# Dubbo系列

## Dubbo面试题

<https://blog.csdn.net/Y0Q2T57s/article/details/83005376>

## rpc和http restful的区别

rpc是远程过程调用,只能用服务本身提供的方法和接口与服务器无关，整个过程，请求，响应和方法客户端都是不可见的，异构的客户端很难与服务器通信，也很难解析服务器的响应请求。

**REST 是面向资源的，这个概念非常重要，而资源是通过 URI 进行暴露**

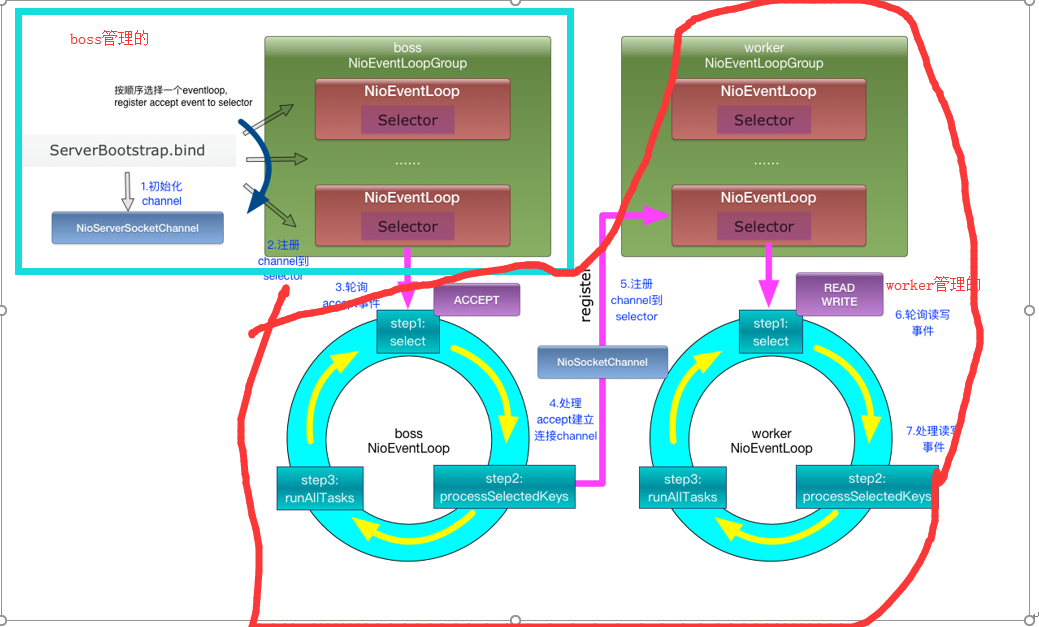
## RPC原理



|  |
| --- |
| 一次完整的RPC调用流程（同步调用，异步另说）如下：  **1）服务消费方（client）调用以本地调用方式调用服务；**  2）client stub接收到调用后负责将方法、参数等组装成能够进行网络传输的消息体；  3）client stub找到服务地址，并将消息发送到服务端；  4）server stub收到消息后进行解码；  5）server stub根据解码结果调用本地的服务；  6）本地服务执行并将结果返回给server stub；  7）server stub将返回结果打包成消息并发送至消费方；  8）client stub接收到消息，并进行解码；  **9）服务消费方得到最终结果。**  RPC框架的目标就是要2~8这些步骤都封装起来，这些细节对用户来说是透明的，不可见的。 |

rpc的底层就是netty

netty基本原理

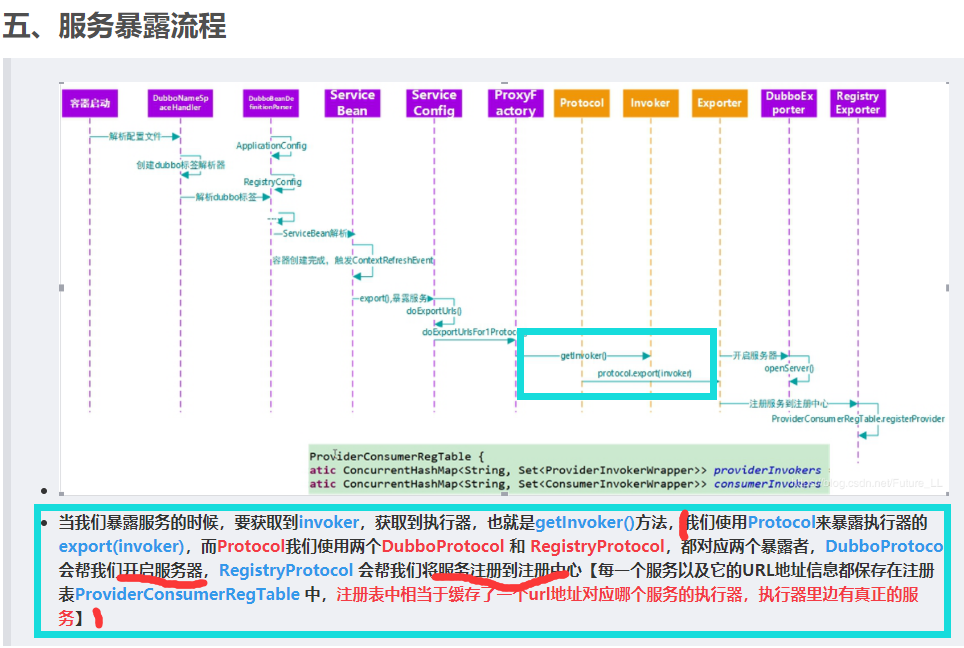


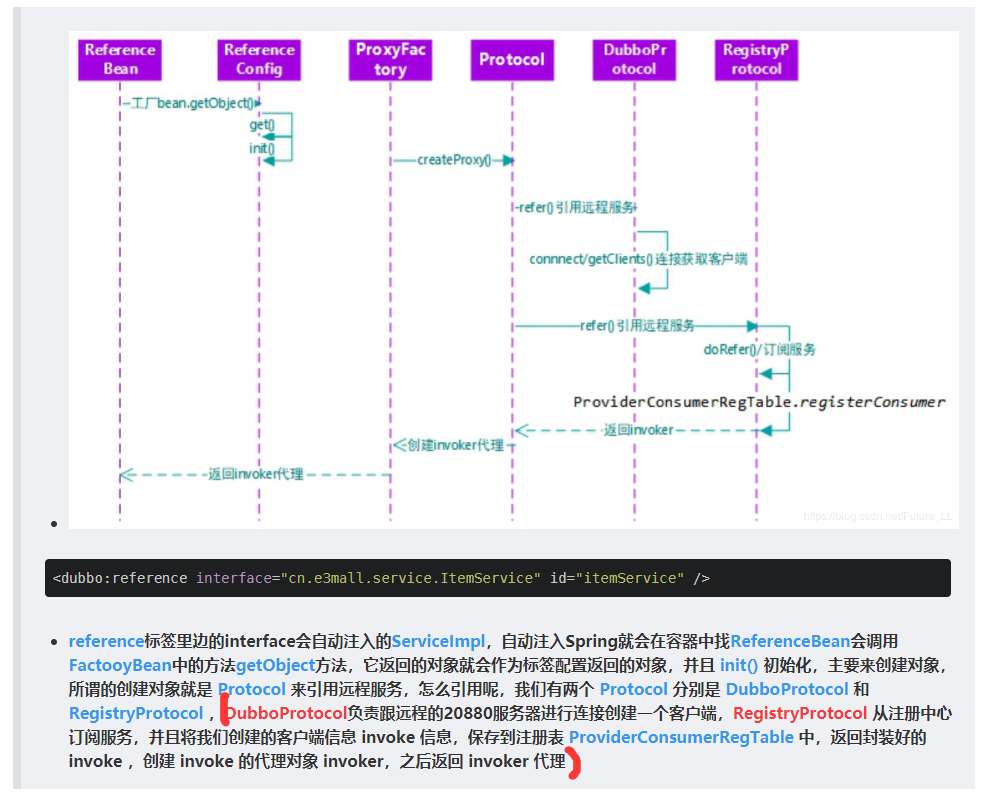
netty服务器启动，绑定监听端口 208080 所有给这个端口发的数据，netty服务器就能收到，netty服务器启动后会初始化通道，通道初始化好后会注册到selector中（selector是一个多路复用器），selector负责监听accept事件，accept事件就是当通道准备就绪，准备处理通道中的数据，怎么处理呢，netty与客户端建立连接，生成niosocketchannel(niosocketchannel是netty与客户端连接的通道)，把这个通道注册到另一个selector中，只不过这个selector监听read和write事件，相当于niosocketchannel中数据读准备就绪，数据写准备就绪，读准备就绪或者写准备就绪就会抛给任务队列来执行

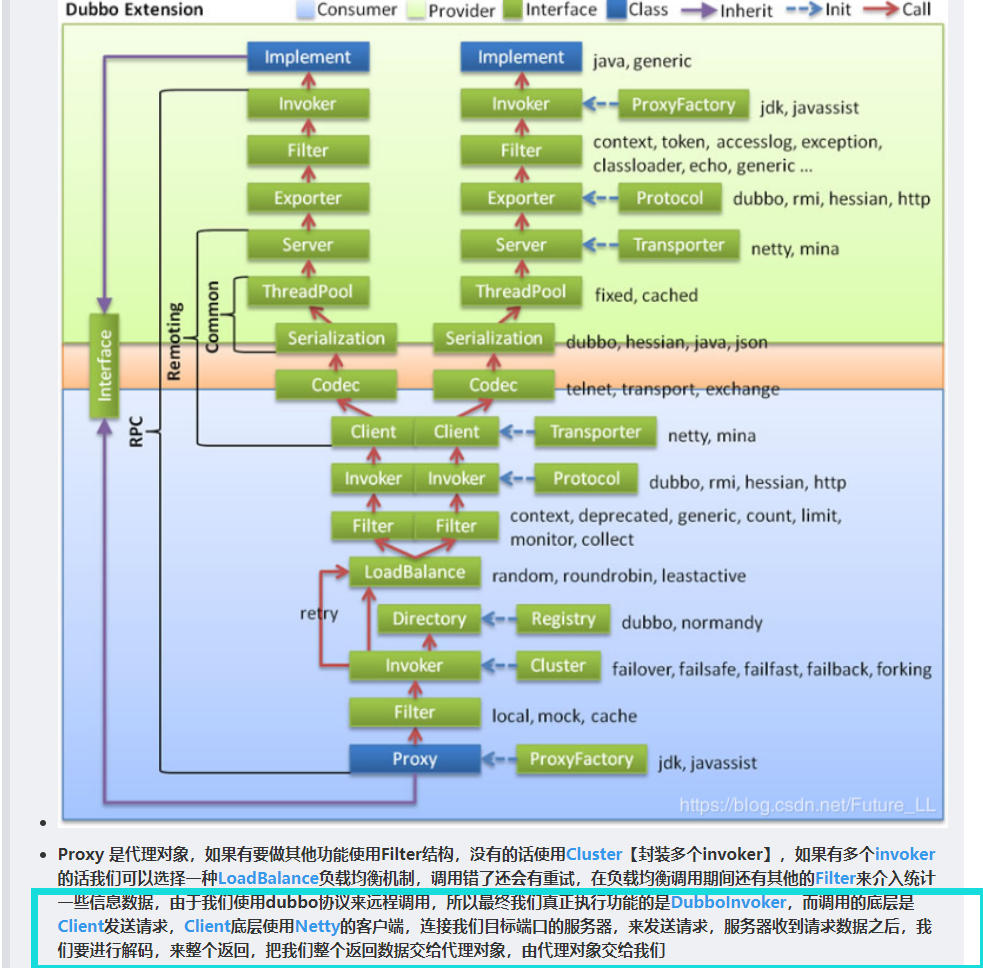
注意：boss线程组，用来监听来自20880所有连接准备就绪事件，worker线程组，就是准备就绪后要做什么工作

## Dubbo服务暴露，服务引用，服务调用

<https://blog.csdn.net/Future_LL/article/details/86721318>



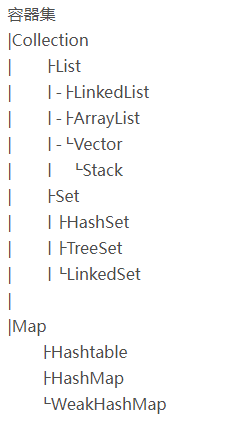




# 面试

## 蚂蚁花呗一面

### 1.Java容器有那些？那些是同步容器，那些是并发容器（容器代码）



同步容器（通过synchronized来实现同步的容器，如果有多个线程调用同步容器的方法，它们将会串行执行）

Vector

Stack

HashTable

Collections.synchronized方法生成

并发容器

ConcurrentHashMap：线程安全的HashMap的实现

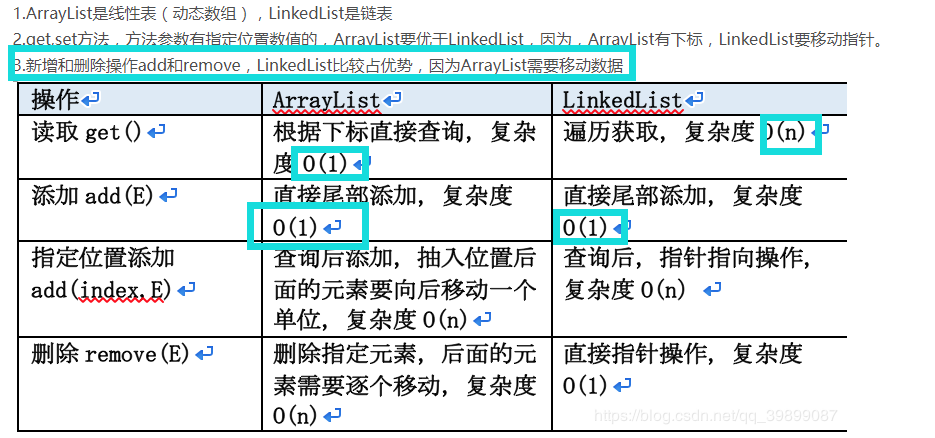
CopyOnWriteArrayList：线程安全且在读操作时无锁的ArrayList

CopyOnWriteArraySet：基于CopyOnWriteArrayList，不添加重复元素

ArrayBlockingQueue：基于数组、先进先出、线程安全，可实现指定时间的阻塞读写，并且容量可以限制

LinkedBlockingQueue：基于链表实现，读写各用一把锁，在高并发读写操作都多的情况下，性能优于ArrayBlockingQueue

### 2ArrayList和linkedList的插入和访问的事件复杂度



### 3.java反射原理，注解原理

#### 反射原理

1.将Java文件保存到本地硬盘

2.编译Java文件，生成.class文件

3.使用Java虚拟机（JVM）将字节码文件加载到内存

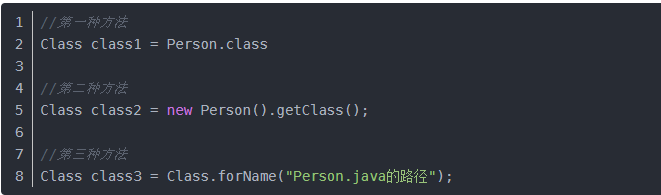
4.字节码文件在内存中使用Class类表示

5.使用反射的时候，首先获取到Class类，就可以得到class文件里的所有内容，包含属性、构造方法、普通方法

6.属性通过Filed类表示

7.构造方法通过Constructor表示

8.普通方法通过Method表示



#### 注解原理

注解是在编译时由编译器进行处理，编译器对注解符号处理并附加到class结构中（即保存到class结构的attributes属性中）

比如在类A上添加一个注解B，当类A被编译的时候，由于注解是作用在类上，所以此属性被添加到类的属性集上，即B注解的键值对会被记录起来。。

其实注解被编译后的本质就是一个继承Annotation接口的接口，

### 4新生代分为几个区？使用什么算法进行垃圾回收？为什么使用这个算法

一个eden，两个survvivor 8:1:1

在新生代中，每次垃圾收集时都发现有大批对象死去，只有少量存活，那就选用复制算法，只需要付出少量存活对象的复制成本就可以完成收集。

### 5HashMap在什么情况下会扩容，或者那些操作会导致扩容

Java中HashMap<K,V>的数据结构,虽然数据结构上稍微有点差异,不过思想都是一样的.我们还是以HashMap<K,V>进行讲解,我们知道HashMap<K,V>有一个叫装载因子的东西,默认情况下HashMap<K,V>的装载因子是75%这是在时间和空间上寻求的一个折衷.那么什么是所谓的装载因子,装载因子其实是用来判断当前的HashMap<K,V>中存放的数据量,如果我们存放的数据量大于了75%,那么HashMap<K,V>就需要进行扩容操作,扩容的空间大小就是原来空间的1.5倍，已有元素个数的2倍。0.75 \*ｌｅｎｇｔｈ。但是扩容的时候需要reshash操作,其实就是讲所有的数据重新计算HashCode,然后赋给新的HashMap<K,V>,rehash的过程是非常耗费时间和空间的,因此在我们对HashMap的大小进行控制的时候,应该要进行相当的考虑.还有一个误区(HashMap<K,V>可不是无限大的.)

多线程下HashMap为啥会出现死循环

Resize步骤：

1. 扩容：创建一个新的Entity空数组，长度是原数组的2倍。
2. ReHash:遍历Entry数组，把所有的Entry重新hash到新数组，为什么要重新hash呢？因为长度扩大以后，hash的规则也随之改变

多个线程rehash的时候会出现死循环

### 6HashMap push方法的执行过程

<https://blog.csdn.net/AJ1101/article/details/79413939>

### 7hashmap检测到hash冲突后，将元素插入在链表的末尾还是开头

8jdk1.8还采用了红黑树，讲讲红黑树的特性，为什么人家一定要用红黑树而不是avl，b数之类的？

9https和http区别，有没有用过其他安全传输手段

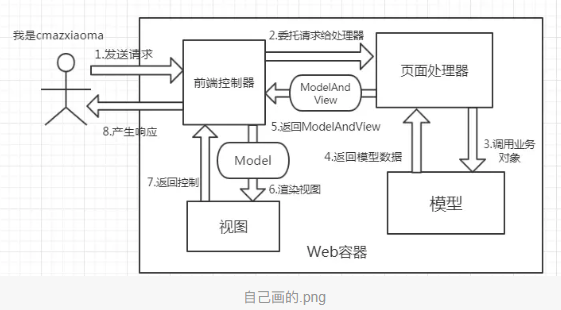
HTTP+Security s可以是TLS(安全传输层协议)，SSL（安全套接字）

* OpenVPN——这个协议可以实现任何出发点到目的地之间的数据安全连接，OpenVPN通过建立一个加密的UDP实现这项功能。其他的一些协议只与特定的操作系统兼容,OpenVPN则可以兼容任何的系统。

10线程池的工作原理，几个重要的参数，然后给了几个参数分析线程池会怎么做，最后问阻塞队列的作用是什么

11linux怎么查看系统负载情况

12请详细描述springmvc处理请求全流程



1：首先客户端发送一个http请求，web服务器接受这个请求，然后web容器交给DispatcherSevlet处理。

2：DispatcherSevlet将负责将请求分发。DispatcherSevlet可以认为是spring提供的前端控制器，所有的请求都要经过他来进行统一的分发。

3:在DispatcherSevlet将请求分发给spring controller之前，需要借助于spring提供的HandleMapping定位到具体的controller。（HandleMapping是这样的一种对象，他能够完成客户请求和controller之间的映射。在structs中，这种映射是通过struts-config.xml文件完成的。spring为controller提供了若干实现，例如：BeanNameUrlHandlerMapping有，SimpleUrlHandlerMapping，CommonsPathMapHandlerMapping。）

4:spring controller将处理来自DispatcherSevlet的请求。spring的controller类似于structs的action。能够接受HttpServletRequest和HttpServletResponse。spring为controller提供了若干实现类，由于controller要为并发用户处理上述请求，因此实现controller接口时，必须保证线程安全并且可重用。controller将处理用户请求，这和structs中的action是一致的。

5:一旦controller处理完用户请求，则返回ModelAndView对象给DispatcherSevlet前端控制。ModelAndView包含了模型（model）和视图（view）。从宏观角度来看，DispatcherSevlet是整个web应用的控制器。从微观来看，controller是单个http请求的控制器。而ModelAndView是http请求过程中返回的模型和视图。

6:ModelAndView中包含了视图逻辑名与模型数据信息。DispatcherSevlet通过ViewResolver完成逻辑视图名到真实视图对象的解析工作。

7:当得到真实对象的view，DispatcherServlet就使用这个View对象对ModelAndView中的模型数据进行视图渲染。然后将渲染后的结果返回给客户。

13讲一讲AtomicInteger,为什么要用CAS而不是synchronized

## 蚂蚁金服电话二面

1. 自我介绍，工作经历，技术栈
2. 项目中你学到了什么技术（把3个项目具体描述了很久）
3. 微服务划分的粒度
4. 微服务的高可用是怎么保证的
5. 常用的负载均衡，该怎么用
6. 网关能够为后端服务带来那些好处
7. Spring bean的生命周期
8. HashSet是不是线程安全的？为什么不是线程安全的

# 待完成事件

容器的底层源码

设计模式