# 百度面试

## 1 Spring AOP是用什么方法具体实现的

AOP技术利用一种称为"横切"的技术，能够剖解开封装的对象内部，并将那些影响了多个类并且与具体业务无关的公共行为 封装成一个独立的模块（称  
为切面）。更重要的是，它又能以巧夺天功的妙手将这些剖开的切面复原，不留痕迹的融入核心业务逻辑中。这样，对于日后横切功能的编辑和重用都能够带来极大的方便。AOP技术的具体实现，无非也就是通过动态代理技术或者是在程序编译期间进行静态的"织入"方式。下面是这方面技术的几个基本术语：1、join point（连接点）：是程序执行中的一个精确执行点，例如类中的一个方法。它是一个抽象的概念，在实现AOP时，并不需要去定义一个join point。2、point cut（切入点）：本质上是一个捕获连接点的结构。在AOP中，可以定义一个point cut，来捕获相关方法的调用。3、advice（通知）：是point cut的执行代码，是执行“方面”的具体逻辑。4、aspect（方面）：point cut和advice结合起来就是aspect，它类似于OOP中定义的一个类，但它代表的更多是对象间横向的关系。5、introduce（引入）：为对象引入附加的方法或属性，从而达到修改对象结构的目的。有的OP工具又将其称为mixin。

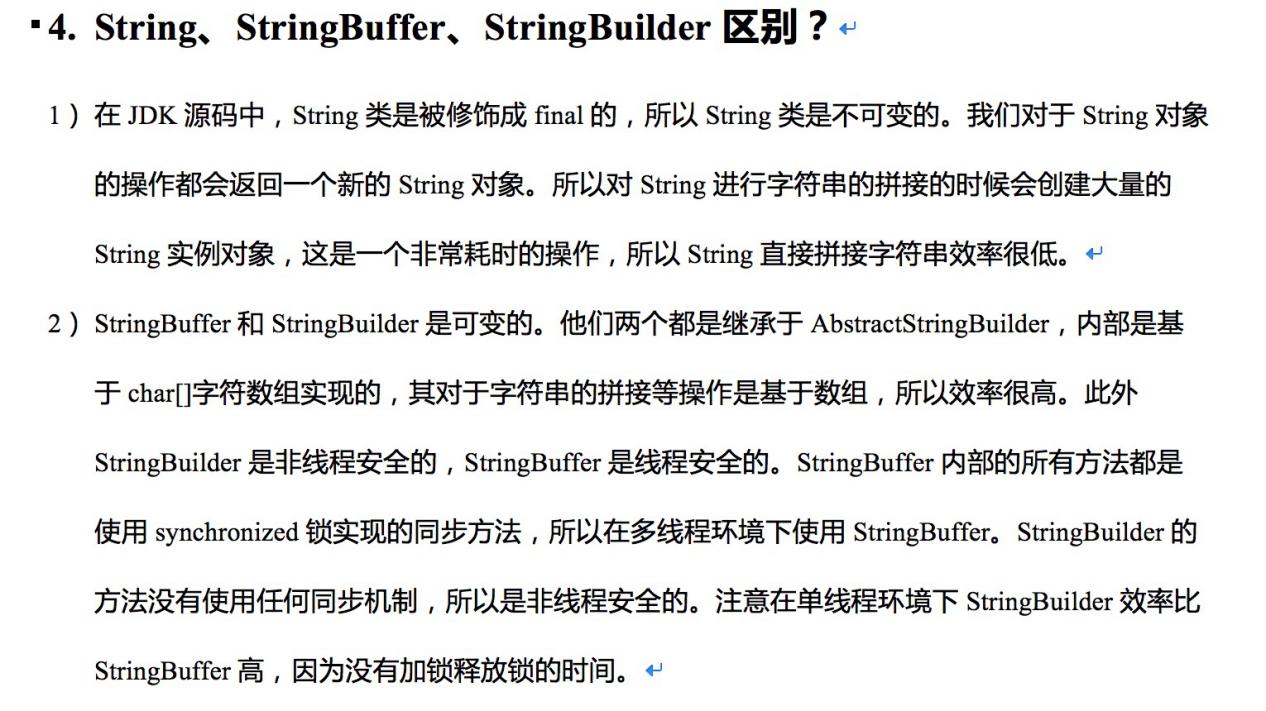
IOC:控制反转也叫依赖注入。利用了工厂模式将对象交给容器管理，你只需要在spring配置文件总配置相应的bean，以及设置相关的属性，让spring容器来生成类的实例对象以及管理对象。在spring容器启动的时候，spring会把你在配置文件中配置的bean都初始化好，然后在你需要调用的时候，就把它已经初始化好的那些bean分配给你需要调用这些bean的类（假设这个类名是A），分配的方法就是调用A的setter方法来注入，而不需要你在A里面new这些bean了。  
注意：面试的时候，如果有条件，画图，这样更加显得你懂了  
AOP:面向切面编程。（Aspect-Oriented Programming）  
AOP可以说是对OOP的补充和完善。OOP引入封装、继承和多态性等概念来建立一种对象层次结构，用以模拟公共行为的一个集合。当我们需要为分散的对象引入公共行为的时候，OOP则显得无能为力。也就是说，OOP允许你定义从上到下的关系，但并不适合定义从左到右的关系。例如日志功能。日志代码往往水平地散布在所有对象层次中，而与它所散布到的对象的核心功能毫无关系。在OOP设计中，它导致了大量代码的重复，而不利于各个模块的重用。  
将程序中的交叉业务逻辑（比如安全，日志，事务等），封装成一个切面，然后注入到目标对象（具体业务逻辑）中去。  
实现AOP的技术，主要分为两大类：一是采用动态代理技术，利用截取消息的方式，对该消息进行装饰，以取代原有对象行为的执行；二是采用静态织入的方式，引入特定的语法创建“方面”，从而使得编译器可以在编译期间织入有关“方面”的代码  
简单点解释，比方说你想在你的biz层所有类中都加上一个打印‘你好’的功能这你经可以用aop思想来做，你先写个类写个方法，方法经实现打印‘你好’让后你Ioc这个类 ref＝“biz.\*”让每个类都注入。

## 2 Spring的事务隔离级别，平常用到哪一级，如何配

声明式事务：（注解，XML）  
编程式事务：（代码中嵌入）  
隔离级别：  
1、Serializable：最严格的级别，事务串行执行，资源消耗最大；  
2、REPEATABLE READ：保证了一个事务不会修改已经由另一个事务读取但未提交（回滚）的数据。避免了“脏读取”和“不可重复读取”的情况，但是带来了更多的性能损失。  
3、READ COMMITTED:大多数主流数据库的默认事务等级，保证了一个事务不会读到另一个并行事务已修改但未提交的数据，避免了“脏读取”。该级别适用于大多数系统。  
4、Read Uncommitted：保证了读取过程中不会读取到非法数据。  
隔离级别在于处理多事务的并发问题。我们知道并行可以提高数据库的吞吐量和效率，但是并不是所有的并发事务都可以并发运行，这需要查看数据库教材的可串行化条件判断了。  
事务的隔离级别的话，和数据库硬件相关

## 3 String stringbuffer stringbuilder区别，平时用到了哪些

1.在执行速度方面的比较：StringBuilder > StringBuffer   
2.StringBuffer与StringBuilder，他们是字符串变量，是可改变的对象，每当我们用它们对字符串做操作时，实际上是在一个对象上操作的，不像String一样创建一些对象进行操作，所以速度就快了。  
3.StringBuilder：线程非安全的  
 StringBuffer：线程安全的  
 当我们在字符串缓冲去被多个线程使用是，JVM不能保证StringBuilder的操作是安全的，虽然他的速度最快，但是可以保证StringBuffer是可以正确操作的。当然大多数情况下就是我们是在单线程下进行的操作，所以大多数情况下是建议用StringBuilder而不用StringBuffer的，就是速度的原因。  
  
对于三者使用的总结：1.如果要操作少量的数据用 = String  
2.单线程操作字符串缓冲区 下操作大量数据 = StringBuilder  
3.多线程操作字符串缓冲区 下操作大量数据 = StringBuffer



## 4 Synchronized两种方法。静态和普通方法加上Synchronized有什么区别

所有的非静态同步方法用的都是同一把锁——实例对象本身，也就是说如果一个实例对象的非静态同步方法获取锁后，该实例对象的其他非静态同步方法必须等待获取锁的方法释放锁后才能获取锁，可是别的实例对象的非静态同步方法因为跟该实例对象的非静态同步方法用的是不同的锁，所以毋须等待该实例对象已获取锁的非静态同步方法释放锁就可以获取他们自己的锁。  
 而所有的静态同步方法用的也是同一把锁——类对象本身，这两把锁是两个不同的对象，所以静态同步方法与非静态同步方法之间是不会有竞态条件的。但是一旦一个静态同步方法获取锁后，其他的静态同步方法都必须等待该方法释放锁后才能获取锁，而不管是同一个实例对象的静态同步方法之间，还是不同的实例对象的静态同步方法之间，只要它们同一个类的实例对象！

## 5 LINUX命令：在一个字符文件中查找一个子串

可以使用vim打开文件，然后通过 vim编辑 中的 /（向后查找）或者 ?（向前查找）来查找相应的字符串。

## 6 谈谈你对内存的了解，（类存在堆还是栈中）？

Java把内存划分成两种：一种是栈内存，一种是堆内存。  
　　在函数中定义的一些基本类型的变量和对象的引用变量都在函数的栈内存中分配。当在一段代码块定义一个变量时，Java就在栈中为这个变量分配内存空间，当超过变量的作用域后，Java会自动释放掉为该变量所分配的内存空间，该内存空间可以立即被另作他用。  
　　堆内存用来存放由new创建的对象和数组。在堆中分配的内存，由Java虚拟机的自动垃圾回收器来管理。  
　　在堆中产生了一个数组或对象后，还可以在栈中定义一个特殊的变量，让栈中这个变量的取值等于数组或对象在堆内存中的首地址，栈中的这个变量就成了数组或对象的引用变量。  
　　引用变量就相当于是为数组或对象起的一个名称，以后就可以在程序中使用栈中的引用变量来访问堆中的数组或对象。  
　　具体的说：栈与堆都是Java用来在Ram中存放数据的地方。与C++不同，Java自动管理栈和堆，程序员不能直接地设置栈或堆。  
　　Java的堆是一个运行时数据区,类的(对象从中分配空间。这些对象通过new、newarray、anewarray和 multianewarray等指令建立，它们不需要程序代码来显式的释放。堆是由垃圾回收来负责的，堆的优势是可以动态地分配内存大小，生存期也不必事先告诉编译器，因为它是在运行时动态分配内存的，Java的垃圾收集器会自动收走这些不再使用的数据。但缺点是，由于要在运行时动态分配内存，存取速度较慢。  
　　栈的优势是，存取速度比堆要快，仅次于寄存器，栈数据可以共享。但缺点是，存在栈中的数据大小与生存期必须是确定的，缺乏灵活性。栈中主要存放一些基本类型的变量（,int, short, long, byte, float, double, boolean, char）和对象句柄。  
　　栈有一个很重要的特殊性，就是存在栈中的数据可以共享。假设我们同时定义：  
　　int a = 3;  
　　int b = 3；  
　　编译器先处理int a = 3；首先它会在栈中创建一个变量为a的引用，然后查找栈中是否有3这个值，如果没找到，就将3存放进来，然后将a指向3。接着处理int b = 3；在创建完b的引用变量后，因为在栈中已经有3这个值，便将b直接指向3。这样，就出现了a与b同时均指向3的情况。这时，如果再令a=4；那么编译器会重新搜索栈中是否有4值，如果没有，则将4存放进来，并令a指向4；如果已经有了，则直接将a指向这个地址。因此a值的改变不会影响到b的值。要注意这种数据的共享与两个对象的引用同时指向一个对象的这种共享是不同的，因为这种情况a的修改并不会影响到b, 它是由编译器完成的，它有利于节省空间。而一个对象引用变量修改了这个对象的内部状态，会影响到另一个对象引用变量。

## 7 垃圾回收是什么？垃圾回收的过程（我问他是说垃圾回收算法吗，他说不是）

java中垃圾回收以前听老师讲好像是内存满了他才去做一次整体垃圾回收,在回收垃圾的同时会调用finalize方法.你在构造一个类时可以构造一个类时覆盖他的finalize方法以便于该类在被垃圾回收时执行一些代码,比如释放资源.

Java虚拟机中有个称之为垃圾回收器的东西，实际上这个东西也许真正不存在，或者是已经集成到JVM中了，但这无关紧要，仍然可以称为为垃圾回收器。  
 垃圾回收器的作用是查找和回收（清理）无用的对象。以便让JVM更有效的使用内存。  
 垃圾回收器的运行时间是不确定的，由JVM决定，在运行时是间歇执行的。虽然可以通过System.gc()来强制回收垃圾，但是这个命令下达后无法保证JVM会立即响应执行，但经验表明，下达命令后，会在短期内执行你的请求。JVM通常会感到内存紧缺时候去执行垃圾回收操作。  
 垃圾回收过于频繁会导致性能下降，过于稀疏会导致内存紧缺。这个JVM会将其控制到最好，不用程序员担心。但有些程序在短期会吃掉大量内存，而这些恐怖的对象很快使用结束了，这时候也许有必要强制下达一条垃圾回命令，这是很有必要的，以便有更多可用的物理内存。  
 垃圾回收器仅仅能做的是尽可能保证可用内存的使用效率，让可用内存得到高效的管理。程序员可以影响垃圾回收的执行，但不能控制。  
 总之，在Java语言中，判断一块内存空间是否符合垃圾收集器收集标准的标准只有两个：  
 1．给对象赋予了空值null，以下再没有调用过。  
 2．给对象赋予了新值，既重新分配了内存空间。

## 8 SQL语句，一个工资表（Date,userid,income）一条记录表示某天某个人当天收入，写两个SQL：每个人当月月的总收入，每个人每天收入的平均值（总收入/有收入的天数）

select \* from T\_news WHERE (DATEPART(yy, addtime) = DATEPART(yy, GETDATE())) AND (DATEPART(mm, addtime) = DATEPART(mm, GETDATE())) GROUPBY userid

## 9 两个单项链表相交求交点

1先遍历第一个链表到他的尾部，然后将尾部的next指针指向第二个链表(尾部指针的next本来指向的是null)。这样两个链表就合成了一个链表，判断原来的两个链表是否相交也就转变成了判断新的链表是否有环的问题了：即判断单链表是否有环？

2仔细研究两个链表，如果他们相交的话，那么他们最后的一个节点一定是相同的，否则是不相交的。因此判断两个链表是否相交就很简单了，分别遍历到两个链表的尾部，然后判断他们是否相同，如果相同，则相交；否则不相交。假设第一个链表长度为len1，第二个问len2，然后找出长度较长的，让长度较长的链表指针向后移动|len1 - len2| (len1-len2的绝对值)，然后在开始遍历两个链表，判断节点是否相同即可。

## 10很多数据内存不能存下，数据是实数，选出n个随机数，要求概率平均

## 11 有一个栈，设计一种方法或结构能够知道当前栈中的最大值，时间复杂度要O（1）

再加一个辅助栈，每次push当前栈的时候，和辅助栈的peek比较，如果大于就push进辅助栈，如果小于，就把辅助栈的peek再push一遍。

## 12 一个int数组（如int[10]）代表一只股票十天的价格，在一天只能买1股或者卖出当前全部股票，写个程序求出10天的最大收益额

贪心算法

## 13 数据库隔离级别介绍、举例说明

  
14. override和overload的区别。

override（重写，覆盖）   
1、方法名、参数、返回值相同。   
2、子类方法不能缩小父类方法的访问权限。   
3、子类方法不能抛出比父类方法更多的异常(但子类方法可以不抛出异常)。   
4、存在于父类和子类之间。   
5、方法被定义为final不能被重写。   
  
overload（重载，过载）   
1、参数类型、个数、顺序至少有一个不相同。     
2、不能重载只有返回值不同的方法名。   
3、存在于父类和子类、同类中。   
  
方法的重写(Overriding)和重载(Overloading)是Java多态性的不同表现。   
重写(Overriding)是父类与子类之间多态性的一种表现，而重载(Overloading)是一个类中多态性的一种表现。

15. 求二叉树的最大距离（即相距最远的两个叶子节点），说思想。

**情况A: 路径经过左子树的最深节点，通过根节点，再到右子树的最深节点。  
情况B: 路径不穿过根节点，而是左子树或右子树的最大距离路径，取其大者**

对于情况A来说，只需要知道左右子树的深度，然后加起来即可。

对于情况B来说，需要知道左子树的最远距离，右子树的最远距离。

只需要计算这两种情况的路径距离，并取其最大值，就是该二叉树的最大距离。

16. 两个栈实现一个队列，说思想。

如果stack2中不为空时，弹出stack2中的栈顶元素即可，如果stack2为空的时候，就先把stack1中的元素弹出依次压入stack2中，之后再从stack2中弹出即可。

## 17 谈谈你对数据库中索引的理解

1、首先要明白无名无实莫要用索引：因为数据中的记录很多，为了方便我们查找，提高查询的效率；

2、索引的原理：对要查询的字段建立索引其实就是把该字段按照一定的方式排序；建立的索引只对该字段有用，如果查询的字段改变，那么这个索引也就无效了，比如图书馆的书是按照书名的第一个字母排序的，那么你想要找作者叫张三的就不能用改索引了；还有就是如果索引太多会降低查询的速度

3、索引是优缺点：首先明白为什么索引会增加速度，DB在执行一条Sql语句的时候，默认的方式是根据搜索条件进行全表扫描，遇到匹配条件的就加入搜索结果集合。如果我们对某一字段增加索引，查询时就会先去索引列表中一次定位到特定值的行数，大大减少遍历匹配的行数，所以能明显增加查询的速度。那么在任何时候都应该加索引么？这里有几个反例：1、如果每次都需要取到所有表记录，无论如何都必须进行全表扫描了，那么是否加索引也没有意义了。2、对非唯一的字段，例如“性别”这种大量重复值的字段，增加索引也没有什么意义。3、对于记录比较少的表，增加索引不会带来速度的优化反而浪费了存储空间，因为索引是需要存储空间的，而且有个致命缺点是对于update/insert/delete的每次执行，字段的索引都必须重新计算更新。所以并不是任何情况下都改建立索引的

## 18 现在普通关系数据库用得数据结构是什么类型的数据结构

查找表等用的是索引结构,数据表记录一般用顺序存储

## 19 索引的优点和缺点

创建索引可以大大提高系统的性能:  
第一，通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性。   
第二，可以大大加快数据的检索速度，这也是创建索引的最主要的原因。   
第三，可以加速表和表之间的连接，特别是在实现数据的参考完整性方面特别有意义。   
第四，在使用分组和排序 子句进行数据检索时，同样可以显著减少查询中分组和排序的时间。   
第五，通过使用索引，可以在查询的过程中，使用优化隐藏器，提高系统的性能。   
增加索引也有许多不利的方面:  
第一，创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加。   
第二，索引需要占物理空间，除了数据表占数据空间之外，每一个索引还要占一定的物理空间，如果要建立聚簇索引，那么需要的空间就会更大。   
第三，当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，这样就降低了数据的维护速度。   
索引是建立在数据库表中的某些列的上面。因此，在创建索引的时候，应该仔细考虑在哪些列上可以创建索引，在哪些列上不能创建索引。一般来说，应该在这些列上创建索引，例如：   
在经常需要搜索的列上，可以加快搜索的速度；   
在作为主键的列上，强制该列的唯一性和组织表中数据的排列结构；   
在经常用在连接的列上，这 些列主要是一些外键，可以加快连接的速度；   
在经常需要根据范围进行搜索的列上创建索引，因为索引已经排序，其指定的范围是连续的；   
在经常需要排序的列上创 建索引，因为索引已经排序，这样查询可以利用索引的排序，加快排序查询时间；   
在经常使用在WHERE子句中的列上面创建索引，加快条件的判断速度。

## 20 session和cache的区别是什么

Session是由应用服务器维持的一个服务器端的存储空间，用户在连接服务器时，会由服务器生成一个唯一的SessionID,用该SessionID 为标识符来存取服务器端的Session存储空间。而SessionID这一数据则是保存到客户端，用Cookie保存的，用户提交页面时，会将这一 SessionID提交到服务器端，来存取Session数据。这一过程，是不用开发人员干预的。所以一旦客户端禁用Cookie，那么Session也会失效。

CACHE ，则是服务器端的缓存，是所有用户都可以访问和共享的。

## 21 如果有几千个session，怎么提高效率

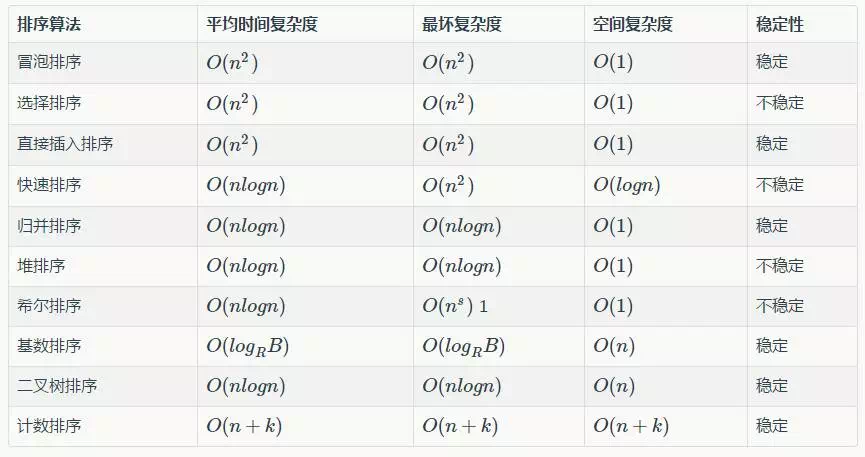
把session放到 redis 或 memcache 等此类内存缓存中或着把session存储在SSD硬盘上。session对应的文件有一个特点就是小，一般在几KB左右，  
如果session以文件方式存储，如果并发数量级有几千个，此时系统硬盘的随机IO早已成了系统中的最大瓶颈了，因为会话文件是存储在多个小文件中，映射到存储空间不是一段连续的地址范围所以硬盘的随机读取能力显得非常重要，而觉机械硬盘的随机IO一般在100/iops上下，(IOPS (Input/Output Operations Per Second)，即每秒进行读写（I/O）操作的次数)SSD固态硬盘可以达几百至上千,所以在这么高并发读写的情况下如果无条件用SSD固态盘可以把session放到 redis 或 memcache 等内存缓存中，系统对内存的操作又是非常快的，  
只要你的内存足够大，再多session并发速度一样不会慢。

## 22 session是存储在什么地方，以什么形式存储的。

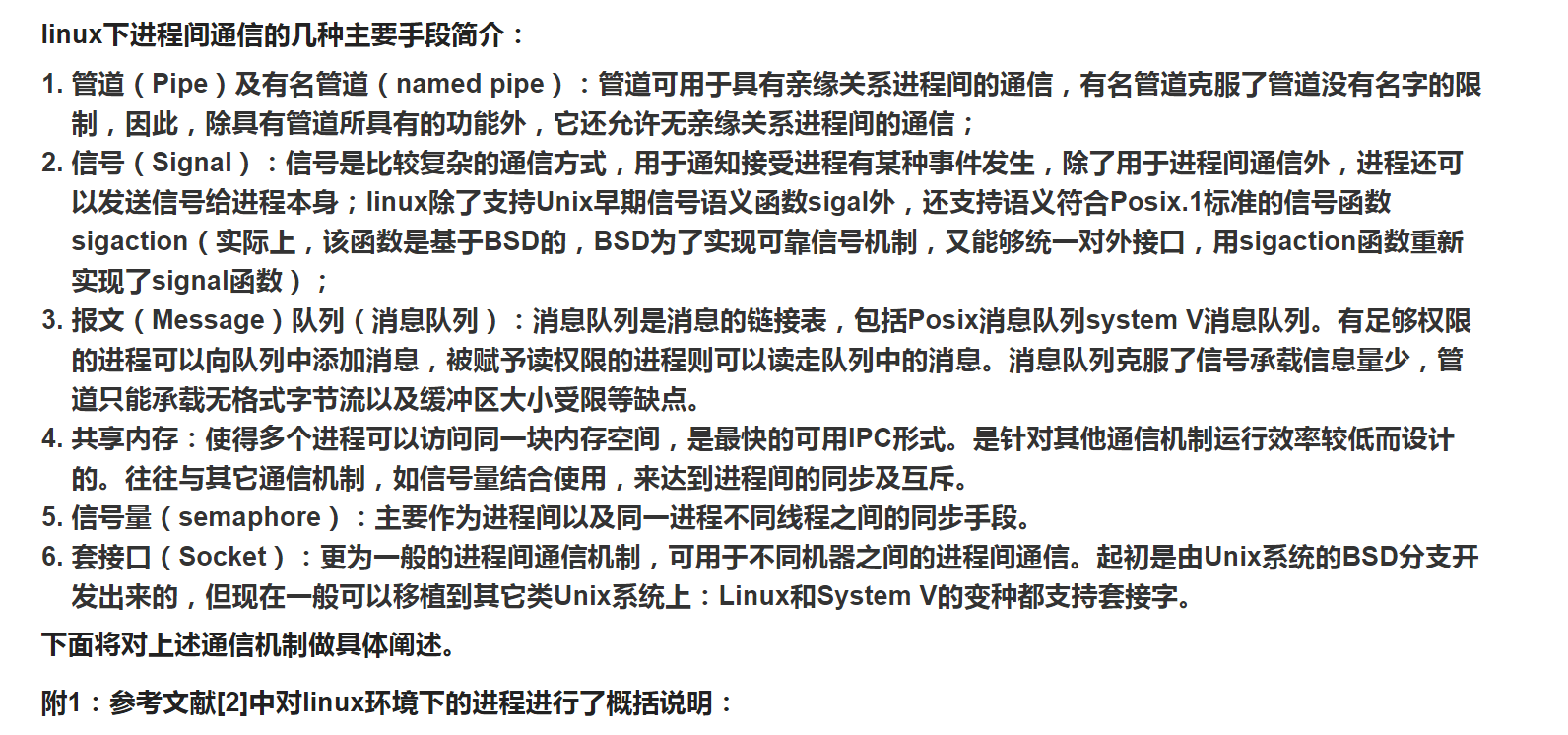
session变量保存在网页服务器中，你不能直接修改，当然，调用程序中的setAttribute()方法当然可以了。cookie存储的可不是具体的数据，要不岂不是太不安全了，谁都可以修改session变量了，网站也毫无安全性可言。实际，在cookie中，存储的是一个sessionId，它标示了一个服务器中的session变量，通过这种方式，服务器就知道你到底是那个session了。顺便说一句，如果客户端不支持cookie,session也是可以实现的，在服务器端通过urlEncoder，可以实现sessionId的传递。所以，记住客户端只存储session标识，实际内容在网页服务器中。

以键值对的方式存储的

## 23 各种排序



## 24 进程、线程之间通信



****Linux下的进程间通信机制大致包括：管道、信号（在Windows上成为消息）、信号队列（实际是消息链表）、共享内存、信号量、套接字。****

线程 1.全局变量 线程间的通信——wait及notify方法

百度实习生电面

1 模拟扑克牌发牌，例如1-54代表每张牌，保证每个数字都出现一次，但是顺序随机

答：把54张牌存到大小为54的数组中，首先产生1-54中的一个随机数，取以该随机数为下标的牌发出来，并用最后一张牌（第54张）占据该位置，然后产生1-53中的以随机数，取以该随机数为下标的牌发出来，并用最后一张牌（第52张）占据该位置；以此类推，直到发完所有牌。

2 有36匹马，1个跑道，每个跑道跑6匹，没有秒表，要比多少次能比出前3名，答案是8次，自己摸索。。。

答：每组单独跑 决出每组的第一名啊 然后每组的第一名在决出123 这样一共是7次吧 最后就是 第一名那组的23名 和第2名那组的12 还有第三名那匹马 一起比 决出2 3名 就这样啊

3 用百度搜索的时候，想除去那些网址不同，但是网页内容相同的地址，怎么做？怎么判断网页内容相同？

答：判断网页内容是否相同的方法，可用于搜索引擎技术领域，过滤网页内容相同的查询结果。根据计算网页标题的相似度和网页正文内容的相似度，根据网页的标题和正文内容的相似度来判断其是否为相同内容。如果二者的相似度达到一定阀值，那么就判定为相同内容的网页，否则就判定为不同内容的网页。

4 怎么找出一个链表的中间节点？

答： 用两个指针从链表头结点开始，一个指针以步长为2向后遍历，另一个以步长为1向后遍历，当步长为2的指针遍历到尾节点时，步长为1的指针刚好指向链表的中间节点

5 怎么判断一个链表是否又环？环的长度？环的的起点？链表长度？

答：1、对于问题1，使用追赶的方法，设定两个指针slow、fast，从头指针开始，每次分别前进1步、2步。如存在环，则两者相遇；如不存在环，fast遇到NULL退出。

6 说几个常用的Linux命令.

pwd   cd   ls  clear  cat   tail   grep  chmod  cp   mv  rm   mkdir  echo   tar

## 7 进程间的死锁是怎么产生的?怎么破除? 答：产生死锁的原因主要是：

（1） 因为系统资源不足。

（2） 进程运行推进的顺序不合适。

（3） 资源分配不当等。

产生死锁的四个必要条件：

（1）互斥条件：一个资源每次只能被一个进程使用。

（2）请求与保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放。

（3）不剥夺条件:进程已获得的资源，在末使用完之前，不能强行剥夺。

（4）循环等待条件:若干进程之间形成一种头尾相接的循环等待资源关系。

8 Sizeof(数组名)的结果是什么？sring 字符串和char\*字符串的区别？

答：Sizeof(数组名)的结果是数组所占内存的大小，如果是作为形参传递，则数组名退化为指针，再使用sizeof就变成求指针（地址）占内存大小。

Char是c++的内置类型char

String是C++标准库中定义的类，c中没有 如果是的话，它两的区别有：

char数组仅仅是存储字符串用的，c库中有一系列操作字符串的函数

String是类，它包含一个可变长度的char数组，封装了常用的字符串操作函数

string 和 vector、list一样，都是标准库类型。 string 类型支持长度可变的字符串，C++ 标准库将负责管理与存储字符相关的内存，以及提供各种有用的操作。

# 关于简历

## 1 mybatis一级缓存与二级缓存

mybatis一级缓存是一个**SqlSession级别**，sqlsession只能访问自己的一级缓存的数据，二级缓存是跨sqlSession，是**mapper级别**的缓存，对于mapper级别的缓存不同的sqlsession是可以共享的。

MyBatis 是支持普通SQL查询，存储过程和高级映射的优秀持久层框架。几乎消除了所有JDBC代码和参数的手工设置。以及结果集的检索。使用简单的XML或注解用于配置和原始映射。

1> 加载配置 ①通过配置文件 ②Java中的注解

2> SQL解析

3> SQL执行

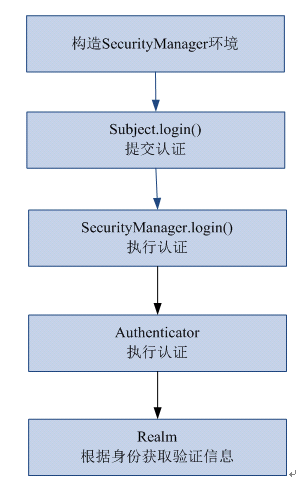
4> 结果集映射：可以将结果集转换成HashMap，JavaBean或基本数据类型

xml: <insert></insert> <delete></delete> id parameterType

@select

DB -> Entity -> Mapper.xml -> mapper.java -> service.java -> controller.java -> jsp

## 2 Shiro认证与授权



自定义realm，然后配置文件配置realm，在controller里面读取配置文件，就可以自动调用realm的授权与认证的方法。

## 3 Hadoop

Hadoop 分布式系统基础框架

Hadoop 实现了一个分布式文件系统 HDFS 它的核心 1> HDFS 提供存储 2> MapReduce 提供计算

## 4 消息队列

服务端是[http://Socket.io](//link.zhihu.com/?target=http://Socket.io" \t "_blank)的集群，供客户端（Web、移动端）连接。集群后面是一个Redis服务器，保存集群中每个节点（我们称之为Cluster）连接的客户端ID。同时Redis里面为每一个Cluster分配了一个队列，保存要推送到这个Cluster的消息。  
  
当有消息从某个客户端发出后，所连接的Cluster从Redis里面获取这个消息的目标客户端ID（由于我们同时支持一对一私聊和群组，因此一条消息可能会被推送到多个客户端），然后把消息Push到每个Cluster的消息队列里面。  
  
每一个Cluster都会以阻塞方式读取它所对应的消息队列，一旦发现有消息，就获取并且查看其目标客户端ID是不是连接在这个Cluster上。如果是，就通过[http://Socket.io](//link.zhihu.com/?target=http://Socket.io" \t "_blank)发送；如果不是就丢弃。然后继续阻塞读取，知道下一条消息到达。

一个Cluster（集群）找到对应的机架（假如一个机架40台服务器）在通过服务器看目标客户端ID是不是连接在上面，如果是通过socket发送，不是就丢弃，然后继续阻塞直到下一条消息到达。

## 5 大规模分布式并发系统

在面对大量用户访问、高并发请求方面，基本的解决方案集中在这样几个环节：

使用高性能的服务器、高性能的**[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "http://blog.csdn.net/bamboolsu/article/details/_blank)**、高效率的编程语言、还有高性能的Web容器。

其他实现技术：

1，HTML静态化

2，  图片服务器分离

3，数据库集群、库表散列

在数据库集群方面，很多数据库都有自己的解决方案，**[Oracle](http://lib.csdn.net/base/oracle" \o "Oracle知识库" \t "http://blog.csdn.net/bamboolsu/article/details/_blank)**、Sybase等都有很好的方案，常用的**[MySQL](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "http://blog.csdn.net/bamboolsu/article/details/_blank)**提供的Master/Slave也是类似的方案，您使用了什么样的DB，就参考相应的解决方案来实施即可。

　　sohu的论坛就是采用了这样的架构，将论坛的用户、设置、帖子等信息进行数据库分离，然后对帖子、用户按照板块和ID进行散列数据库和表，最终可以在配置文件中进行简单的配置便能让系统随时增加一台低成本的数据库进来补充系统性能。

4， 缓存

     缓存一词搞技术的都接触过，很多地方用到缓存。网站架构和网站开发中的缓存也是非常重要。这里先讲述最基本的两种缓存。高级和分布式的缓存在后面讲述。网站程序开发方面的缓存，**[Linux](http://lib.csdn.net/base/linux" \o "Linux知识库" \t "http://blog.csdn.net/bamboolsu/article/details/_blank)**上提供的Memory Cache是常用的缓存接口，可以在web开发中使用，比如用**[Java](http://lib.csdn.net/base/javase" \o "Java SE知识库" \t "http://blog.csdn.net/bamboolsu/article/details/_blank)**开发的时候就可以调用MemoryCache对一些数据进行缓存和通讯共享，一些大型社区使用了这样的架构。另外，在使用web语言开发的时候，各种语言基本都有自己的缓存模块和方法，**[PHP](http://lib.csdn.net/base/php" \o "PHP知识库" \t "http://blog.csdn.net/bamboolsu/article/details/_blank)**有Pear的Cache模块，Java就更多了，.net不是很熟悉，相信也肯定有。

5，镜像

6，负载均衡

负载均衡将是大型网站解决高负荷访问和大量并发请求采用的高端解决办法。　　负载均衡技术发展了多年，有很多专业的服务提供商和产品可以选择，我个人接触过一些解决方法，其中有两个架构可以给大家做参考。

（1）、硬件四层交换

(2)、软件四层交换

7最新：CDN加速技术

什么是CDN？

CDN的全称是内容分发网络。其目的是通过在现有的Internet中增加一层新的网络架构，将网站的内容发布到最接近用户的网络“边缘”，使用户可以就近取得所需的内容，提高用户访问网站的响应速度。

CDN有别于镜像，因为它比镜像更智能，或者可以做这样一个比喻：CDN=更智能的镜像+缓存+流量导流。因而，CDN可以明显提高Internet网络中信息流动的效率。从技术上全面解决由于网络带宽小、用户访问量大、网点分布不均等问题，提高用户访问网站的响应速度。

 CDN的类型特点

CDN的实现分为三类：镜像、高速缓存、专线。

镜像站点（Mirror Site），是最常见的，它让内容直接发布，适用于静态和准动态的数据同步。但是购买和维护新服务器的费用较高，还必须在各个地区设置镜像服务器，配备专业技术人员进行管理与维护。对于大型网站来说，更新所用的带宽成本也大大提高了。

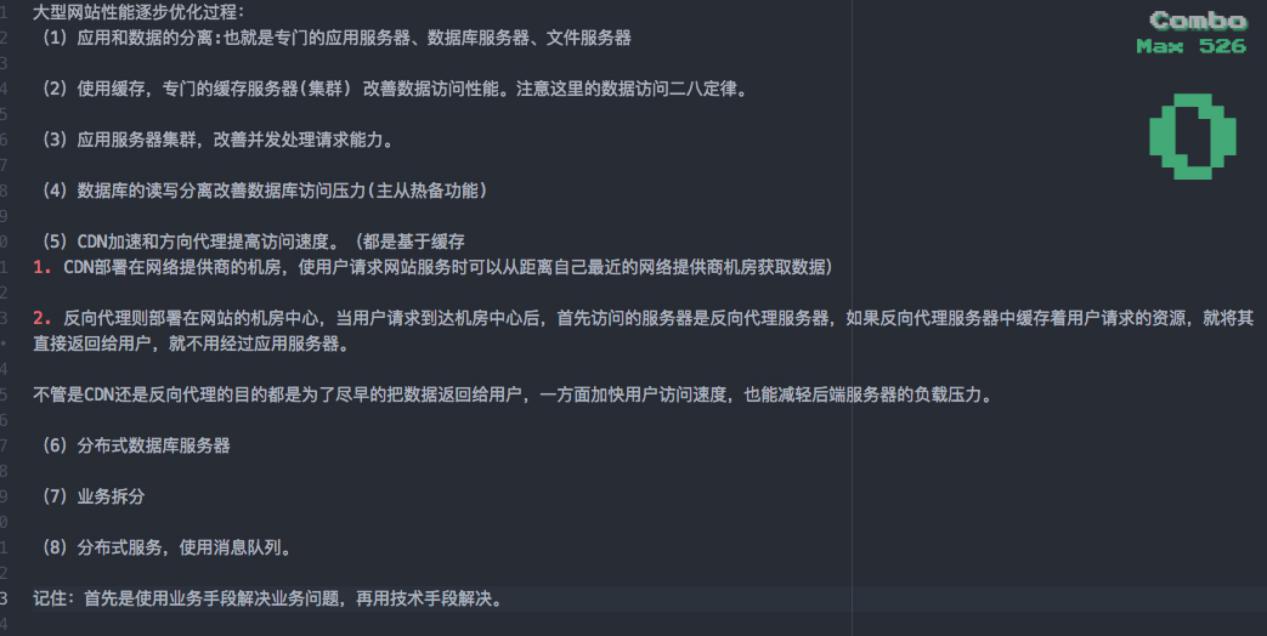
高速缓存，成本较低，适用于静态内容。Internet的统计表明，超过80%的用户经常访问的是20%的网站的内容，在这个规律下，缓存服务器可以处理大部分客户的静态请求，而原始的服务器只需处理约20%左右的非缓存请求和动态请求，于是大大加快了客户请求的响应时间，并降低了原始服务器的负载。

CDN服务一般会在全国范围内的关键节点上放置缓存服务器。

专线，让用户直接访问数据源，可以实现数据的动态同步。

CDN的实例

举个例子来说，当某用户访问网站时，网站会利用全球负载均衡技术，将用户的访问指向到距离用户最近的正常工作的缓存服务器上，直接响应用户的请求。



## 6 SpringMVC

**SpringMVC 如何解析视图**

请求处理方法返回值类型是

**String**   
**ModelAndView**   
**View**

SpringMVC都会奖其转化为**ModelAndView**对象，再给ViewResolver，最后变成视图对象，如JSP、JSTL、PDF等。