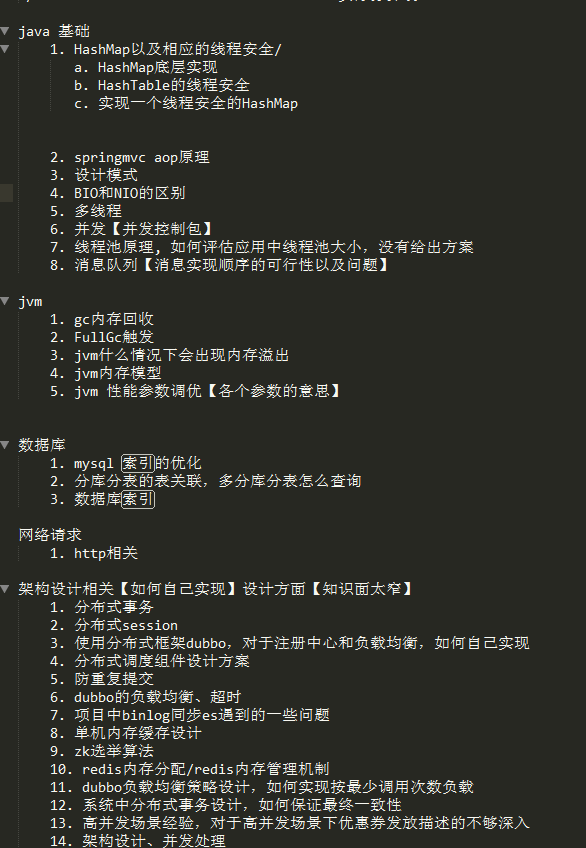
# Java面试题集



## Java基础

### JAVA 中的几种基本数据类型是什么，各自占用多少字节。

char 16

byte 8

short 16

int 32

long 64

boolean 1

float 32

double 64

### String 类能被继承吗，为什么。

String类型不能被继承，因为String为final修饰的类，final修饰的类不能被继承。

### 3. String，Stringbuffer，StringBuilder 的区别。

String字符串常量（final修饰，不可被继承），String是常量，当创建之后既不可被改变。（可以通过StringBuffer和StringBuilder创建String对象）

StringBuffer字符串变量线程安全，也是final类别的，不允许被继承，其中的绝大多数方法都进行了同步处理，包括常用的Append方法都是synchronize修饰。其toString方法会将对象缓存，以减少元素复制开销。

public synchronized String toString() {   
if (toStringCache == null) {   
toStringCache = Arrays.copyOfRange(value, 0, count);   
}   
return new String(toStringCache, true);   
}

StringBuilder字符串变量线程不安全，JDK1.5出现。与StringBuffer一样都继承和实现了同样的接口类方法，方法除了没被synchronize修饰外基本一致。不同之处在于最后toString的时候，会直返回一个新对象。New String();

public String toString() {   
// Create a copy, don’t share the array   
return new String(value, 0, count);   
}

### 4.ArrayList 和 LinkedList 有什么区别。

1、ArrayList是基于索引的数据接口，它的底层是数组。它可以以O(1)时间复杂度对元素进行随机访问。与此对应，LinkedList是以元素列表的形式存储它的数据，每一个元素都和它的前一个和后一个元素链接在一起，在这种情况下，查找某个元素的时间复杂度是O(n)。   
2、相对于ArrayList，LinkedList的插入，添加，删除操作速度更快，因为当元素被添加到集合任意位置的时候，不需要像数组那样重新计算大小或者是更新索引。   
3、LinkedList比ArrayList更占内存

### 5. 讲讲类的实例化顺序，比如父类静态数据，构造函数，字段，子类静态数据，构造函数， 字段，当 new 的时候，他们的执行顺序。

父类的静态属性

父类的静态代码块

子类类静态属性

子类静态代码块

父类成员变量

父类构造方法

子类成员变量

子类构造方法

### 6. 用过哪些 Map 类，都有什么区别，HashMap 是线程安全的吗,并发下使用的 Map 是什么， 他们内部原理分别是什么，比如存储方式，hashcode，扩容，默认容量等。

HashMap HashTable ConcurrentHashMap TreeMap LinkedHashMap

#### HashMap的实现原理

1、基于哈希表（数组+链表+二叉树（红黑树））JDK1.8

2、默认加载因子为0.75，默认数组大小是16（1 << 4）

3、把对象存储到哈希表中，如何存储？

把key对象通过hash()方法计算hash值，然后用这个hash值对数组长度取余数（默认16），来决定对该key的对象在数组中的位置index，

当这个位置有多个对象时，以链表结构存储，JDK1.8后，当链表长度大于8时，链表将转换为红黑树结构存储。

（这样的目的是为了取值更快，存储数据量越大，性能表现越明显。）

4、扩容原理：当数组的容量超过了负载因子，你们表示该数组需要扩容，如何扩容？

扩充算法是：当前数组容量的<<1（相对应乘2），扩大一倍，扩容次数过多，会影响性能，每次扩容表示哈希表重新散列（重新计算每个对象的存储位置），

我们在开发中尽量要减少扩充次数带来的性能问题。

5、线程不安全，适合在单线程中使用。

### 7. JAVA8 的 ConcurrentHashMap 为什么放弃了分段锁，有什么问题吗，如果你来设计， 你如何设计。

### 8. 有没有有顺序的 Map 实现类，如果有，他们是怎么保证有序的。

SortedMap<K,V>

### 9. 抽象类和接口的区别，类可以继承多个类么，接口可以继承多个接口么,类可以实现多个接口么。

类不能继承多个类，接口不能继承多个接口，类可以实现多个接口。

**抽象类和接口的区别：**

1、抽象类和接口都不能直接实例化，如果要实例化，抽象类变量必须指向实现所有抽象方法的子类对象，接口变量必须指向实现所有接口方法的类对象。

2、抽象类要被子类继承，接口要被类实现。

3、接口只能做方法申明，抽象类中可以做方法申明，也可以做方法实现

4、接口里定义的变量只能是公共的静态的常量，抽象类中的变量是普通变量。

5、抽象类里的抽象方法必须全部被子类所实现，如果子类不能全部实现父类抽象方法，那么该子类只能是抽象类。同样，一个实现接口的时候，如不能全部实现接口方法，那么该类也只能为抽象类。

6、抽象方法只能申明，不能实现。abstract void abc();不能写成abstract void abc(){}。

7、抽象类里可以没有抽象方法

8、如果一个类里有抽象方法，那么这个类只能是抽象类

9、抽象方法要被实现，所以不能是静态的，也不能是私有的。

10、接口可继承接口，并可多继承接口，但类只能单根继承。

## JVM知识

参考：https://blog.csdn.net/u012257955/article/details/70890702

### 什么情况下会发生栈内存溢出。

线程在请求栈深度的时候超过了虚拟机设置的最大栈深度，就会抛出StackOverflowError异常；栈向虚拟机无法申请到更多内存，就会抛出OutOfMemoryError异常。

### 2. JVM的内存结构，Eden和Survivor比例

Jvm内存结构分为**堆内存**，**方法区**，**程序计数器**，**本地方法区**，和**虚拟机栈**。

**堆内存** 是java虚拟机所管理的最大一块内存，被所有线程共享，主要存储对象实例和数组等。

**方法区** 线程共享内存区域，存储被虚拟机加载的类信息，常量，静态变量等数据。

**程序计数器** 标记线程执行字节码的行数，线程独享。

**本地方区** 和虚拟机栈类似区别是本方法栈执行native方法，虚拟机栈执行java方法。

**虚拟机栈** 虚拟机栈是线程私有的，每个方法在执行的同时，都会创建一个栈桢，用于存储局部变量表，操作数栈，动态连接和方法出口等信息的。一个方法从调用到返回的过程就是栈桢入栈到出栈的过程；

Eden 和Survivor 的比例：8:1 两块survivor （To/From） Old

### JVM内存为什么要分成新生代，老年代，持久代。新生代中为什么要分为Eden和Survivor。

参考：https://blog.csdn.net/u011445156/article/details/73441618

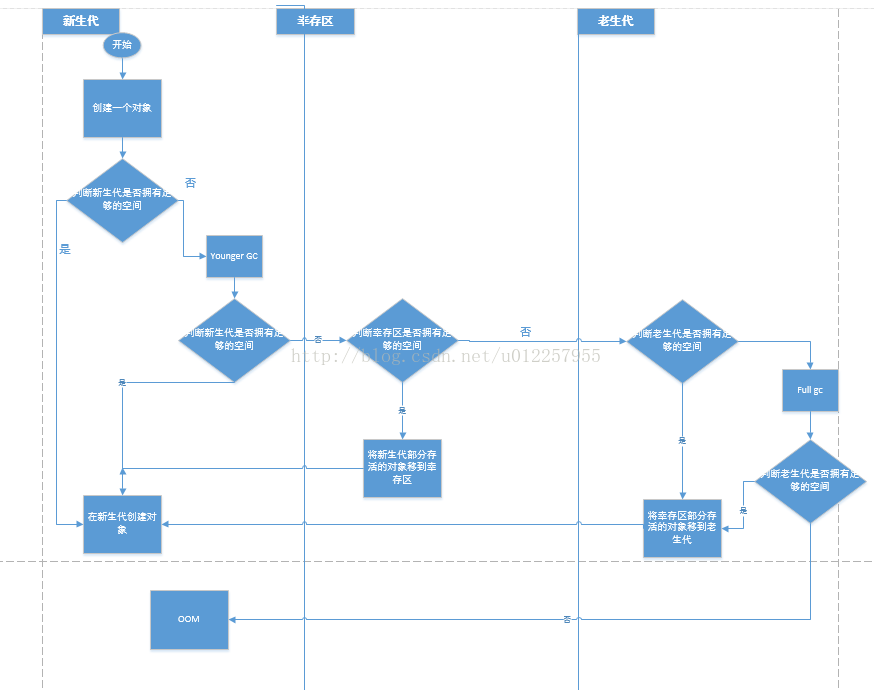
提高GC的效率，因为不是所有的对象都需要频繁的去GC，频繁GC很耗时的。

新生代的GC频繁，老年代相对低，持久代不进行GC。

**为什么新生代分Eden和Survivor** 如果没有分Survivor，Eden区每进行一次minor GC，存活的对象就会被送到老年代。老年代很快就会被填满，老年代一旦填满就会触发Full GC，频繁的FullGC消耗的时间是非常可观的，这一点会影响大型程序的执行速度和响应速度。

**为什么设置两个Survivor区** 整理内存碎片化，保证有一块空的survivor用来整理连续的内存空间。

### JVM 中一次完整的 GC 流程是怎样的，对象如何晋升到老年代，说说你知道的几种主要的 JVM 参数。



GC流程图

对象晋升老生代一共有三个可能：

1.当对象达到成年，经历过15次GC（默认15次，可配置），对象就晋升为老生代

2.大的对象会直接在老生代创建

3.新生代跟幸存区内存不足时，对象可能晋升到老生代

jvm参数：

-Xms 初始化堆大小

-Xmx 堆最大内存

-Xss 栈内存

-XX：PermSize 初始化永久带内存

-XX：MaxPermSize最大永久带内存

### 5. 你知道哪几种垃圾收集器，各自的优缺点，重点讲下 cms 和 G1，包括原理，流程，优缺点。

串行垃圾收集器：收集时间长，停顿时间久

并发垃圾收集器：碎片空间多

CMS：并发标记清除。主要步骤有：初始收集，并发标记，重新标记，并发清除（删除），重置。

G1：主要步骤：初始标记，并发标记，重新标记，复制清除（整理）。

CMS的缺点是对cpu的要求比较高。G1是将内存化成了多块，所有对内段的大小有很大的要求。

CMS是清除，所以会存在很大的内存碎片。G1是整理，所以碎片空间较小。

CMS 初始收集 -> 并发标记 -> 重新标记 -> 并发清除 -> 重置。

G1 初始标记 -> 并发标记 -> 重新标记 -> 复制清除。

### 6. 垃圾回收算法的实现原理。

常用的垃圾回收算法有两种： **引用计数**和**可达性分析**

引用计数是增加一个字段来标识当前的引用次数，引用计数为0的就是可以GC的。但是引用计数不能解决循环引用的问题  
可达性分析：就是通过一系列GC ROOT的对象作为起点，向下搜索，搜索所有没有与当前对象GC ROOT 有引用关系的对象。这些对象就是可以GC的。

### 7. 当出现了内存溢出，你怎么排错。

1.首先控制台查看错误日志

2.然后使用jdk自带的jvisualvm工具查看系统的堆栈日志

3.定位出内存溢出的空间：堆，栈还是永久代（jdk8以后不会出现永久代的内存溢出）。

4.如果是堆内存溢出，看是否创建了超大的对象

5.如果是栈内存溢出，看是否创建了超大的对象，或者产生了死循环。

### 8. JVM 内存模型的相关知识了解多少，比如重排序，内存屏障，happen-before，主内存， 工作内存等。

重排序：jvm虚拟机允许在不影响代码最终结果的情况下，可以乱序执行。

内存屏障：可以阻挡编译器的优化，也可以阻挡处理器的优化

happens-before原则：

1：一个线程的A操作总是在B之前，那多线程的A操作肯定实在B之前。  
2：monitor 再加锁的情况下，持有锁的肯定先执行。  
3：volatile修饰的情况下，写先于读发生  
4：线程启动在一起之前 strat  
5：线程死亡在一切之后 end  
6：线程操作在一切线程中断之前  
7：一个对象构造函数的结束都该对象的finalizer的开始之前  
8：传递性，如果A肯定在B之前，B肯定在C之前，那A肯定是在C之前。

主内存：所有线程共享的内存空间

工作内存：每个线程特有的内存空间

### 9. 简单说说你了解的类加载器，可以打破双亲委派么，怎么打破。

Bootstra ClassLoader（引导类加载器）

Extension ClassLoader（扩展类加载器）

Application ClassLoader（系统加载器）

Customer ClassLoader（自定义类加载器）

双亲委派：双亲委派模式会保证父类加载器先加载类。

### 10. 讲讲 JAVA 的反射机制。

在运行期间，对于任意一个类，都能知道当前类的属性和方法，对于一个对象，都能调用类的方法和属性，这种动态加载机制就是Java反射机制。

### 11. 你们线上应用的 JVM 参数有哪些。

-XX:PermSize=128M  
-XX:MaxPermSize=512m  
-XX:PermSize=128M  
-XX:MaxPermSize=512m  
-Xms512m  
-Xmx1024m  
-XX:PermSize=640m  
-XX:MaxPermSize=1280m  
-XX:NewSize=64m  
-XX:MaxNewSize=256m  
-verbose:gc  
-XX:+PrintGCDetails  
-XX:+PrintGCTimeStamps

### 12. g1 和 cms 区别,吞吐量优先和响应优先的垃圾收集器选择。

Cms是以获取最短回收停顿时间为目标的收集器。基于标记-清除算法实现。比较占用cpu资源，切易造成碎片。

G1是面向服务端的垃圾收集器，是jdk9默认的收集器，基于标记-整理算法实现。可利用多核、多cpu，保留分代，实现可预测停顿，可控。

### 13. 怎么打出线程栈信息。

### 14. 请解释如下 jvm 参数的含义：

-server -Xms512m -Xmx512m -Xss1024K -XX:PermSize=256m -XX:MaxPermSize=512m -XX:MaxTenuringThreshold=20 XX:CMSInitiatingOccupancyFraction=80 -XX:+UseCMSInitiatingOccupancyOnly。

## 操作系统

### 1. Linux 系统下你关注过哪些内核参数，说说你知道的。

### Linux 下 IO 模型有几种，各自的含义是什么。

**同步/异步 和 阻塞/非阻塞**

**同步和异步（消息的通知机制）**

同步sync：发出一个函数调用时，会一直等待，直到方法返回才继续执行后续操作。（几乎Java的使用方法都是同步调用的）

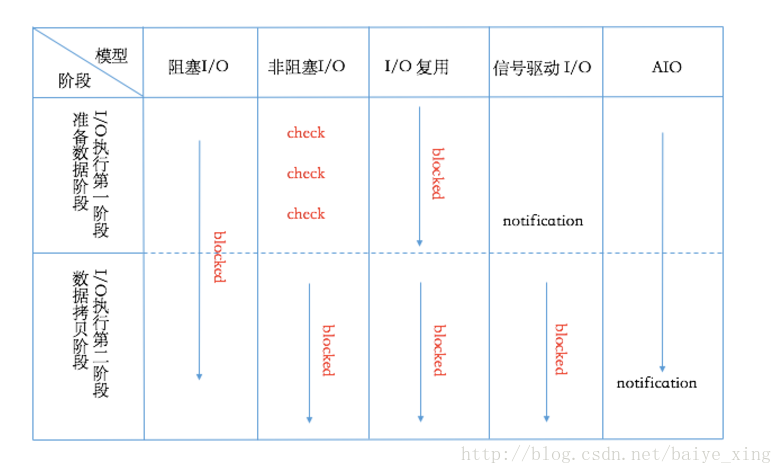
异步Async：发出一个函数调用后，调用者可以在没得到结果之前继续后续操作。当这个调用结束后，一般通过状态、通知或者回调来通知调用者。对于异步来说，调用的返回并不受调用者控制。

**阻塞和非阻塞（程序等待调用结果时的状态）**

阻塞：就是调用结果返回之前，该执行的线程会被挂起，不释放CPU执行权，线程不做其他事情，等待调用结果的返回，返回后才能继续执行。

非阻塞：就是在没有获取调用结果之前，不是一直等待，线程可以往下执行，如果是同步的，通过轮询的方式检查有没有调用结果返回，如果是异步的，就会通知回调。

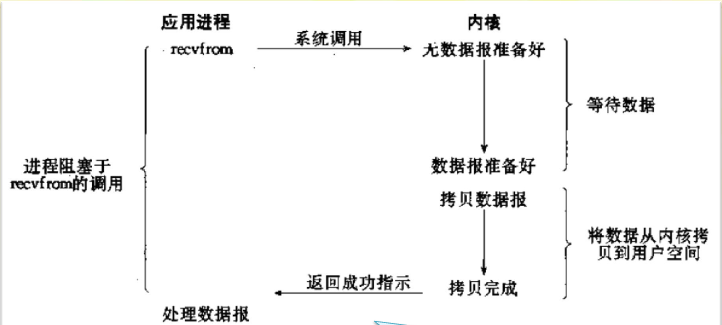
**五种IO模型（）**

****

**（1）阻塞式IO（BIO）**

简介：进程会一直阻塞，直到数据拷贝完成。

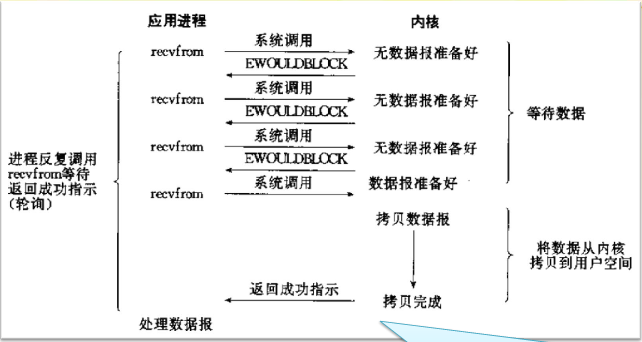
应用程序调用一个IO函数，导致应用程序阻塞，等待数据准备好。如果数据准备好了，从内核拷贝到用户空间。执行完毕后，IO函数会向应用程序返回成功响应，应用程序得到响应后，就不在阻塞，进行后续工作。



**（2）非阻塞式IO（NIO）**

简介：非阻塞IO通过进程反复调用IO函数（多次系统调用，并马上返回）；在数据拷贝过程中，进程是阻塞的。

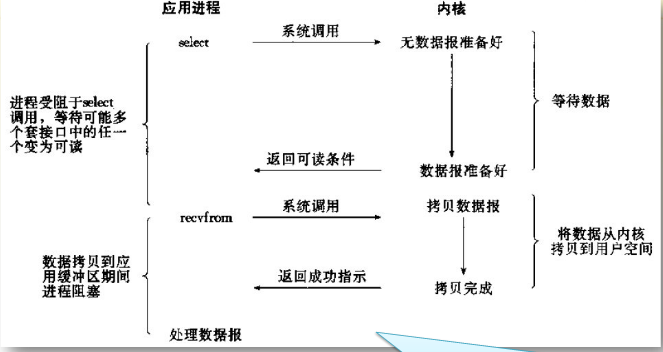
应用程序调用一个IO函数，这个IO操作会从内核中立刻返回。但是这个IO函数具体要执行的事情（写数据）可能还没完成。而对于应用程序，虽然这个IO很快就返回了，但是它并不知道这个IO操作是否真的成功了，为了知道IO操作是否成功，一般有两种策略，一是要应用程序主动轮询，知道数据准备好为止，在这个不断询问的过程中，会大量占用CPU的时间。二是采用IO通知机制，比如IO多路复用或者信号驱动IO。



**（3）IO多路复用模型**

简介：比阻塞IO并没有什么优越性，关键是能实现同时对多个IO端口进行监听。

IO复用模型会用到select、poll、epoll函数，这几个函数也会使进程阻塞，但是和阻塞IO不同的，这两个函数可以同时阻塞多个IO操作，而且可以同时对多个读操作，多个写操作的IO函数进行检测、直到有数据可读或者可写时，才真正调用IO操作函数。

****

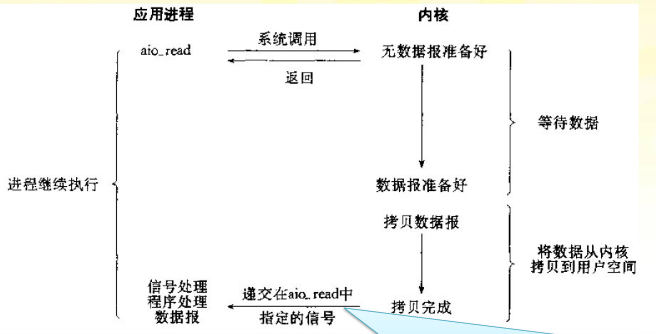
**（4）信号驱动IO模型（signal-driver IO）**

进程预先告知内核，使得当某个Socketfd有events事件发生时，内核使用信号通知相关进程。

**（5）异步IO模型**

简介：告知内核启动某个操作，并让内核在整个操作完成后通知我们；数据拷贝的时候进程无需要阻塞。

当一个异步过程调用发出后，调用者不能立即得到结果，实际处理这个调用函数完成后，通过状态、通知和回调者的输入输出操作。



### 3. epoll 和 poll 有什么区别。

Poll在每次调用都需要拷贝一次fd\_set（或者pollfd），在描述数量大时开销较大。内核需要遍历传入的fd\_set，效率较低。

Epoll主要通过epoll-ctl使每次注册新事件时拷贝一次所有的fd；epoll\_wait只需要轮询一个就绪队列是否为空，这是因为epoll采用了回调机制，将就绪的fd插入到就绪队列，因此不像select和poll中，需要轮询一个fd\_set（很耗CPU）。

### 4. 平时用到哪些 Linux 命令。

Ls ，find，tar，tail，cp，rm，mv，vi，vim，grep，ps，kill等

### 5. 用一行命令查看文件的最后五行。

Tail -n 5 filename

### 6. 用一行命令输出正在运行的 java 进程。

ps -ef|grep java

### 7. 介绍下你理解的操作系统中线程切换过程。

控制权的转换，根据优先级切换上下文（用户，寄存器，系统）   
<http://www.cnblogs.com/kkshaq/p/4544426.html>

### 8. 进程和线程的区别。

进程：是程序运行的实例，是系统进行资源分配和调度的一个独立单位，它包括独立的地址空间，资源以及1个或多个线程。

线程：cpu调度和分配的最小单位，一个轻量级的进程。

### 9. top 命令之后有哪些内容，有什么作用。

top命令是Linux下常用的性能分析工具，能够实时显示系统中各个进程的资源占用状况，类似于Windows的任务管理器。

### 10. 线上 CPU 爆高，请问你如何找到问题所在

## 四、开源框架知识

### 1. 简单讲讲tomcat结构，以及其类加载器流程，线程模型等。

### 2. tomcat 如何调优，涉及哪些参数。

### 3. 讲讲 Spring 加载流程。

### 4. Spring AOP 的实现原理。

### 5.讲讲 Spring 事务的传播属性。

### 6. Spring 如何管理事务的。

### 7. Spring 怎么配置事务（具体说出一些关键的 xml 元素）。

### 8. 说说你对 Spring 的理解，非单例注入的原理？它的生命周期？循环注入的原理，aop 的实 现原理，说说 aop 中的几个术语，它们是怎么相互工作的。

### 9. Springmvc 中 DispatcherServlet 初始化过程。

### 10. netty 的线程模型，netty 如何基于 reactor 模型上实现的。

### 11. 为什么选择 netty。 12. 什么是 TCP 粘包，拆包。解决方式是什么。

### 13. netty 的 fashwheeltimer 的用法，实现原理，是否出现过调用不够准时，怎么解决。

### 14. netty 的心跳处理在弱网下怎么办。

### 15. netty 的通讯协议是什么样的。

### 16. springmvc 用到的注解，作用是什么，原理。

### 17. springboot 启动机制。