## 一、Spring

### 1、谈谈自己对SpringAOP和IOC的理解，核心机制、原理及项目中的运用。

Ioc：控制反转是一种设计模式，将对象的控制权交由spring容器管理，传统的程序中，通常由调用者创建被调用者的实例，在Spring中创建实例的工作由spring完成，最后注入给调用者。  
aop:面向切面编程是一种编程思想，提供声明式事务管理，支持用户自定义切面。

AOP和IOC在项目中都是为了解决项目中耦合度较高的问题，aop常用语项目中的事务管理、日志以及权限拦截等。

### 2、Spring上下文的加载方式有哪些？

答：通过配置xml方式加载，有6种：

①、spring-mvc.xml中import引入

②、在web.xml中配置，应用服务去加载

③、引用资源用XMLBeanFactory

④、引用应用上下文classPathXMLApplicationContext

⑤、用文件系统的路径引用，用FileSystemXMLApplicationContext

⑥、web工程定制的加载方法，XMLWebApplicationContext

### 3、以往项目中是如何对事务进行管理的？

答：使用spring声明式事务管理。

①、在spring中配置事务管理器

②、开启事务注解

③、在对应的service层加Transactional注解

### 4、Autowired和resource的区别？

答：

共同点：两者都可以写在字段和setter方法上。

不同点：@Autowired是spring的注解，默认根据类型注入，如果想根据名称注入，则需要搭配@Qualifier注解使用。

@resource是j2ee的注解，它有name和type这两个重要的属性，当使用name时为根据名称注入，当使用type时则根据类型注入，当都不指定时通过反射机制默认使用name注入。

### 5、使用SpringBoot和以往的SSM有什么不同，为什么要选用SpringBoot？

答：springboot开发效率高，可以提高生产力，所以选用springboot，具体选择得看场景。  
springboot是一个应用是一个可执行的jar，内置tomcat，并将原来的xml配置简化成java配置。

### 6、谈谈SpringCloud和Dubbo？

### 7、描述一下SpringMVC的工作流程。其包含哪4个主要组件？

答：流程步骤如下，  
1. 用户发送请求至前端控制器DispatcherServlet。  
2. DispatcherServlet收到请求调用HandlerMapping处理器映射器。  
3. 处理器映射器找到具体的处理器，生成处理器对象及处理器拦截器（如果有则生成）一并返回给DispatcherServlet。  
4. DispatcherServlet调用HandlerAdapter处理器适配器。  
5. HandlerAdapter经过适配调用具体的处理器（controller，也叫后端控制器）。  
6. Controller执行完成返回ModelAndView。  
7. HandlerAdapter将Controller返回的执行结果ModelAndView返回给DispatcherServlet。  
8. DispatcherServlet将ModelAndView传给ViewReslover视图解析器。  
9. ViewReslover解析之后返回具体的view。  
10.  DispatcherServlet根据View进行渲染视图（即将模型数据填充至视图jsp/freemaker..中）。11. DispatcherServlet响应用户。

主要包含如下：  
1、前端控制器DispatcherServlet（不需要程序员开发）  
作用接受请求，响应结果，相当于转发器,中央处理器  
有了DispatcherServlet，减少了与其他组件之间的耦合度  
2、处理器映射器HandlerMapping（不需要程序员开发）  
作用：根据请求的url查找Handler  
3、处理器Handler

作用：编写Handler时按照HandlerAdapter的要求去做，这样适配器才可以去正确执行Handler  
4、处理器适配器HandlerAdapter  
根据特定规则（HandlerAdapter的规则）执行handler  
5、视图解析器 View resolver  
作用：进行视图解析，根据逻辑视图名解析成真正的视图（view）  
5、视图View  
View是一个接口，实现类支持不同的view类型（jsp，framemark，pdf…）

### 8、描述一下bean的生命周期和加载过程。

答：  
1、实例化一个Bean－－也就是我们常说的new；  
2、按照Spring上下文对实例化的Bean进行配置－－也就是IOC注入；  
3、如果这个Bean已经实现了BeanNameAware接口，会调用它实现的setBeanName(String)方法，此处传递的就是Spring配置文件中Bean的id值  
4、如果这个Bean已经实现了BeanFactoryAware接口，会调用它实现的setBeanFactory(setBeanFactory(BeanFactory)传递的是Spring工厂自身（可以用这个方式来获取其它Bean，只需在Spring配置文件中配置一个普通的Bean就可以）；  
5、如果这个Bean已经实现了ApplicationContextAware接口，会调用setApplicationContext(ApplicationContext)方法，传入Spring上下文（同样这个方式也可以实现步骤4的内容，但比4更好，因为ApplicationContext是BeanFactory的子接口，有更多的实现方法）；  
6、如果这个Bean关联了BeanPostProcessor接口，将会调用postProcessBeforeInitialization(Object obj, String s)方法，BeanPostProcessor经常被用作是Bean内容的更改，并且由于这个是在Bean初始化结束时调用那个的方法，也可以被应用于内存或缓存技术；  
7、如果Bean在Spring配置文件中配置了init-method属性会自动调用其配置的初始化方法。8、如果这个Bean关联了BeanPostProcessor接口，将会调用postProcessAfterInitialization(Object obj, String s)方法、；      
9、当Bean不再需要时，会经过清理阶段，如果Bean实现了DisposableBean这个接口，会调用那个其实现的destroy()方法；  
10、最后，如果这个Bean的Spring配置中配置了destroy-method属性，会自动调用其配置的销毁方法。

### 在什么情况下事务会进行回滚？

答：在Java中异常的基类为Throwable，他有两个子类Exception与Error，同时RuntimeException就是Exception的子类；

RuntimeException，即运行时异常，为非受检（UNCHECKED）异常；Exception的其他子类异常，为非运行时异常，为受检异常（CHECKED）异常；

Spring 管理的事务：默认配置下，事务只会对Error与RuntimeException及其子类这些UNChecked异常，做出回滚。一般的Exception这些Checked异常不会发生回滚。

如果想让checked异常进行回滚，则可以通过注解@Transactional(rollbackFor=Exception.class)

和配置拦截器来实现。

### 10、描述一下AOP的代码实现？

答：共有3种方式实现，基于xml实现、基于注解实现、基于自定义注解实现。我在项目中主要利用注解实现，步骤如下，  
1.在spring-mvc.xml里加上自动代理的配置。

2.编写切面类，需要在类上加上@Aspect注解，在相应的@Before、@After等注解下方法实现具体的逻辑，在注解上还需通过execution表达式配置切点。

## 二、Mybatis

### 1、例举Mybatis里的常用标签。

Select，update，insert，delete，resultMap，sql，if，foreach，where等

### Mybatis二级缓存？

答：二级缓存需要借助外部程序实现，可以提高查询效率，可以利用ehcache、redis、memcached来实现分布式的缓存。

Redis：①配置redis.xml 设置redis服务连接各参数

②在配置文件中使用 <setting> 标签，设置开启二级缓存；

③在mapper.xml 中使用<cache type="com.demo.RedisCacheClass" /> 将cache映射到指定的RedisCacheClass类中；

④映射类RedisCacheClass 实现 MyBatis包中的Cache类，并重写其中各方法；

在重写各方法体中，使用redisFactory和redis服务建立连接，将缓存的数据加载到指定的redis内存中(putObject方法)或将redis服务中的数据从缓存中读取出来(getObject方法)；

在redis服务中写入和加载数据时需要借用spring-data-redis.jar中JdkSerializationRedisSerializer.class中的序列化(serialize)和反序列化方法(deserialize),此为包中封装的redis默认的序列化方法；

⑤映射类中的各方法重写完成后即可实现mybatis数据二级缓存到redis服务中；

### 3、#和$有什么不同，两者分别在什么场景下使用？

答：

1、#是将传入的值当做字符串的形式。

2、$是将传入的数据直接显示生成sql语句。

3、使用#可以很大程度上防止sql注入。(语句的拼接)。

4、但是如果使用在order by 中就需要使用 $。

### 4、#底层是如何实现防止SQL注入的？如果不用Mybaits你会通过什么方式实现防止SQL注入？

答：主要是利用了PreparedStatement，#是经过预编译的，而$则没有通过预编译，仅仅是取变量的值，不安全。

不用mybatis的话，则需要手动判断参数格式是否正确即可。

## 三、Java集合类

### 1、List和Set集合的区别？

答：list和set都继承Collection接口。  
List：元素有序可重复，查询元素效率高，插入删除元素效率较低，因为会引起其他元素位置变化。  
Set：无序不可重复，检索元素效率较低，插入和删除效率较高，因为不会改变元素位置。

### 2、ArrayList的加载因子，默认容量和扩容增量是多少？是容量不够了还是快不够了进行扩容？

答：ArrayList的底层是一个Object数组，容量为10，所以默认容量是10；每当使用add方法进行添加元素时，都会检查添加元素后的容量是否超出当前容量，如果超出则进行扩容，扩容增量是1.5倍。

### 3、谈谈HashMap的元素碰撞？如何解决hash冲突。

答：两个对象通过hashcode方法获取到它们的hash值是一样的，这就是hash碰撞。

防止哈希碰撞的最有效方法，就是扩大哈希值的取值空间，具体方法如下，

①开放地址法

②再哈希法（发生冲突时，再次进行hash函数计算，直到不冲突为止）

③拉链法（就是在每个位桶实现的时候，我们采用链表（jdk1.8之后采用链表+红黑树）的数据结构来去存取发生哈希冲突的输入域的关键字）

④建立一个公共溢出区

### 4、怎么去删除ArrayList中不符合条件的元素？

答：遍历ArrayList，找到要删的remove掉即可

### 5、Java线程安全的集合是通过什么手段实现线程安全的？

答：同步锁synchronized方法和lock  
concurrentHashmap使用CAS算法无锁化实现同步。

### 6、有什么其他方法可代替synchronized实现线程安全类。

答：利用CAS算法实现。但是不适合高并发场景，（可能会引发ABA问题）。

### 7、HashMap除了使用链表，还使用到了什么数据结构？

答：数组。  
数组：优点随机访问性强，查询速度快；缺点插入删除效率低，可能浪费内存，必须要有足够的连续内存空间，数组大小固定，不能动态扩展。

链表：优点插入删除速度快，内存利用率高，大小不固定，扩展容易；缺点不能随机查找，必须从第一个开始遍历，效率低下。

### 8、ArrayList中的元素是通过什么方式实现排序的？

答：通过使用comparator或者comparable方法。  
Comparator可以比较两个不同类型的对象，comparable则是自己和另一个对象进行比较

### 9、HashTable和HashMap的区别？

答：

1.hashtable是synchronized，线程安全，hashmap非synchronized，所以线程不安全。所以单线程情况下HashMap效率大于HashTable。

2.hashmap可以接受null键值，hashtable不可以  
3.两者的迭代器不同。  
4.HashMap不能保证随着时间的推移Map中的元素次序是不变的。

### 10、常用集合介绍？

答：

（1）Set集合

存放对象的引用，不允许有重复对象。

HashSet（），调用对象的hashCode（）方法，获得哈希码，然后再集合中计算存放对象的位置;

TreeSet（），继承ShortedSet接口，能够对集合中对象排序。默认排序方式是自然排序，但该方式只能对实现了Comparable接口的对象排序，java中对Integer、Byte、Double、Character、String等数值型和字符型对象都实现了该接口。

（2）List集合

线性集合接口，能够以线性方式储蓄对象，有序，并允许存放重复对象；

ArrayList：动态数组[可变长度的动态数组]；

优势：对于随机访问get和set，ArrayList使用索引的方式来快速定位对象的位置；ArrayList效率优于LinkedList，因为LinkedList要移动指针。

LinkedList：链表结构的集合

优势：在进行insert和remove动作时在效率上要比ArrayList要好得多；ArrayList是使用数组实现的,若要从数组中删除或插入某一个对象，需要移动后段的数组元素，从而会重新调整索引顺序,调整索引顺序会消耗一定的时间，所以速度上就会比LinkedList要慢许多；LinkedList适合用来实现Stack(堆栈)与Queue(队列),前者后进先出，后者是先进先出；

（3）Map集合

键值对存储结构的集合，对健要求唯一，如果加入已有的健，原有的值对象将被覆盖

HashMap：以数组方式存储key/value，按照哈希算法来存取键对象，线程非安全的；允许键、值均为null，元素迭代顺序是随机的；

TreeMap：基于红黑二叉树的NavigableMap的实现，线程非安全；键、值都不能为null，存储元素默认按键的升序排序，想要对存入TreeMap的元素重新进行排序，可实现Comparable接口或者实现Comparator接口；

总结：一般情况下我们选用HashMap，因为HashMap的键值对在取出时是随机的，其依据键的hashCode和键的equals方法存取数据，具有很快的访问速度，所以在Map中插入、删除及索引元素时其是效率最高的实现。而TreeMap的键值对在取出时是排过序的，所以效率会低点。

### map下有哪些线程安全的集合？

答：HashTable、SynchronizedMap、concurrentHashMap（推荐，无锁化）

### hashmap的get和put原理是什么？

答：

get：通过hashcode获取key的hash值，根据hash值可以计算出entry的下标位置，再通过equals方法比较传入的key和要获取的key的hash值是否相等，如果相等返回entry对象，entry对象使用getvalue方法即可。

Put：hashmap有两个重要属性，一个是容量（默认16），一个是加载因子（0.75），hashmap的初始化和扩容会用到这2个属性。

通过hashcode获取key的hash值，根据hash值计算元素下标位置，如果这个位置上没有元素，那么直接存放entry对象在哪个位置；如果有，则替换。

### 13、有看过concurrentHashMap的源码吗？它是如何保证线程安全的？

答：通过使用CAS算法保证线程安全。CAS即compare and swap，比较并交换。CAS主要利用V内存值、A旧值、B新值来实现，当A=V时，将v改成b。主要实现方法类：atomicInt、atomicLong.incrementAndGet。

## 四、多线程

### 1、对多线程的理解？多线程的作用是什么？多线程为什么要使用线程池（好处/优点）。在所作项目中什么场景会用到？

答：一个程序在一个时间段内同时执行多个任务，这就是多线程。作用是提高应用程序的工作效率。线程池的好处是减少了线程创建和销毁的次数，每个线程得以复用，可以调整线程池内的线程数量而不至于拖垮服务器。  
Niufind一级市场采集器的服务端用到了多线程，主要使用的是newCacheThreadPool这种类型的线程池，此线程池会回收长时间不工作的线程。

### 2、如何获取线程的返回值？

答：通过实现callable接口实现；ExecutorService.submit函数会返回一个Future 对象，可以调用Future.isDone函数Future是否完成。如果完成，接下来就可以调用Future.get函数获取线程返回值。

### 3、谈谈对乐观锁和悲观锁的理解

答：

乐观锁：别人在拿数据的时候，总是认为别人不会修改，所以不上锁，但是在更新的时候需要先判断一下在此期间别人有没有更新此数据，常用实现方法为版本号机制和CAS算法。乐观锁适用于多读场景，可以提高效率。  
悲观锁：别人在拿数据时，总是认为别人已经对他进行了修改，所以拿数据的时候先上锁，别人想拿数据就必须一直等待直到拿到锁为止。常用实现方法为数据库上锁（表锁、行锁、读写锁），synchronized和lock。多写场景下悲观锁比较适合。

### 4、谈一谈Lock锁，lock和synchronized的区别？

答：lock和synchronized一样都是用于线程之间的同步互斥。常用reentrantlock类实现lock，相比synchronized更加灵活。

Lock和synchronized区别：

①synchronized是java内置关键字；lock是一个类。

②synchronized无法判断是否获取锁的状态；lock可以获取。

③synchronized会自动释放锁（代码运行结束或者出现异常）；lock则需要手动释放（在finally里面unlock），否则会死锁。

④lock锁适合大量同步代码的同步问题，synchronized适合少量同步代码的同步问题。

⑤synchronized的锁可重入、不可中断、非公平；lock锁可重入、可中断、可公平。

⑥用synchronized关键字的两个线程1和线程2，如果当前线程1获得锁，线程2线程等待。如果线程1阻塞，线程2则会一直等待下去，而Lock锁就不一定会等待下去，如果尝试获取不到锁，线程可以不用一直等待就结束了；

### 5、线程池的作用。

答：线程池用于处理高并发的业务场景，比起传统多线程，线程池对线程进行了管理，减少线程创建和销毁的次数，线程得以复用，优化系统性能。

### 6、Java线程池有哪几种？各有什么特性？

答：共4种  
1.newCacheThreadPool:可缓存线程池，当有线程长时间不工作时，会释放他。  
2.newSingleExecutor：单线程化的Executor  
3.newFixedThreadPool:固定数量的线程池，即使有线程不工作，也不会释放他，所以会占用一些资源。  
4.newScheduleThreadPool:一个定长的线程池，可以定时和周期性的执行任务。

### 7、多个线程如何共享变量？

答：使用threadLocal。

### 8、wait()和sleep()的区别？

答：  
1.wait是obejct类的方法，sleep是Thread类的方法。  
2.sleep和wait都是暂停当前线程让出cpu的执行时间，但是sleep不会释放对象锁，等指定时间过了会继续执行，而wait则会释放对象锁资源，必须通过notify、notifyAll来手动唤醒。  
3.使用范围不同，wait必须在同步代码快内使用，sleep在任何地方都可以使用。

4.sleep必须try catch。

### 9、如何唤醒一个线程？

答：notify、notifyAll

### 10、怎么保证T1执行完后执行T2？

答：线程顺序调用join方法。

### 11、有用过CountdownLatch()方法吗？

答：CountDownLatch类位于java.util.concurrent包下，利用它可以实现类似计数器的功能。比如有一个任务A，它要等待其他4个任务执行完毕之后才能执行，此时就可以利用CountDownLatch来实现这种功能了。

### 13、悲观锁除了synchronized还有哪些？

答：lock锁

### 14、乐观锁的缺点？

答：如果写入操作较多的情况使用乐观锁，会导致系统效率降低，严重可能会影响数据正确性。

### 15、synchronized有哪几种使用方式？

答：修饰方法、修饰代码块、修饰静态方法、修饰一个类。

### 16、Volatile和synchronized的区别？

答：  
a. volatile不需要同步操作，所以效率更高，不会阻塞线程，但是适用情况比较窄

b. volatile读变量相当于加锁（即进入synchronized代码块），而写变量相当于解锁（退出synchronized代码块）

c. synchronized既能保证共享变量可见性，也可以保证锁内操作的原子性；volatile只能保证可见性

### 17、当线程池中的线程最大数为10，并发的请求数是100，那么有效的请求是多少？

答：都是有效的，只不过有执行先后顺序。线程池底层是一个阻塞队列。

### 18、ThreadLocal和Synchronized的区别？

答：  
1.synchronized关键字主要解决多线程共享数据同步问题。ThreadLocal使用场合主要解决多线程中数据因并发产生不一致问题。ThreadLocal和Synchonized都用于解决多线程并发访问。但是ThreadLocal与synchronized有本质的区别:  
1.synchronized是利用锁的机制，使变量或代码块在某一时该只能被一个线程访问。而ThreadLocal为每一个线程都提供了变量的副本，使得每个线程在某一时间访问到的并不是同一个对象，这样就隔离了多个线程对数据的数据共享。而Synchronized却正好相反，它用于在多个线程间通信时能够获得数据共享。  
2.Synchronized用于线程间的数据共享，而ThreadLocal则用于线程间的数据隔离。当然ThreadLocal并不能替代synchronized,它们处理不同的问题域。Synchronized用于实现同步机制，比ThreadLocal更加复杂。

### 线程的生命周期？

答：①新建状态（创建实例） ②就绪状态（调用start方法） ③运行状态（当线程获得CPU时间后，它才进入运行状态，真正开始执行run()方法）

④阻塞状态（处于运行状态中的线程由于某种原因，暂时放弃对CPU的使用权，停止执行，此时进入阻塞状态，直到其进入到就绪状态，才 有机会再次被CPU调用以进入到运行状态），主要有3中类型的阻塞：等待阻塞wait、同步阻塞synchronized、其他阻塞sleep ⑤死亡状态（run方法异常结束或线程执行完毕）

### 实现线程安全的方式有哪些？实现线程同步的方式有哪些？

答：线程安全有3种：

①同步代码块synchronized ②同步方法synchronized ③Lock锁机制

线程同步有7种：

①同步方法synchronized ②同步代码块synchronized ③volatile变量实现同步（修饰词）

④可重入锁concurrent包下的ReentrantLock ⑤线程变量threadLocal ⑥阻塞队列LinkedBlockingQueue ⑦原子变量atomic的CAS算法实现

### 21、乐观锁是怎么实现的？谈谈CAS

答：乐观锁主要通过数据库version机制和CAS算法实现。  
JDK1.5之前都是使用synchronized来实现同步，这样就会产生锁，坏处就是一个线程有锁，其他线程需要此锁都得挂起，多线程下不停加锁释放锁会引起较多的上下文切换和调度延时，引起性能问题。CAS则是一种无锁化的算法，在concurrent包下，他具有3个操作数，V内存值，B要修改的新值，A旧的预期值，当且仅当预期值A和内存值V相同时，将内存值V修改为B，否则什么都不做。

## 五、JVM

### 1、描述JVM类加载机制（过程）。

答：一个.java文件在编译后会形成相应的一个或多个Class文件（若一个类中含有内部类，则编译后会产生多个Class文件），但这些Class文件中描述的各种信息，最终都需要加载到虚拟机中之后才能被运行和使用。事实上，虚拟机把描述类的数据从Class文件加载到内存，并对数据进行校验，转换解析和初始化，最终形成可以被虚拟机直接使用的Java类型的过程就是虚拟机的 类加载机制。

过程：

①加载（Loading）：在加载阶段（可以参考java.lang.ClassLoader的loadClass()方法），虚拟机需要完成以下三件事情：

　　(1). 通过一个类的全限定名来获取定义此类的二进制字节流（并没有指明要从一个Class文件中获取，可以从其他渠道，譬如：网络、动态生成、数据库等）；

　　(2). 将这个字节流所代表的静态存储结构转化为方法区的运行时数据结构；

　　(3). 在内存中(对于HotSpot虚拟就而言就是方法区)生成一个代表这个类的java.lang.Class对象，作为方法区这个类的各种数据的访问入口；

　　加载阶段和连接阶段（Linking）的部分内容（如一部分字节码文件格式验证动作）是交叉进行的，加载阶段尚未完成，连接阶段可能已经开始，但这些夹在加载阶段之中进行的动作，仍然属于连接阶段的内容，这两个阶段的开始时间仍然保持着固定的先后顺序。

② 验证（Verification）：验证是连接阶段的第一步，这一阶段的目的是为了确保Class文件的字节流中包含的信息符合当前虚拟机的要求，并且不会危害虚拟机自身的安全。 验证阶段大致会完成4个阶段的检验动作：

(1)文件格式验证：验证字节流是否符合Class文件格式的规范(例如，是否以魔术0xCAFEBABE开头、主次版本号是否在当前虚拟机的处理范围之内、常量池中的常量是否有不被支持的类型)

(2)元数据验证：对字节码描述的信息进行语义分析，以保证其描述的信息符合Java语言规范的要求(例如：这个类是否有父类，除了java.lang.Object之外)；

(3)字节码验证：通过数据流和控制流分析，确定程序语义是合法的、符合逻辑的;

(4)符号引用验证：确保解析动作能正确执行。

　　验证阶段是非常重要的，但不是必须的，它对程序运行期没有影响。如果所引用的类经过反复验证，那么可以考虑采用-Xverifynone参数来关闭大部分的类验证措施，以缩短虚拟机类加载的时间。

③准备（Preparation）：准备阶段是正式为类变量(static 成员变量)分配内存并设置类变量初始值（零值）的阶段，这些变量所使用的内存都将在方法区中进行分配。

④：解析（Resolution）：解析阶段是虚拟机将常量池内的符号引用替换为直接引用的过程。解析动作主要针对类或接口、字段、类方法、接口方法、方法类型、方法句柄和调用点限定符7类符号引用进行。

⑤初始化（Initialization）：类初始化阶段是类加载过程的最后一步。在前面的类加载过程中，除了在加载阶段用户应用程序可以通过自定义类加载器参与之外，其余动作完全由虚拟机主导和控制。到了初始化阶段，才真正开始执行类中定义的java程序代码(字节码)。初始化阶段是执行类构造器<clinit>()方法的过程。

### 2、JVM原理

答：JVM是java的核心和基础，在java编译器和os平台之间的虚拟处理器。它是一种利用软件方法实现的抽象的计算机基于下层的操作系统和硬件平台，可以在上面执行java的字节码程序。

java编译器只要面向JVM，生成JVM能理解的代码或字节码文件。Java源文件经编译成字节码程序，通过JVM将每一条指令翻译成不同平台机器码，通过特定平台运行。

### 3、垃圾回收原理

答：垃圾回收就是通过遍历托管堆中的对象，发现无用信息对象。然后回收被无用对象占用的内存空间，使该空间可被程序再次使用。

### 4、谈谈JVM的新生代、老年代、永久代。

答：

新生代：所有新生成的对象首先都是放在新生代的。新生代的目标就是尽可能快速的收集掉那些生命周期短的对象。

老年代：在年轻代中经历了N次垃圾回收后仍然存活的对象，就会被放到年老代中。因此，可以认为年老代中存放的都是一些生命周期较长的对象。内存比新生代也大很多(大概比例是1:2)，当老年代内存满时触发Major GC即Full GC，Full GC发生频率比较低，老年代对象存活时间比较长，存活率标记高。

永久带：用于存放静态文件，如Java类、方法等。持久代对垃圾回收没有显著影响，但是有些应用可能动态生成或者调用一些class，例如Hibernate 等，在这种时候需要设置一个比较大的持久代空间来存放这些运行过程中新增的类。

### 5、谈谈JVM运行数据区？

答：jvm运行数据区只要包括5个模块，

①程序计数器：是一块较小的内存空间，它可以看做是当前线程所执行的字节码的行号指示器，所以是每个线程私有的，它的生命周期随线程启动还产生，随线程结束而消亡。

②虚拟机栈：虚拟机栈也是线程私有的，虚拟机栈描述的是Java的方法执行的内存模型，每个方法在执行的同时都会创建一个[栈桢](http://blog.csdn.net/qq_25253287/article/details/51626374" \t "https://blog.csdn.net/fanxing1964/article/details/_blank)用于存储局部变量表，操作数栈，动态链接，方法出口等信息。

③本地方法栈：本地方法栈和虚拟机栈相似，区别就是虚拟机为虚拟机栈执行Java服务（字节码服务），而本地方法栈为虚拟机使用到的Native方法服务。本地方法栈中使用的语言，使用方式，数据结构没有强制要求。

④堆：jvm内最大的内存区域，所有线程共享。在虚拟机启动时创建，用于存放对象实例和数组，堆是垃圾收集管理的主要区域，因为垃圾收集器是分代收集，所以又分新生代、老年代和永久带。无论怎么分都是为了更好的分配、回收、利用内存

⑤方法区：也是一块线程共享的区域，用于存储已被虚拟机加载的类信息，常量、静态变量、JIT编译后的代码等数据。

### stack堆栈的使用？

答：一种后进先出的模式。

栈（stack）又名堆栈，它是一种运算受限的线性表。其限制是仅允许在表的一端进行插入和删除运算。这一端被称为栈顶，相对地，把另一端称为栈底。向一个栈插入新元素又称作进栈、入栈或压栈，它是把新元素放到栈顶元素的上面，使之成为新的栈顶元素；从一个栈删除元素又称作出栈或退栈，它是把栈顶元素删除掉，使其相邻的元素成为新的栈顶元素。

### 什么叫内存泄漏（memory leak）、内存溢出（out of memory）？

答：内存泄漏是指程序向系统申请分配内存进行使用(new)，可是使用完了以后却不归还(delete)，结果你申请到的那块内存你自己也不能再访问（也许你把它的地址给弄丢了），而系统也不能再次将它分配给需要的程序，这就导致无论多少内存,迟早会被占光，最终导致内存溢出。

内存泄漏指程序在申请内存后，无法释放已申请的内存空间，一次内存泄露危害可以忽略，但内存泄露堆积后果很严重；

以发生的方式来分类，内存泄漏可以分为4类：

1. 常发性内存泄漏；

2. 偶发性内存泄漏；

3. 一次性内存泄漏；

4. 隐式内存泄漏： 程序在运行过程中不停的分配内存，但是直到结束的时候才释放内存。但是对于一个服务器程序，需要运行几天，几周甚至几个月，不及时释放内存也可能导致最终耗尽系统的所有内存。

从这个角度来说，一次性内存泄漏并没有什么危害，因为它不会堆积，而隐式内存泄漏危害性则非常大，因为较之于常发性和偶发性内存泄漏它更难被检测到；

### 8、JVM有哪几种GC回收器？

答：

新生代收集器使用的收集器：  
①Serial（复制算法）：单线程，默认新生代收集器，简单高效。  
②PraNew（停止-复制算法）：serial的多线程版本  
③Parallel Scavenge（停止-复制算法）：并行多线程，吞吐量优先收集器。  
老年代收集器使用的收集器：  
④Serial Old（标记-整理算法）：serial收集器的老年代版本，单线程  
⑤Parallel Old（停止-复制算法）：Parallel Scavenge的老年代版本，多线程  
⑥CMS（标记-清理算法）：它是一种以获取最短回收停顿时间为目标的收集器。并发收集，低停顿

⑦G1（Garbage First）：区域划分、有优先级的区域回收，保证了G1收集器在有限的时间内可以获得最高的收集效率。

## 六、MySQL

### 1、MySQL常用存储引擎有哪些，都分别有哪些特性。

答：  
1.MyISAM:不支持事务：MyISAM存储引擎不支持事务，所以对事务有要求的业务场景不能使用表级锁定：其锁定机制是表级索引，这虽然可以让锁定的实现成本很小但是也同时大大降低了其并发性能；读写互相阻塞：不仅会在写入的时候阻塞读取，MyISAM还会在读取的时候阻塞写入，但读本身并不会阻塞另外的读；只会缓存索引：MyISAM可以通过key\_buffer缓存以大大提高访问性能减少磁盘IO，但是这个缓存区只会缓存索引，而不会缓存数据  
2.InnoDB:具有较好的事务支持：支持4个事务隔离级别，支持多版本读；行级锁定：通过索引实现，全表扫描仍然会是表锁，注意间隙锁的影响；读写阻塞与事务隔离级别相关；具有非常高效的缓存特性：能缓存索引，也能缓存数据；整个表和主键以Cluster方式存储；组成一颗平衡树，所有Secondary Index都会保存主键信息  
3.Memory:1、不支持事务、不支持外键   
2、将数据存储在内存中，访问速度非常快  
3、支持HASH和BTREE索引，默认使用hash索引  
4、每个memory表只对应一个格式是.frm的磁盘文件

### 2、InnoDB是行锁还是页锁？

答：行锁

### 3、谈一下以往工作中对SQL优化的策略措施。

答：①不使用select \*这种查询方式，写出具体查询字段 ②对order by和where后面的字段适当加上索引 ③避免在where后面进行is null的判断、使用!=、<>等操作符 ④带通配符的like语句，字段前不要加上通配符 ⑤避免在where后面加or连接条件，可以考虑使用union all ⑥避免在where后面进行函数操作 ⑦一个表的索引不能建立太多 ⑧设置合适的字段属性和长度

### 4、查询出各班成绩前五名的学生（MySQL）。

答：cid：班级id，score得分

select a.\*

from 成绩表 a

where (select count(\*) from 成绩表 where cid=a.cid and a.score<score)<5

order by a.cid,a.score desc;

### 5、Having的作用和用法?

答：having 相当于where，与where的唯一区别是 当查询语句中有 聚合函数 的时候 就不能用where 了 只能用having；一般跟在group by后使用。

### 6、索引的原理

答：数据库表实质上是一个平衡树结构的数据结构。一般通过主键索引（聚合索引）定位到数据，根据树的分支则可以大大提高查询效率，快速定位到id值对应的数据。非聚合索引是一个独立的索引结构，每个结构互不关联，每新建一个非聚集索引，就会在磁盘里创建相关的数据，因此会加大表的体积，占用磁盘存储空间；非聚集索引在查询数据时会先查询到相关的id值，再通过聚合索引快速定位到数据。

### 7、为什么有主键索引却还有唯一索引，两者的区别是什么？

答：  
1  主键是一种约束，唯一索引是一种索引，两者在本质上是不同的。  
2  主键创建后一定包含一个唯一性索引，唯一性索引不一定就是主键。  
3   唯一性索引列允许空值， 而主键列不允许为空值。  
4   主键可以被其他表引用为外键，而唯一索引不能。  
5   一个表最多只能创建一个主键，但是可以创建多个唯一索引。  
6   主键更适合那些不容易改变的唯一标识，如自动递增列，身份证号等。  
7   在RBO 模式下，主键的执行计划优先级高于唯一索引。两者可以提高查询的速度。

### 8、索引的数据结构了解吗？

答：平衡树结构。

### 10、谈谈联合索引，如何实现？

答：where多条件联合查询时最好建联合索引,因为多个单列索引在多条件查询时只会生效第一个索引！创建复合索引时，应该仔细考虑复合索引列的顺序，要根据业务需求，where子句中使用最频繁的一列放在索引的最左边；

如：联合索引：userId, mobile, billMonth

userid 经常需要作为查询条件，而 mobile 不常常用，则需要把 userid 放在联合索引的第一位置，即最左边

### 11、MySQL默认的隔离级别？

答：  
（1）read uncommitted 未提交读

所有事务都可以看到没有提交事务的数据。

（2）read committed 提交读

事务成功提交后才可以被查询到。

（3）repeatable 重复读

同一个事务多个实例读取数据时，可能将未提交的记录查询出来，而出现幻读。mysql默认级别

（4）Serializable可串行化

强制的进行排序，在每个读读数据行上添加共享锁。会导致大量超时现象和锁竞争。

### 13、MySQL怎么实现全文检索查询。

答：主要通过加全文索引，match...against来实现。

### 14、事务的四大基本要素是什么？

答：  
原子性：整个事物中的所有操作，要么全部成功，如果遇到问题，则会进行回滚。

隔离性：一个事物在对一条数据进行操作时，其他事物不能进行。

持久性：事物完成后，指数据能进行持久化保存。

一致性：事物在完成前和完成后，其数据完整性并没有被破坏。

### 15、数据库那些地方需要优化

答：（1）查看sql语句的执行计划是否正常：注意SQL的优化包含哪些？

（2）减少应用和数据库的交互次数、以及同一个sql语句的执行次数；

（3）对访问频繁的数据，尽可能使用项目的应用缓存，数据库索引、以及冷热数据分离，分库分表等方式；

### 16、Mysql锁有哪些？发生死锁怎么解决？

答：MyISAm和Memory是表锁、Innodb行锁、BDB页面锁

MyISAM表锁：表共享读锁、表独占写锁。都是直接加锁。

InnoDB行锁：共享锁（读锁，LOCK IN SHARE MODE）、排它锁（写锁，FOR UPDATE）、意向共享锁（表级锁）、意向排它锁（表级锁）

死锁解决：死锁是指2个以上进程争夺资源导致相互等待的情况，无外力干扰，他们会一直相互等待下去。

解决死锁的基本方法：①预防死锁 ②避免死锁 ③检测死锁 ④解除死锁（剥夺资源、撤销进程、重启）

### 17、MySQL索引哪些情况下会失效？

答：①like 通配符的设置 ②where后面跟<>、!=等符号 ③where后面跟计算，函数，（自动或者手动）类型装换 ④is null和is not null判断也会引起失效 ⑤字符串不加引号

### 什么是存储过程？优缺点？和函数比较

答：https://www.seoxiehui.cn/article-52344-1.html

### 19、union和union all的区别？

答：

Union：对两个结果集进行并集操作，不包括重复行，同时进行默认规则的排序；

Union All：对两个结果集进行并集操作，包括重复行，不进行排序；

UNION ALL 要比UNION快很多，因为UNION ALL不去重，且不排序；

使用union和union all必须保证各个select 集合的结果有相同个数的列，并且每个列的类型是一样的。

### 20、对于MySQL你是怎么去进行SQL优化的？选用什么工具？

答：定位到查询速度较慢的sql语句，对它进行优化。可以使用慢查询工具profiling.。

## 七、Dubbo

### 1、Dubbo是怎么进行监控的？

### 2、描述一下Dubbo的实现原理？

## 八、Tomcat

### 1、Tomcat启动后，Web项目的加载顺序？

答：web项目运行时首先会加载web.xml配置文件，web.xml中的内容加载不依赖于内容书写顺序，而是以 context-param 、 listener、 filter、servlet的顺序进行加载。

### 2、Tomcat优化

答：  
①tomcat内存优化，通过更改catalina.sh改变内存值。  
②tomcat线程优化，在server.xml中更改  
③tomcat的IO优化，同在server.xml中更改，可以切换AIO（异步非阻塞）、BIO（同步阻塞）、NIO（异步阻塞）三种方式

④使用APR，从操作系统级别来解决异步的IO问题,大幅度的提高性能

## 九、其他

### 1、java常用数据类型有哪些？

答：byte、short、long、int、double、float、char、boolean

### 讲一下java的反射？

答：指在运行状态中,对于任意一个类,都能够知道这个类的所有属性和方法,对于任意一个对象,都能调用它的任意一个方法.这种动态获取信息,以及动态调用对象方法的功能叫java语言的反射机制。  
只要给定类的名字，就可以通过反射机制来获取类的所有信息，可以动态的创建对象和编译。原理：JAVA语言编译之后会生成一个.class文件，反射就是通过字节码文件找到某一个类、类中的方法以及属性等。反射的实现主要借助以下四个类：

Class：类的对象

Constructor：类的构造方法

Field：类中的属性对象  
Method：类中的方法对象

### 3、前后端是通过什么协议通信的？HTTP或Socket

答：http。

### 有哪些常见的io流？

答：

字节流：inputStream（FileInputStream）、outputStream（FileOutPutStream）

缓冲字节流：BufferedInputStream、BufferedOutputStream

字符流：Reader（FileReader）、Writer（FileWriter）

缓冲字符流：BufferedReader、BufferedWriter

转换流：InputStreamReader、OutputStreamWriter

### 常用设计模式有哪些？

答：

①单例设计模式：一个类只能允许一个实例；让类的构造方法私有化，同时提供一个静态方法创建实例。一般分为懒汉式（在静态方法中初始化）和饿汉式（在声明对象就初始化），饿汉式用的较多。

②简单工厂设计模式：又称静态工厂，由工厂对象决定创建哪个产品对象。就是写一个类，让他创建我们想要的对象。

③适配器设计模式：某类继承这个适配器，从而实现我们需要实现的方法。通过写一个适配器类，里面写了所有的抽象方法，但是这些方法是空的，并且适配器类要定义成抽象的，如果适配器类可以自己实现就没有意义了。适配器的作用，继承适配器，重新需要重新的方法就可以了，简化操作。

④模板设计模式：定义一个算法骨架将具体实现交给子类去实现。在类中定义一个抽象方法，距离实现交由子类去实现

### 谈谈静态代理、JDK动态代理、CGlib代理

答：

静态代理：1.目标对象必须要实现接口

2.代理对象要实现与目标对象一样的接口

JDK动态代理：1、通过jdk的api在运行时期动态的生成代理对象的!

2.目标对象一定要实现接口,代理对象不用实现接口!

CGlib代理：1.目标对象可以不实现接口

2.目标类不能为final，如果为final报错

3、在运行时期动态在内存中构建一个子类对象,从而对目标对象扩展

### 7、ibatis和hibernate的区别？

答：  
①ibatis是半自动化的orm框架，主要是pojo和sql之间的影射，所以需要手写大部分sql；而hibernate是全自动的orm框架，主要是pojo和表之间的映射，会自动生成sql语句。  
②当系统属于二次开发，无法对数据库结构做到控制和修改，那iBATIS的灵活性将比Hibernate更适合。  
③ 系统数据处理量巨大，性能要求极为苛刻，这往往意味着我们必须通过经过高度优化的SQL语句（或存储过程）才能达到系统性能设计指标。在这种情况下iBATIS会有更好的可控性和表现。iBatis比Hibernate更容易进行sql的优化。鉴于一般系统性能的瓶颈都在数据库上，所以这一点是iBatis非常重要的一个优势。  
④ iBatis 可以进行细粒度的优化  
⑤iBatis需要手写sql语句，也可以生成一部分，Hibernate则基本上可以自动生成，偶尔会写一些Hql。同样的需求，iBATIS的工作量比Hibernate要大很多。类似的，如果涉及到数据库字段的修改，Hibernate修改的地方很少，而iBATIS要把那些sql mapping的地方一一修改。  
⑥开发的可维护性方面，我觉得iBatis更好一些。因为iBatis的sql都保存到单独的文件中。而Hibernate在有些情况下可能会在java代码中保存sql/hql

### 8、抽象类和接口的区别？

答：

### 9、什么是第三方支付？

答：主要是微信支付和支付宝支付等。

### 10、String、StringBuilder、StringBuffer的区别？

答：①String是不可变量，一旦创建不能修改，所谓的修改都是新建一个对象，然后把新值赋值进去，string为final类不可继承。运行速度最慢。

②Stringbuffer则是可变对象，他只能通过构造函数创建，通过append方法来改变值，StringBuffer是带有缓冲的，执行效率更高些。stringBuffer是线程安全的，内部有很多synchronized方法来实现同步。

③StringBulder是字符串可变变量。他不是线程安全的，所以他的运行速度三者之中最快。

String适用于少量字符串操作的场景；

Stringbuilder适用于单线程下在字符缓冲区大量操作的场景；

Stringbuffer适用于多线程下在字符缓冲区大量操作的场景。

### 11、冒泡排序算法的实现

答：

N个数字要排序完成，总共进行N-1趟排序，每i趟的排序次数为(N-i)次，所以可以用双重循环语句，外层控制循环多少趟，内层控制每一趟的循环次数，即

for(int i=1;i<arr.length;i++){

for(int j=1;j<arr.length-i;j++){

//交换位置

}

}

### 14、HTTP和HTTPS的区别？

答：①https需要ca证书  
②http是超文本传输协议，明文传输；https是基于ssl协议加密传输，安全性更高。  
③两者连接方式不同，端口号也不同。

### 15、怎么防范redis缓存穿透

答：出现redis缓存穿透的原因：  
1、恶意攻击，猜测你的key命名方式，然后估计使用一个你缓存中不会有的key进行访问。

2、第一次数据访问，这时缓存中还没有数据，则并发场景下，所有的请求都会压到数据库。

3、数据库的数据也是空，这样即使访问了数据库，也是获取不到数据，那么缓存中肯定也没有对应的数据。这样也会导致穿透。

解决方式：

1.可以提前将可能产生大量并发的数据写入缓存，避免刚启动时产生大量请求压倒数据库。  
2.规范key的命名，在入口处对key进行检验，保证恶意的key被拦截。  
3.Synchronized双重检测机制，这时我们就需要使用同步（Synchronized）机制，在同步代码块前查询一下缓存是否存在对应的key，然后同步代码块里面再次查询缓存里是否有要查询的key。 这样“双重检测”的目的，还是避免并发场景下导致的没有意义的数据库的访问  
4.不管数据库中是否有数据，都在缓存中保存对应的key，值为空就行。–这样是为了避免数据库中没有这个数据，导致的平凡穿透缓存对数据库进行访问。

### 16、Redis用过哪些数据结构？

答：String（字符串）、list（列表）、set（集合）、hash（哈希）、zset（有序集合）  
用过string（get，del，set）、list（lpush，lrange，lindex，lpop）、hash（hset，hget，hdel，hgetall）

### 数据库的数据存储在哪儿？

答：通过查看mysql目录下的mysql.ini文件查看datadir。Mysql是物理存储在硬盘内。

### 17、了解jar包热加载及隔离技术吗？

答：

热加载：自定义一个加载器，继承URLClassLoader，重写相关的方法即可。

热隔离：自定义 ClassLoader，使其 Parent = null，避免其使用系统自带的 ClassLoader 加载 Class。

在调用相应版本的方法前，更改当前线程的 ContextClassLoader，避免扩展包的依赖包通过Thread.currentThread().getContextClassLoader()获取到非自定义的 ClassLoader 进行类加载

通过反射获取 Method 时，如果参数为自定义的类型，一定要使用自定义的 ClassLoader 加载参数获取 Class，然后在获取 Method，同时参数也必须转化为使用自定义的 ClassLoade 加载的类型（不同 ClassLoader 加载的同一个类不相等）

### 18、谈谈你对分布式的理解？

答：https://blog.csdn.net/u012547613/article/details/79283196

### 19、例举Linux的常用命令及功能？

答：ls、ll：展示当前目录内容

cd：进入文件夹

Cat：显示文件内容

mv：移动文件

Cp：复制文件

Rm：删除文件

Vim：编辑文件

Grep：查找

Kill -9 进程号：杀进程

Ifconfig、tail -f

### 20、BIO、NIO、AIO的区别？

答：

BIO：同步并阻塞，实现模式为一个连接一个线程，即客户端有连接请求时服务器端就需要启动一个线程进行处理，如果这个连接不做任何事情会造成不必要的线程开销，当然可以通过线程池机制改善。

NIO： 同步非阻塞，服务器实现模式为一个请求一个线程，即客户端发送的连接请求都会  
册到多路复用器上，多路复用器轮询到连接有I/O请求时才启动一个线程进行处理。

AIO： 异步非阻塞，服务器实现模式为一个有效请求一个线程，客户端的I/O请求都是由OS先完成了再通知服务器应用去启动线程进行处理

### 21、Redis为什么采用单线程？

答：redis是内存数据库，在效率上单线程比多线程效率要高。单线程还有以下优点，

①代码更清晰，处理逻辑简单

②不需要考虑锁的问题

③不存在多线程之间互相切换造成的CPU损耗。

### 22、容器是什么结构（如Spring和Tomcat容器）？

答：spring容器是整个spring框架的核心,通常我们说的spring容器就是bean工厂,bean工厂负责创建和初始化bean、装配bean并且管理应用程序中的bean.spring中提供了两个核心接口：BeanFactory和ApplicationContext,ApplicationContext是BeanFactory子接口,它提供了比BeanFactory更完善的功能。

Tomcat容器是一种serlvet容器，它可以帮我们对接http请求（做些通用处理），然后将请求转发到我们的servlet处理器进行处理，我们只需要把自己的业务处理放在servlet的service方法即可，不需要关注其他多余的事情。

### 23、Linux支持的最大线程数是多少？

答：1024，通过命令ulimit -u查看。

### 24、分布式锁的实现方式有哪几种？如何实现？

答：

①基于数据库实现分布式锁 ：基于数据库表来实现，插入一条数据上锁，释放锁则删除该数据。使用行锁（排他锁）select...for update。

②基于缓存（redis，memcached，tair）实现分布式锁

③基于Zookeeper实现分布式锁

### 25、Redis有几种部署方式？各有什么优缺点？

答：

①单节点部署：优点是结构简单，部署方便，性能也比较高。缺点是不能保证数据可靠性。

②主从模式：优点是相比单节点可以主从实例间实现数据同步，此模式还提供了数据持久化和备份策略，可靠性较高。还可以配置读写分离策略，有效应对高并发场景。缺点是出现故障后恢复比较复杂，主库的写、存储能力都受到单机的限制。

③哨兵模式：优点是集群部署简单，能解决主从模式下的高可用切换问题；可以实现一套sentinel监控一组redis数据节点或多组数据节点。缺点部署较主从复杂一点，原理理解繁琐；浪费资源，redis数据节点中slave节点作为备份节点不提供服务；不能解决读写分离问题，实现起来很复杂。

④分布式集群部署：优点是无中心架构；数据按照slot存储分布在多个节点，节点之间数据共享，可动态调整；可扩展性和可用性高；还可以降低运维成本。缺点是client实现复杂；节点发生阻塞会被判定为下线；数据通过异步复制，不能保证数据的强一致性；不支持多数据空间，只能使用db0。

### redis同步锁实现原理？

答：实现同步锁原理

1）加锁：“锁”就是一个存储在redis里的key-value对，只要这个唯一的key-value存在，就表示这个操作已经上锁。

2）解锁：既然key-value对存在就表示上锁，那么释放锁就自然是在redis里删除key-value对。

3）阻塞、非阻塞：阻塞式的实现，若线程发现已经上锁，会在特定时间内轮询锁。非阻塞式的实现，若发现线程已经上锁，则直接返回

4）处理异常情况：假设当投资操作调用其他平台接口出现等待时，自然没有释放锁，这种情况下加入锁超时机制，用redis的expire命令为key设置超时时长，过了超时时间redis就会将这个key自动删除，即强制释放锁。

### 26、redis分片知道吗？

答：分片(partitioning)就是将你的数据拆分到多个 Redis 实例的过程，这样每个实例将只包含所有键的子集。如果只使用一个redis实例时，其中保存了服务器中全部的缓存数据，这样会有很大风险，如果单台redis服务宕机了将会影响到整个服务。解决的方法就是我们可以采用分片/分区的技术，将原来一台服务器维护的整个缓存，现在换为由多台服务器共同维护内存空间。  
分片的方式：

一是范围分片

二是哈希分片：取模运算  
redis集群也是最常用的分片方式。

### 27、Netty与Java NIO的对比？

答：netty是基于NIO的一个CS框架。Netty提供异步的、[事件驱动](https://baike.baidu.com/item/%E4%BA%8B%E4%BB%B6%E9%A9%B1%E5%8A%A8" \t "https://blog.csdn.net/qq285016127/article/details/_blank)的网络应用程序框架和工具，用以快速开发高性能、高可靠性的[网络服务器](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E6%9C%8D%E5%8A%A1%E5%99%A8" \t "https://blog.csdn.net/qq285016127/article/details/_blank)和客户端程序

Netty相比NIO，他的API使用简单，开发门槛低。

### 28、现在正在看的有哪几本书？

答：spring源码解析

### 29、用过的Java构建工具有哪些？Gradle是什么语言开发的？

答：maven、ant、gradle；java和Groovy语言

### 30、通过RMI接口作为分布式请求调用转发，用的是什么协议？TCP、UDP、Socket还是HTTP？

答：tcp

### 31、JDK1.7在JDK1.6基础上有哪些不同？

答：

### 32、Redis有哪几种持久化方式？各有什么特性，项目中运用到哪种持久化方式？

答：由于Redis的数据都存放在内存中，如果没有配置持久化，redis重启后数据就全丢失了，于是需要开启redis的持久化功能，将数据保存到磁盘上，当redis重启后，可以从磁盘中恢复数据。

方式一：是RDB持久化（原理是将Reids在内存中，在指定的时间间隔能对内存中的数据集进行快照存储）

方式二：是AOF（append only file）持久化（原理是将Reids的操作日志以追加的方式写入文件，以日志的形式记录服务器所处理的每一个写、删除操作，查询操作不会记录，以文本的方式记录）

### 你作为架构师的话，会考虑如何架构设计一个系统？

## 项目相关

### socket和udp具体如何实现的？

答：项目中本socket主要作为客户端，以设备重启功能为例，通过使用DataOutputStream写入约定好的重启标识字符串然后传输给服务器端，服务端根据标识调用重启脚本。

Udp在此项目中主要作为服务器端，通过使用监听器，在后台开启一个线程持续监听其他客户端传输过来的日志信息，解析后入库。

### vue相关

答：常用标签v-for、v-if、v-on；

①npm install命令安装依赖包，

②npm run dev 打包

组件化开发步骤：

①引入组件②注册组件，components:{FooterNav}

③使用组件，直接通过<footer-nav></footer-nav>引入

### 3、python相关

答：

### Nginx部署

答：①下载完通过tar命令解压

②编译安装

③配置多个转发路径（ip+端口，weight权重，ip\_hash解决session问题）

④重启Nginx服务

### 5、MongoDB相关