1. **object 类有哪些方法？**

1，构造函数   
2，hashCode和equale函数用来判断对象是否相同,   
3，wait(),wait(long),wait(long,int),notify(),notifyAll()   
4，toString()和getClass,   
5，clone()   
6，finalize()用于在垃圾回收

1. **String类能否被继承？**

不可以，因为String类有final修饰符，而final修饰的类是不能被继承的，实现细节不允许改变。

public final class String implements java.io.Serializable, Comparable<String>, CharSequence

1. **Lru算法以及它的java实现。**

博客0章节第21篇。

1. **说一说syncronized和reenterlock的区别。**

博客4章节第13篇。

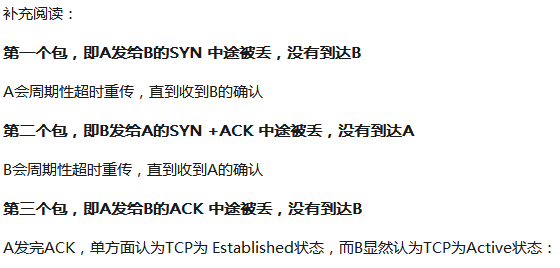
1. **tcp三次握手过程？为什么一定要三次握手？**

这道题我们可以从三次握手的过程来回答，把三次握手的过程说出来。

首先是客户端SYN,之后是服务器这边的SYN+ACK,最后一次是客户端的ACK，如果少了客户端的ACK，有可能服务器这边的SYN+ACK第二次就建立连接，那么客户端收没收到服务器的信号都不知道，**会无法让双方都证实对方能发收。**

之所以存在 3-way hanshake 的说法，是因为 TCP 是双向通讯协议，作为响应一方(Responder) 要想初始化发送通道，**必须也进行一轮 SYN + ACK。**

由于 SYN ACK 在 TCP 分组头部是两个标识位，因此处于优化目的被合并了。所以达到双方都能进行收发的状态只需要 3 个分组。

实际上理解成两次（单向通讯）和四次（不考虑合并）也未尝不可。

1. **如果只有两次握手，当某个请求因为网络不稳定而丢失，最后又收到了这个请求，会怎么样处理？**

1.假设改为两次握手，client端发送的一个连接请求在服务器滞留了，这个连接请求是无效的，client已经是closed的状态了，而服务器认为client想要建立

一个新的连接，于是向client发送确认报文段，**而client端是closed状态**，无论收到什么报文都会丢弃。而如果是两次握手的话，此时就已经建立连接了。

**服务器此时会一直等到client端发来数据，这样就浪费掉很多server端的资源。**

2.有可能服务器这边的SYN+ACK第二次就建立连接，**那么客户端收没收到服务器的信号都不知道。**

1. **get和post请求有什么区别，post请求能否被缓存？**

**1.首先是区别。**



**2.就是get和post区别的缓存问题。**

首先要了解什么是缓存。

**HTTP缓存**的基本目的就是使应用执行的更快，更易扩展，但是HTTP缓存通常只适用于idempotent request（**可以理解为查询请求，也就是不更新服务端数据的请求**），这也就导致了在HTTP的世界里，一般都是对Get请求做缓存，Post请求很少有缓存。

get多用来直接获取数据，不修改数据，主要目的就是DB的search语句的感觉。**用缓存(有个代理服务器的概念)的目的就是查db的速度变快。**

post则是发送数据到服务器端去存储。类似db里的update delete和insert语句的感觉。更新db的意思。数据必须放在数据库，所以一般都得去访问服务器端。

**3.安全问题。**

get到服务器过程中数据都是在url中，也就是说要传送的数据是可以在链接里面看到，就有安全问题。因为是一个url，所以就跟百度网址一样，好理解啦吧。

post就不是在url里面所有还是比较安全的。

1. **说说你对RESTful的理解。**

RESTful是符合REST原则的表现形式。Rest架构的主要原则：

1. 网络上的所有事物都被抽象为资源
2. 每个资源都有一个唯一的资源标识符
3. 同一个资源具有多种表现形式(xml,json等)
4. 对资源的各种操作不会改变资源标识符
5. 所有的操作都是无状态的
6. **https请求具体过程。ssl四次握手。**

|  |
| --- |
| 我们都知道HTTPS能够加密信息，以免敏感信息被第三方获取。所以很多银行网站或电子邮箱等等安全级别较高的服务都会采用HTTPS协议。  **HTTPS简介**  HTTPS其实是有两部分组成：HTTP + SSL / TLS，也就是在HTTP上又加了一层处理加密信息的模块。**服务端和客户端的信息传输都会通过TLS进行加密，所以传输的数据都是加密后的数据。**具体是如何进行加密，解密，验证的，且看下图。  http://cdn.liqwei.com/upload/www/201211/20121130144420002.png  **1. 客户端发起HTTPS请求**  这个没什么好说的，就是用户在浏览器里输入一个https网址，然后连接到server的443端口。，把自身支持的一系列Cipher Suite（密钥算法套件，简称Cipher）发送给服务端  **2. 服务端的配置**  采用**HTTPS协议的服务器必须要有一套数字证书**，可以自己制作，也可以向组织申请。区别就是自己颁发的证书**需要客户端验证通过**，才可以继续访问，而使用受信任的公司申请的证书则不会弹出提示页面(startssl就是个不错的选择，有1年的免费服务)。**这套证书其实就是一对公钥和私钥。**  **3. 传送证书**  **这个证书其实就是公钥，只是包含了很多信息，如证书的颁发机构，过期时间等等。**  **4. 客户端解析证书**  这部分工作是有客户端的TLS来完成的，首先**会验证公钥是否有效，**比如颁发机构，过期时间等等，如果发现异常，则会弹出一个警告框，提示证书存在问题。如果**证书没有问题，那么就生成一个随机值。然后用证书对该随机值进行加密。**就好像上面说的，把随机值用锁头锁起来，这样除非有钥匙，不然看不到被锁住的内容。  **5. 传送加密信息**  这部分**传送的是用证书加密后的随机值，目的就是让服务端得到这个随机值，**以后**客户端和服务端的通信就可以通过这个随机值来进行加密解密了。**  **6. 服务段解密信息**  服务端用**私钥解密后，得到了客户端传过来的随机值(私钥)，**然后**把内容通过该值进行对称加密。所谓对称加密就是，将信息和私钥通过某种算法混合在一起，这样除非知道私钥，不然无法获取内容，**而正好客户端和服务端都知道这个私钥，所以只要加密算法够彪悍，私钥够复杂，数据就够安全。  **7. 传输加密后的信息**  **这部分信息是服务段用私钥加密后的信息，可以在客户端被还原**  **8. 客户端解密信息**  **客户端用之前生成的随机值(私钥)解密服务段传过来的信息，于是获取了解密后的内容。**整个过程第三方即使监听到了数据，也束手无策。 |

非对称加密算法：RSA，DSA/DSS     在客户端与服务端相互验证的过程中用的是非对称加密   
对称加密算法：AES，RC4，3DES     客户端与服务端相互验证通过后，以随机数作为密钥时，就是对称加密  
HASH算法：MD5，SHA1，SHA256  在确认握手消息没有被篡改时

1. **https的缺点是什么？**

缺点：

1.  https 需要证书。

2.  因为对传输进行加密，会一定程度增加cpu消耗。

3.  由于https 要还密钥和确认加密算法的需要，所以首次建立连接会慢一些。

4.  带宽消耗会增加。

1. **事务怎么样才算提交成功。**

MySQL是通过WAL方式，来保证数据库事务的一致性和持久性，即ACID特性中的C(consistent)和D(durability)。

WAL（Write-Ahead Logging）是一种实现事务日志的标准方法，具体而言就是:

**1、修改记录前，一定要先写日志；**

**2、事务提交过程中，一定要保证日志先落盘，才能算事务提交完成。**

通过WAL方式，在保证事务特性的情况下，可以提高数据库的性能。

1. **线程池中所有的参数讲解一下**

* corePoolSize

线程池的核心线程数。在没有设置 allowCoreThreadTimeOut 为 true 的情况下，核心线程会在线程池中一直存活，即使处于闲置状态。

* maximumPoolSize

线程池所能容纳的最大线程数。当活动线程(核心线程+非核心线程)达到这个数值后，后续任务将会根据 RejectedExecutionHandler 来进行拒绝策略处理。

* keepAliveTime

非核心线程闲置时的超时时长。超过该时长，非核心线程就会被回收。若线程池通过 allowCoreThreadTimeOut() 方法设置 allowCoreThreadTimeOut 属性为 true，则该时长同样会作用于核心线程，AsyncTask 配置的线程池就是这样设置的。

* unit

keepAliveTime 时长对应的单位。

* workQueue

线程池中的任务队列，通过线程池的 execute() 方法提交的 Runnable 对象会存储在该队列中。

* ThreadFactory

线程工厂，功能很简单，就是为线程池提供创建新线程的功能。这是一个接口，可以通过自定义，做一些自定义线程名的操作。

* RejectedExecutionHandler

当任务无法被执行时(超过线程最大容量 maximum 并且 workQueue 已经被排满了)的处理策略，这里有四种任务拒绝类型。

1. **线程池中从corePoolSize到maximumPoolSize转变过程.**

## 线程池工作原则

* 1、当线程池中线程数量小于 corePoolSize 则创建线程，并处理请求。
* 2、当线程池中线程数量大于等于 corePoolSize 时，则把请求放入 workQueue 中,随着线程池中的核心线程们不断执行任务，只要线程池中有空闲的核心线程，线程池就从workQueue 中取任务并处理。
* 3 、当 taskQueue 已存满，放不下新任务时则新建非核心线程入池，并处理请求直到线程数目达到 maximumPoolSize（最大线程数量设置值）。
* 4、如果线程池中线程数大于 maximumPoolSize 则使用 RejectedExecutionHandler 来进行任务拒绝处理。

1. **403和500状态分别讲解一下，他们之间有什么区别；**

403 - 禁止访问;500 - 内部服务器错误   
区别：   
403：服务器拒绝请求。如果您在 Googlebot 尝试抓取您网站上的有效网页时看到此状态码，可能是您的服务器或主机拒绝了 Googlebot 访问。   
500：服务器遇到错误，无法完成请求。 

1. **数据库的乐观锁原理和实现；**

悲观锁：在关系数据库管理系统里，悲观并发控制（又名“悲观锁”，Pessimistic Concurrency Control，缩写“PCC”）是一种并发控制的方法。它可以阻止一个事务以影响其他用户的方式来修改数据。**如果一个事务执行的操作都某行数据应用了锁，那只有当这个事务把锁释放，其他事务才能够执行与该锁冲突的操作。**  
悲观并发控制主要用于数据争用激烈的环境，以及发生并发冲突时使用锁保护数据的成本要低于回滚事务的成本的环境中。   
乐观锁：在关系数据库管理系统里，乐观并发控制（又名“乐观锁”，Optimistic Concurrency Control，缩写“OCC”）是一种并发控制的方法。**它假设多用户并发的事务在处理时不会彼此互相影响，各事务能够在不产生锁的情况下处理各自影响的那部分数据。在提交数据更新之前，每个事务会先检查在该事务读取数据后，有没有其他事务又修改了该数据。如果其他事务有更新的话，正在提交的事务会进行回滚。**乐观事务控制最早是由孔祥重（H.T.Kung）教授提出。

乐观锁的实现  
        使用数据版本（Version）记录机制实现，这是乐观锁最常用的一种实现方式。何谓数据版本？即为数据增加一个版本标识，一般是通过为数据库表增加一个数字类型的 “version” 字段来实现。当读取数据时，将version字段的值一同读出，数据每更新一次，对此version值加一。当我们提交更新的时候，判断数据库表对应记录的当前版本信息与第一次取出来的version值进行比对，如果数据库表当前版本号与第一次取出来的version值相等，则予以更新，否则认为是过期数据

**另外，java中的compareandswap即cas，解决多线程并行情况下使用锁造成性能损耗的一种机制。**

          CAS操作包含三个操作数，内存位置（V）,预期原值（A）和新值（B）。如果内存位置的值与预期原值相匹配，那么处理器会西东将该位置值更新为新值。否则，处理器不做任何操作。

          记录2: id，stauts，status 包含3种状态值 1，2，3

           操作，update status＝3 where id＝111 and status＝1；

           即 如果内存值为1，预期值为1，则修改新值。对于没有执行的操作则丢弃。

1. **Spring中的事务原理讲一下；**

配置文件开启注解驱动，在相关的类和方法上通过注解@Transactional标识。  
    spring 在启动的时候会去解析生成相关的bean，这时候会查看拥有相关注解的类和方法，并且为这些类和方法生成代理，并根据@Transaction的相关参数进行相关配置注入，这样就在代理中为我们把相关的事务处理掉了（开启正常提交事务，异常回滚事务）。

1. **数据库里的事务原理讲一下；**

真正的数据库层的事务提交和回滚是通过binlog或者redo log实现的。

1. **HashMap和HashSet的实现原理**

**HashSet实现Set接口，由哈希表（实际上是一个HashMap实例）支持。**它不保证set 的迭代顺序；特别是它不保证该顺序恒久不变。此类允许使用null元素。HashSet中不允许有重复元素，这是因为HashSet是基于HashMap实现的，HashSet中的元素都存放在HashMap的key上面，而value中的值都是统一的一个**private static final Object PRESENT = new Object();**。HashSet跟HashMap一样，都是一个存放链表的数组。

**HashSet中add方法调用的是底层HashMap中的put()方法，而如果是在HashMap中**调用put，首先会判断key是否存在，如果key存在则修改value值，如果key不存在这插入这个key-value。而在set中，因为value值没有用，也就不存在修改value值的说法，因此往HashSet中添加元素，首先判断元素（也就是key）是否存在，**如果不存在这插入，如果存在着不插入，这样HashSet中就不存在重复值。**

HashSet的实现：

对于HashSet而言，它是基于HashMap实现的，HashSet底层使用HashMap来保存所有元素，更确切的说，HashSet中的元素，只是存放在了底层HashMap的key上， 而value使用一个static final的Object对象标识。因此HashSet 的实现比较简单，相关HashSet的操作，基本上都是直接调用底层HashMap的相关方法来完成， HashSet的源代码如下：

1. **动态代理的原理**

博客2章15篇

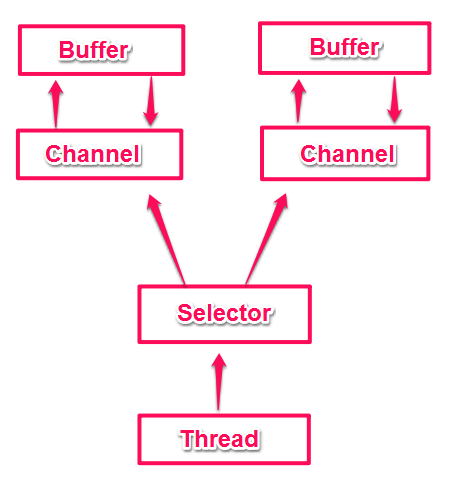
1. **数据库挂了怎么怎么办**

首先查看日志吧，看看有没有error或alert之类的

1. **分布式事务怎么保证一致性**

博客7章24篇

1. **Channel和buffer**



Java NIO的主要构成核心就是Buffer、Channel和Selector这三个

对于Channel我想要提醒的是，Channel中的数据总是要先读到一个Buffer，或者总是要从一个Buffer中写入

使用Selector，得向Selector注册Channel，然后调用它的select()方法。这个方法会一直阻塞到某个注册的通道有事件就绪。一旦这个方法返回，线程就可以处理这些事件

**源文件中将数据通过通道读到这个缓冲区中，然后再通过通道将缓冲区写入目标文件**

1. **directBuffer和buffer的区别**

**direct buffer 直接分配在JVM之外的物理内存，而不是 JVM 中的逻辑内存。**优点在于用它存储、发送数据比在堆内的 buffer 快，因为少了一次拷贝（后面说），但你要对其中数据进行操作就有点麻烦了哦，毕竟人家在堆外，很难管得到。

1. **怎么解决缓存和主存的一致性问题**

　1）通过在总线加LOCK#锁的方式

2）**通过缓存一致性协议**

这2种方式都是硬件层面上提供的方式。

**Volatile关键字，synchronized加锁等。**

1. **枚举可以继承抽象类吗？枚举可以实现接口吗，为什么？**

**可以实现接口，但不能继承类，**因为所有枚举类都继承自java.lang.Enum（由编译器添加），同时java不支持多继承。

当使用enum定义，非抽象的枚举类时，默认会使用final来修饰，因此枚举类不能派生子类。  
但，当是抽象枚举类时，系统会用abstract来修饰，此时可以派生匿名子类实例，就像在枚举类中为每个枚举值实现继承自接口的各自的行为时，使用匿名子类实例，来完成各自的行为。  
所以当是关键在于看枚举类前面的修饰符，

1. **spring@resource和@autowired区别。**

@Autowired 与@Resource：  
**1、@Autowired与@Resource都可以用来装配bean. 都可以写在字段上,或写在setter方法上。**  
  
2、@Autowired**默认按类型装配**（这个注解是属业spring的），默认情况下必须要求依赖对象必须存在，如果要允许null值，可以设置它的required属性为false，如：  
@Autowired(required=false) ，如果我们想使用名称装配可以结合@Qualifier注解进行使用，如下：  
Java代码

@Autowired() @Qualifier("baseDao")

**private BaseDao baseDao;**

3、@Resource **是JDK1.6支持的注解**，**默认按照名称进行装配**，名称可以通过name属性进行指定，如果没有指定name属性，当注解写在字段上时，默认取字段名，按照名称查找，如果注解写在setter方法上默认取属性名进行装配。当找不到与名称匹配的bean时才按照类型进行装配。但是需要注意的是，如果name属性一旦指定，就只会按照名称进行装配。  
  
Java代码

**@Resource(name="baseDao")**

**private BaseDao baseDao;**

他们的主要区别就是@Autowired是默认按照类型装配的 @Resource默认是按照名称装配的  
**byName 通过参数名 自动装配，**如果一个bean的name 和另外一个bean的 property 相同，就自动装配。  
**byType 通过参数的数据类型自动装配，**如果一个bean的数据类型和另外一个bean的property属性的数据类型兼容，就自动装配

****

<bean id="userServiceImpl"

class="cn.com.bochy.service.impl.UserServiceImpl"

autowire="byName">

</bean>

<bean id="userDao"

class="cn.com.bochy.dao.impl.UserDaoImpl">

</bean>

比如说如上这段代码，byName就是通过Bean的id或者name，byType就是按Bean的Class的类型。

若autowire="byType"意思是通过 class="cn.com.bochy.dao.impl.UserDaoImpl"来查找UserDaoImpl下所有的对象。

代码autowire="byName"意思是通过id="userDao"来查找Bean中的userDao对象

1. **http报头**



1. **ThreadLocal，知不知道这个？ 作用是什么？用过没？ 应用场景有哪些？**

ThreadLocal，很多地方叫做**线程本地变量**，也有些地方叫做**线程本地存储**，其实意思差不多。可能很多朋友都知道**ThreadLocal为变量在每个线程中都创建了一个副本**，那么每个线程可以访问自己内部的副本变量。

**应用场景有数据库连接，session管理等。**

1. **32位操作系统与64位操作系统，最大的区别是什么？**

32位和64位的区别就是一次性的运算量不一样，理论上64位的会比32位快1倍，内存寻址也不一样

1. **在网站前台和后台数据传输，采用json的格式，你觉得这种格式有什么优缺点？ json的缺点是有数据中有很多重复项，你觉得怎么优化？**

对于JSON，首先要明白JSON和XML一样也是一种简单文本格式。相对于XML**，它更加易读、更便于肉眼检查。**在语法的层面上，JSON与其他格式的区别是**在于分隔数据的字符，**JSON中的分隔符限于单引号、小括号、中括号、大括号、冒号和逗号。

1、轻量级的数据交换格式  
2、人们读写更加容易  
3、易于机器的解析和生成  
4、能够通过JavaScript中eval()函数解析JSON  
5、JSON支持多语言。包括：ActionScript, C, C#, ColdFusion, E, Java, JavaScript, ML, Objective CAML, Perl, PHP, [**Python**](http://lib.csdn.net/base/11), Rebol, Ruby, and Lua.

**太多的重复可能要在前端写**JOSN数组去重的算法 去优化数据了。