# 自我介绍

**自我介绍：**

我叫叶盛，目前是中国人民大学硕士在读，专业是软件工程。我老家浙江金华，目前的打算是毕业以后回浙江工作。我们实验室的研究方向是高性能数据库，研究工作主要围绕PostgreSQL展开。我申请的Java开发实习生职位，因为相对C来说，我觉得Java更具魅力，随着对Java学习的深入，发现自己的漏洞越来越多，但这也让我对Java技术越来越感兴趣。目前我接触了一个开源的Java软件-kettle，一款国外的ETL工具。所谓的ETL就是数据的抽取，转换和装载。

**面试题：**

**Part1：介绍实习，项目经历**

**Part2：**

# J2SE基础

1. Java多线程一定快吗？

不一定。多线程并发并不是在任何情况下都比串行执行块。原因在于，多线程在执行的时候需要不断进行上下文切换，要面临死锁的威胁(加锁，释放所)，还要面临资源有限的挑战。一般而言，小规模的数据处理中，串行执行比多线程执行要有优势，大规模数据处理中，多线程才能发挥优势。

1. JDK1.8新特性总结。

JDK1.8的最主要新特性主要是：引入了Lambda表达式，函数式接口，流(Stream)以及对集合类型的流化处理，引入了默认方法。

\*什么是Lambda表达式？

Lambda表达式是一种代码即数据的编程方式。形如Operator<Integer>add=(x,y)->x+y;

\*什么是函数式接口？

函数式接口是Lambda表达式的类型，如上述的Operator，它是一个只有一个抽象方法的接口。

函数式接口以’@FunctionalInterface’进行注解，当然，该注解也可以省略。但是一旦有该注解，JVM会强制检查该接口是不是函数式接口。

\*什么是流？什么是惰性求值和及早求值？如何流化？

JDK1.8引入了Stream类，即流。引入流的目的是让我们在操作集合类时能够从现在的外部迭代(生成Iterator对象)改成流中的内部迭代。

惰性求值和及早求值：

所谓惰性求值，即描述Stream但不产生新集合，

所谓及早求值，即从Stream中产生值

如何进行集合类的流化处理？

调用集合类的stream()方法即可完成流化处理。

\*Stream提供了哪些基本方法？

collect(toList())：收集Stream里的值并产生一个列表

map：将Stream中的一个值转化成另一个

filter：过滤掉Stream中的一些值

flatMap：将Stream中的值转换成新的Stream对象

reduce：从一组值中生成一个值，如求和

\*什么是默认方法？为什么要引进默认方法？

默认方法在接口中已经实现，继承即可使用，无需重写，用***default***关键字修饰。

引进默认方法的目的在于新增第三方集合类库打破了二进制兼容性。JDK1.8修改了核心类库，导致新增的第三方集合类不能通过编译，那么直接让第三方集合类继承核心集合类的方法（如stream()）来解决这个问题。

1. java的基础数据类型有哪些？

Java数据类型有两大类：基本数据类型和引用数据类型。

基本数据类型：

一、数值型①整数型：byte(1字节)、short(2)、int(4)、long(8)②浮点型：float(4)、double(8)

二、字符型：char(2)

三、布尔型：boolean

引用数据类型：类、接口、数组

1. 集合有哪些？哪些是线程安全的？

Collection接口的子类，Map接口，Iterator接口。

Collection接口的继承有List，Set，Queue

HashMap，Hashtable，LinkedHashMap和TreeMap实现了Map接口

HashMap

HashMap是最常用的Map，它根据键的HashCode值存储数据，根据键可以直接获取它的值，具有很快的访问速度，遍历时，取得数据的顺序是完全随机的。因为键对象不可以重复，所以HashMap最多只允许一条记录的键为Null，允许多条记录的值为Null，是非同步的

Hashtable

Hashtable与HashMap类似，是HashMap的线程安全版，它支持线程的同步，即任一时刻只有一个线程能写Hashtable，因此也导致了Hashtale在写入时会比较慢，它继承自Dictionary类，不同的是它不允许记录的键或者值为null，同时效率较低。

ConcurrentHashMap

线程安全，并且锁分离。ConcurrentHashMap内部使用段(Segment)来表示这些不同的部分，每个段其实就是一个小的hash table，它们有自己的锁。只要多个修改操作发生在不同的段上，它们就可以并发进行。

LinkedHashMap

LinkedHashMap保存了记录的插入顺序，在用Iteraor遍历LinkedHashMap时，先得到的记录肯定是先插入的，在遍历的时候会比HashMap慢，有HashMap的全部特性。

TreeMap

TreeMap实现SortMap接口，能够把它保存的记录根据键排序，默认是按键值的升序排序（自然顺序），也可以指定排序的比较器，当用Iterator遍历TreeMap时，得到的记录是排过序的。不允许key值为空，非同步的；

所有的集合类，都实现了Iterator接口，这是一个用于遍历集合中元素的接口

集合中，除了vector，stack，hashtable，enumeration，其他均不是线程安全的。

1. 你现在使用JDK的那个版本？该版本有什么新特性

Jdk1.8的主要新特性：1、接口默认方式2、Lambda表达式3、函数式接口4、方法与构造函数引用

Jdk1.7的主要新特性：1、支持二进制变量表示2、switch支持string类型3、Try-with-resource语句4、catch多个异常

1. 有什么线程安全的方法可以替代HashMap？

可以使用ConcurrentHashMap或Hashtable,线程安全总是以牺牲效率为代价的

1. 具体说一说ConcurrentHashMap的实现原理。

ConcurrentHashMap在底层实现时由Segment数组和HashEntry数组构成。Segment是一种可重入锁，充当锁的角色。HashEntry则是用于存储键值对。一个ConcurrentHashMap中会有一个Segment数组，Segment数组中的每个元素又维护一个HashEntry数组，数组中的每个元素是链表结构的。实际上，Segment的存在就是为了细化管理HashMap，进行分段锁而不是全锁。

1. synchronized关键字是否有效率之分？

Synchronized以它所作用的整个代码块为同步对象，实际上这个代码块中只有部分语句需要实现同步，而将整个代码块都作为同步对象显然是一种非常浪费的现象。

1. Tomcat是怎么实现的
2. Servlet的生命周期

|  |
| --- |
| Servlet 生命周期：Servlet 加载--->实例化--->服务--->销毁。  Init()：在Servlet的生命周期中，仅执行一次init()方法。它是在服务器装入Servlet时执行的，负责初始化Servlet对象。可以配置服务器，以在启动服务器或客户机首次访问Servlet时装入Servlet。无论有多少客户机访问Servlet，都不会重复执行init（）。  Service()：它是Servlet的核心，负责响应客户的请求。每当一个客户请求一个HttpServlet对象，该对象的Service()方法就要调用，而且传递给这个方法一个“请求”（ServletRequest）对象和一个“响应”（ServletResponse）对象作为参数。在HttpServlet中已存在Service()方法。默认的服务功能是调用与HTTP请求的方法相应的do功能。  Destroy()： 仅执行一次，在服务器端停止且卸载Servlet时执行该方法。当Servlet对象退出生命周期时，负责释放占用的资源。一个Servlet在运行service()方法时可能会产生其他的线程，因此需要确认在调用destroy()方法时，这些线程已经终止或完成。 |

1. 什么叫做双亲委派模型？工作过程是怎么样的？意义是什么？

(1)BootStrap ClassLoader：启动类加载器，负责加载存放在%JAVA\_HOME%\lib目录中的，或者通被-Xbootclasspath参数所指定的路径中的，并且被java虚拟机识别的(仅按照文件名识别，如rt.jar，名字不符合的类库，即使放在指定路径中也不会被加载)类库到虚拟机的内存中，启动类加载器无法被java程序直接引用。

(2)Extension ClassLoader：扩展类加载器，由sun.misc.Launcher$ExtClassLoader实现，负责加载%JAVA\_HOME%\lib\ext目录中的，或者被java.ext.dirs系统变量所指定的路径中的所有类库，开发者可以直接使用扩展类加载器。

(3)Application ClassLoader：应用程序类加载器，由sun.misc.Launcher$AppClassLoader实现，负责加载用户类路径classpath上所指定的类库，是类加载器ClassLoader中的getSystemClassLoader()方法的返回值，开发者可以直接使用应用程序类加载器，如果程序中没有自定义过类加载器，该加载器就是程序中默认的类加载器。

这里需要注意的是上述三个JDK提供的类加载器虽然是父子类加载器关系，但是没有使用继承，而是使用了组合关系。

从JDK1.2开始，java虚拟机规范推荐开发者使用双亲委派模式(ParentsDelegation Model)进行类加载，其加载过程如下：

(1).如果一个类加载器收到了类加载请求，它首先不会自己去尝试加载这个类，而是把类加载请求委派给父类加载器去完成。

(2).每一层的类加载器都把类加载请求委派给父类加载器，直到所有的类加载请求都应该传递给顶层的启动类加载器。

(3).如果顶层的启动类加载器无法完成加载请求，子类加载器尝试去加载，如果连最初发起类加载请求的类加载器也无法完成加载请求时，将会抛出ClassNotFoundException，而不再调用其子类加载器去进行类加载。

双亲委派 模式的类加载机制的优点是java类它的类加载器一起具备了一种带优先级的层次关系，越是基础的类，越是被上层的类加载器进行加载，保证了java程序的稳定运行。

|  |  |
| --- | --- |
| 机制 | 启动（Bootstrap）类加载器----->标准扩展（Extension）类加载器--->系统（System）类加载器---->上下文(Custom)类加载器  从左到右加载：首先将加载任务委托给父类加载器，依次递归，如果父类加载器可以完成类加载任务，就成功返回；只有父类加载器无法完成此加载任务时，才自己去加载。 |
| 意义 | 防止内存中出现多份同样的字节码  使用委托机制，会递归的向父类查找，如果类A中引用了类B，Java虚拟机将使用加载类A的类加载器去加载类B，如果A加载器已加载类A，那么B使用A的类加载器进行加载时，就不会在加载类A的字节码了 |
| 方法 | 1启动(Bootstrap)类加载器  2标准扩展(Extension)类加载器  3应用程序(Application)类加载器  4上下文(Custom)类加载器 |

1. Java中的BIO,NIO,AIO分别是什么？

IO的方式通常分为几种，同步阻塞的BIO、同步非阻塞的NIO、异步非阻塞的AIO。

Java对BIO、NIO、AIO的支持：

Java BIO:同步并阻塞，服务器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。

Java 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注册到多路复用器上，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。

Java AIO(NIO.2):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。

1. JVM分哪些区？这些区分别是干什么的？

见笔记。

1. JVM如何GC，新生代，老年代，持久代，都存储哪些东西？

根搜索可达算法，GCroot不可达。

GCroot可以是：①虚拟机栈中的引用对象②方法区类的静态属性引用对象③方法区的常量引用对象④本地方法栈中的Native方法。

JVM虚拟机共划分为三个代：年轻代、年老代和持久代。

持久代主要存放的是 Java 类的类信息，与垃圾收集关系不大。

年老代中存放的都是一些生命周期较长的对象。

年轻代和年老代的划分是对垃圾收集影响比较大的。

所有新生成的对象首先都是放在年轻代。年轻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。年轻代又分成三个区。一个 Eden 区，两个 Survivor 区。两个 Survivor 区中总有一个是空的。

1. 线程同步与阻塞的关系？同步一定阻塞吗？阻塞一定同步吗？

在线程层次，或者说操作系统调度单元的层次，操作系统为了减轻程序员的思考负担，将底层的异步非阻塞的IO方式进行封装，把相关系统调用（如read，write等）以同步的方式展现出来。然而，同步阻塞的IO会使线程挂起，同步非阻塞的IO会消耗CPU资源在轮询上。为了解决这一问题，就有3种思路：

多线程（同步阻塞）；

IO多路复用（select，poll，epoll）（同步非阻塞，严格地来讲，是把阻塞点改变了位置）；

直接暴露出异步的IO接口，如kernel-aio和IOCP（异步非阻塞）。

1. HashMap的实现

HashMap利用链地址法解决冲突，具体来说HashMap就是数组加链表，key值经过一次hash如果没有冲突就放在数组的相对位置，否者就借用链地址法解决冲突。在相应位置加上next指针。

1. 什么是泛型，有什么作用？

泛型是Java SE 1.5的新特性，泛型的本质是参数化类型，也就是说所操作的数据类型被指定为一个参数。这种参数类型可以用在类、接口和方法的创建中，分别称为泛型类、泛型接口、泛型方法。 Java语言引入泛型的好处是安全简单。

泛型的好处是在编译的时候检查类型安全，并且所有的强制转换都是自动和隐式的，以提高代码的重用率。避免了强制类型转换带来的代码不安全性。

1. 为什么Java中会出现乱码？

为了让使用Java语言编写的程序能在各种语言的平台下运行，Java在其内部使用Unicode字符集来表示字符，这样就存在Unicode字符集和本地字符集进行转换的过程。当在Java中读取字符数据的时候，需要将本地字符集编码的数据转换为Unicode编码，而在输出字符数据的时候，则需要将Unicode编码转换为本地字符集编码。

在Java语言中，不同字符集编码的转换，都是通过Unicode编码作为中介来完成的。例如，GBK编码的字符“中”要转换为ISO-8859-1（同ISO8859-1）编码，其过程如下：

（1）因为在Java中的字符，都是用Unicode来表示的，所以GBK编码的字符“中”要转换为Unicode表示：0xD6D0->0x4E2D。

（2）将字符“中”的Unicode编码转换为ISO-8859-1编码，因为Unicode编码0x4E2D在ISO-8859-1中没有对应的编码，于是得到0x3f，也就是字符“?”。

# 数据结构

1. 说出你知道的几种排序算法？

插入排序主要包括直接插入排序，折半插入排序和希尔排序两种;

选择排序主要包括直接选择排序和堆排序;

交换排序主要包括冒泡排序和快速排序;

归并排序主要包括二路归并(常用的归并排序)和自然归并。

分配排序主要包括箱排序和基数排序。

其中冒泡,插入,基数,归并属于稳定排序;选择,快速,希尔,堆属于不稳定排序。

快速排序：快速排序（Quicksort）是对冒泡排序的一种改进。它的基本思想是：通过一趟排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一部分的所有数据都要小，然后再按此方法对这两部分数据分别进行快速排序，整个排序过程可以递归进行，以此达到整个数据变成有序序列。

时间复杂度：最好，平均都是O(nlogn),最坏O()

|  |
| --- |
| void qsort(int a[],int l,int r){  int pvt=a[(l+r)/2];  int i=l,j=r;  while(i<=j){  while(a[i]<pvt)  i++;  while(a[j]>pvt)  j--;  if(i<=j){  if(i!=j)  swap(a[i],a[j]);  i++;  j--;  }  }  if(j>l)  qsort(a,l,j);  if(i<r)  qsort(a,i,r);  }  void quick\_sort(int a[],int n){  qsort(a,0,n-1);  } |

1. 请详细说说链表和数组的区别。

数组静态分配内存，链表动态分配内存；

数组在内存中连续，链表不连续；

数组元素在栈区，链表元素在堆区；

数组利用下标定位，时间复杂度为O(1)，链表定位元素时间复杂度O(n)；

数组插入或删除元素的时间复杂度O(n)，链表的时间复杂度O(1)。

1. 双向链表有什么好处，有什么坏处？

双向链表支持前后的顺序访问，但是以牺牲空间为代价

1. 如何判定一个单链表是否有环？

方法一：使用p、q两个指针，p总是向前走，但q每次都从头开始走，对于每个节点，看p走的步数是否和q一样。

方法二：使用p、q两个指针，p每次向前走一步，q每次向前走两步，若在某个时候p == q，则存在环。

1. 常用的Hash算法有哪些？什么事一致性哈希？

SHA，MD5

广义hash:加法Hash；位运算Hash；乘法Hash；除法Hash;查表Hash;混合Hash；

一致性哈希提出了在动态变化的Cache环境中，哈希算法应该满足的4个适应条件：

均衡性，单调性，分散性，负载

1. 什么是平衡二叉树，如何插入节点，删除节点，说出关键步骤

它是一棵空树或它的左右两个子树的高度差的绝对值不超过1，并且左右两个子树都是一棵平衡二叉树。

插入或者删除，导致树不平衡，则需要旋转某些子树。

# 网络基础

1. OSI与TCP/IP各层的结构与功能，都有哪些协议。

**OSI七层模型**

|  |  |  |
| --- | --- | --- |
| 名称 | 功能 | 协议 |
| 应用层 | 文件传输，电子邮件，文件服务，虚拟终端 | TFTP,HTTP,SNMP,FTP,SMTP,DNS,Telnet |
| 表示层 | 数据格式化，代码转换，数据加密 | / |
| 会话层 | 解除或建立与别的节点的联系 | / |
| 传输层 | 提供端对端接口 | TCP,UDP |
| 网络层 | 为数据包选择路由 | IP,ICMP,RIP,OSPF,BGP,IGMP |
| 数据链路层 | 传输有地址的帧以及错误检测功能 | SLIP,CSLIP,PPP,ARP,RARP,MTU |
| 物理层 | 以二进制数据形式在物理媒体上传输数据 | ISO2110,IEEE802,IEEE802.2 |

**TCP/IP五层模型的协议**

|  |  |
| --- | --- |
| 应用层 |  |
| 传输层 | 四层交换机、也有工作在四层的路由器 |
| 网络层 | 路由器、三层交换机 |
| 数据链路层 | 网桥、以太网交换机（二层交换机）、网卡（其实网卡是一半工作在物理层、一半工作在数据链路层） |
| 物理层 | 中继器、集线器、还有我们通常说的双绞线也工作在物理层 |

1. TCP如何保证可靠传输？三次握手过程？

TCP通过3次握手协议保证传输安全。

第一次握手：客户端发送SYN包，进入SYN\_SEND状态

第二次握手：服务器收到SYN包，确认之后发送SYN+ACK包，进入SYN\_RECV状态

第三次握手：客户端收到SYN+ACK包，发送ACK包确认，进入ESTABLISHED状态

1. TCP和UDP区别？

1、TCP面向连接（如打电话要先拨号建立连接）;UDP是无连接的，即发送数据之前不需要建立连接

2、TCP提供可靠的服务。也就是说，通过TCP连接传送的数据，无差错，不丢失，不重复;UDP尽最大努力交付，即不保证可靠交付

3、TCP面向字节流，实际上是TCP把数据看成一连串无结构的字节流;UDP是面向报文的

UDP没有拥塞控制，因此网络出现拥塞不会使源主机的发送速率降低（对实时应用很有用，如IP电话，实时视频会议等）

4、每一条TCP连接只能是点到点的;UDP支持一对一，一对多，多对一和多对多的交互通信

5、TCP首部开销20字节;UDP的首部开销小，只有8个字节

6、TCP的逻辑通信信道是全双工的可靠信道，UDP则是不可靠信道

# 数据库基础

1. 说一说数据库的ACID原则。

ACID，指数据库事务正确执行的四个基本要素的缩写。包含：原子性（Atomicity）、一致性（Consistency）、隔离性（Isolation）、持久性（Durability）。

1. Postgresql有什么好处？与MySQL相比各有什么特点？为什么国内公司大都采用MySQL而非Postgresql？

|  |  |
| --- | --- |
| PostgreSQL | MySQL |
| The world's most advanced open source database | The world's most popular open source database |
| 基于进程结构 | 基于线程结构 |
| 支持OVER字句，解决“每组取Top5”问题 | 不支持 |
| 存储array和json，可以在它们上面建索引 | 不支持 |
| 支持服务器脚本：TCL,Perl… |  |
| 支持地理信息处理扩展（不仅现实世界，也可以游戏中）  有丰富的集合类型，字典、数组、bitmap等类型（因此Instagram采用了pg） |  |
| 自带全文搜索，trigram索引 |  |
| 高效处理树状，图状结构 | 设计复杂，效率低 |
|  |  |
| 稳定性极强 | InnoDB引擎较好，但是MyISAM有所欠缺 |
| 高并发读写的情况下pg性能指标依旧可以维持双曲线，到顶峰后维持不变 | 波峰之后开始下滑 |
| 强悍的SQL变成能力 |  |
|  | 超大内存的情况下，InnoDB可以充分利用内存，pg略差 |

1. 数据库有哪些范式？

第一范式:数据库表的每一列都是不可分割的原子数据项

第二范式:在1NF的基础上，非码属性必须完全依赖于主键

第三范式:在2NF基础上消除传递依赖

BC范式：在3NF基础上消除对主键子集的依赖

1. 事务的隔离级别。

SQL标准定义了四个级别的事务隔离。最严格的是可串行化，是通过标准定义，即保证并发执行和顺序执行的结果相同；其他三个级别是通过现象定义的。

| **隔离级别** | **脏读** | **不可重复读** | **幻读** |
| --- | --- | --- | --- |
| 读未提交（read uncommitted） |  |  |  |
| 读已提交（read committed） | 避免 |  |  |
| 可重复读（repeatable read） | 避免 | 避免 |  |
| 可串行化（serializable） | 避免 | 避免 | 避免 |

**幻读：**重新执行一个查询，由于最近另一个事务的提交，返回的结果（一批数据）和刚才不同；

**不可重复读：** 针对同一个数据，一个事务内多次查询，由于期间另一个事务的提交，导致结果（同一个数据）不同；

**脏读：**一个事务读取了另一个事务还未提交的改动

1. 数据库的MVCC机制

PostgreSQL的：<http://www.jianshu.com/p/04b542aeebac>

<http://blog.csdn.net/pandaxsc1991/article/details/47090337>

1. 数据库中的索引的结构？什么情况下适合建索引？

一般用B+树结构。

1、表的主键、外键必须有索引；

2、数据量超过300的表应该有索引；

3、经常与其他表进行连接的表，在连接字段上应该建立索引；

4、经常出现在Where子句中的字段，特别是大表的字段，应该建立索引；

# 操作系统基础

1. 进程间通信(ICP)方式？

|  |  |
| --- | --- |
| 进程间通信 | 描述 |
| 环境变量/文件描述符 | 子进程接受父进程环境数据的拷贝以及所有文件描述符。父进程可以在它的数据片断或环境中设置一定的变量，同时子进程接收这些值。父进程可以打开文件，同时推进读/写指针的位置，而且子进程使用相同的偏移访问该文件。 |
| 命令行参数 | 在调用exec或派生函数期间，命令行参数可以传递给子进程。 |
| 管道 | 用于相关和无关进程间的通信，而且形成两个进程间的一个通信通道，通常使用文件读写程序访问。 |
| 共享内存 | 使用客户机/服务器模型（C/S），服务器对客户的数据或动作请求作出应。 |
| DDE | 两个进程之外的内存块，两个进程均可以访问它。 |

1. Linux下你常用的命令有哪些?

cd,ls,sudo,apt-get,mv,rm,mkdir,vim

1. 操作系统什么情况下会死锁？

**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：:若干进程之间形成一种头尾相接的循环等待资源关系。

# 大数据及NoSQL

1. Hadoop的组件

HDFS，MapReduce，Hbase，Hive，Zookeeper

1. Redis的使用？

Redis是一个key-value存储系统。和Memcached类似，它支持存储的value类型相对更多，包括string(字符串)、list(链表)、set(集合)、zset(sorted set --有序集合)和hash(哈希类型)。尽可能避免与硬盘IO，从而提高读写速度，有点类似于Java的Map数据结构。

1. 你知道的开源协议有哪些？开源软件有哪些？

Apache Licence 2.0,CPL,GPL,LGPL,BSD,MIT

PostgreSQL-BSD

Kettle- Apache Licence 2.0

Eclipse-CPL

Linux,MySQL-GPL

1. 关系型数据库和非关系型数据库各有什么优缺点？