装RREST支持