**Java基础部分**

**&与&&区别？**

**&和&&都是逻辑运算符，都是判断两边同时真则为真，否则为假；但是&&当第一个条件不成之后，后面的条件都不执行了，而&则还是继续执行，直到整个条件语句执行完为止。**

**使用 final 关键字修饰一个变量时，是引用不能变，还是引用的对象不能变？**

**使用 final 关键字修饰一个变量时，是指引用变量不能变，引用变量所指向的对象中的内容 还是可以改变的。**

**--修饰变量，变量不能被修改，修饰方法方法不能被覆盖，修饰类，类不能被继承**

**静态变量和实例变量的区别？**

**在语法定义上的区别：**

**--static修饰的方法或者变量变为类级别的变量，与类同生同死，存储在方法区当中**

**静态变量前要加 static 关键字，而实例变量前则不加。**

**在程序运行时的区别：实例变量属于某个对象的属性，必须创建了实例对象，其中的实例变 量才会被分配空间，才能使用这个实例变量。**

**静态变量不属于某个实例对象，而是属于类， 所以也称为类变量，只要程序加载了类的字节码，不用创建任何实例对象，静态变量就会被分配空间，静态变量就可以被使用了。**

**总之，实例变量必须创建对象后才可以通过这个对象 来使用，静态变量则可以直接使用类名来引用。**

**静态变量使用时,通过类名.名称,实例变量必须要初始化后才能使用。实例变量是实例化后才会分配空间,而静态变量当类加载时会分配空间。**

**是否可以从一个 static 方法内部发出对非 static 方法的调用？**

**不可以。因为非 static 方法是要与对象关联在一起的，必须创建一个对象后，才可以在该对 象上进行方法调用，而 static 方法调用时不需要创建对象，可以直接调用。也就是说，当一 个 static 方法被调用时，可能还没有创建任何实例对象，如果从一个 static 方法中发出对非 static 方法的调用，那个非 static 方法是关联到哪个对象上的呢？这个逻辑无法成立，所以， 一个 static 方法内部发出对非 static 方法的调用。**

**非static方法可以访问static方法.**

**static方法不能访问非static方法**

**"=="和 equals 方法究竟有什么区别？**

**==如果判断值类型的话，判断内容是否相同。如果判断引用类型则是判断内存地址是否相同**

**Equals判断值内容是否相等**

**Integer 与 int 的区别**

**Integer 是引用类型,默认值是null。而int是是值类型默认值是0**

**请说出作用域 public， private， protected，以及不写时的区别**

**这四个作用域的可见范围如下表所示。**

**说明：如果在修饰的元素上面没有写任何访问修饰符，则表示 friendly。**

**作用域 当前类 同一包（ package） 子孙类 其他包（ package）**

**public     √              √                                      √                 √**

**protected √            √                                      √                 ×**

**friendly     √             √                                      ×                ×**

**private      √              ×                                   ×                  ×**

**重载与重写区别？**

**重载是同一个类中，方法名称相同， 但是参数或个数不同。与返回值没有关系。**

**重写是在多个类中， 产生继承关系。父类与子类的方法方法必须相同。**

**接口与抽象类的区别？**

**区别:定义接口的关键字是：interface 而定义抽象类的关键字是：abstract。**

**接口中成员不能有私有， 抽象类可以。**

**接口中定义的成员， 是finl public static 类型， 抽象类没有。**

**接口中的不能有普通方法， 抽象类中可以。**

**相同：**

**两个都不new**

**但是 接口与抽象类是面向对象必备知识，设计模式、重构代码有必然作用**

**final, finally, finalize 的区别。**

**final 用于声明属性，方法和类，分别表示属性不可变，方法不可覆盖，类不可继承。**

**内部类要访问局部变量，局部变量必须定义成 final 类型，例如，一段代码……**

**finally 是异常处理语句结构的一部分，表示总是执行。**

**finalize 是 Object 类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，可**

**以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等。 JVM 不保证此方法总被**

**调用**

**[String、StringBuffer与StringBuilder的区别](http://www.cnblogs.com/yw-ah/p/5841353.html)**

**String 字符串常量**

**StringBuffer 字符串变量（线程安全）**

**StringBuilder 字符串变量（非线程安全）**

**所有的类都继承于object类，你用过的object类的直接子类有哪些，object类常用的方法**

**有哪些**

**1．clone方法**

**保护方法，实现对象的浅复制，只有实现了Cloneable接口才可以调用该方法，否则抛出CloneNotSupportedException异常。**

⑴浅复制（浅克隆）   
  
    被复制对象的所有变量都含有与原来的对象相同的值，而所有的对其他对象的引用仍然指向原来的对象。换言之，浅复制仅仅复制所考虑的对象，而不复制它所引用的对象。   
  
⑵深复制（深克隆）   
  
    被复制对象的所有变量都含有与原来的对象相同的值，除去那些引用其他对象的变量。那些引用其他对象的变量将指向被复制过的新对象，而不再是原有的那些被引用的对象。换言之，深复制把要复制的对象所引用的对象都复制了一遍。

**2．getClass方法**

**final方法，获得运行时类型。**

**3．toString方法**

**该方法用得比较多，一般子类都有覆盖。**

**4．finalize方法**

**该方法用于释放资源。因为无法确定该方法什么时候被调用，很少使用。**

**5．equals方法**

**该方法是非常重要的一个方法。一般equals和==是不一样的，但是在Object中两者是一样的。子类一般都要重写这个方法。**

**6.数组有没有length()这个方法? String有没有length()这个方法？**

**答：数组没有length()这个方法，有length的属性。String有有length()这个方法。**

**7．hashCode方法**

**该方法用于哈希查找，重写了equals方法一般都要重写hashCode方法。这个方法在一些具有哈希功能的Collection中用到。**

**一般必须满足obj1.equals(obj2)==true。可以推出obj1.hash- Code()==obj2.hashCode()，但是hashCode相等不一定就满足equals。不过为了提高效率，应该尽量使上面两个条件接近等价。**

**7．wait方法**

**wait方法就是使当前线程等待该对象的锁，当前线程必须是该对象的拥有者，也就是具有该对象的锁。wait()方法一直等待，直到获得锁或者被中断。wait(long timeout)设定一个超时间隔，如果在规定时间内没有获得锁就返回。**

**调用该方法后当前线程进入睡眠状态，直到以下事件发生。**

**（1）其他线程调用了该对象的notify方法。**

**（2）其他线程调用了该对象的notifyAll方法。**

**（3）其他线程调用了interrupt中断该线程。**

**（4）时间间隔到了。**

**此时该线程就可以被调度了，如果是被中断的话就抛出一个InterruptedException异常。**

**8．notify方法**

**该方法唤醒在该对象上等待的某个线程。**

**9．notifyAll方法**

**该方法唤醒在该对象上等待的所有线程**

**反射的优缺点？**

**反射：就是正在运行动态读取这个类的完整信息。**

**优点：java的反射机制就是增加程序的灵活性、**

**缺点：缺点：（1）性能问题：使用反射基本上是一种解释操作，**

**用于字段和方法接入时要远慢于直接代码。因此反射机制主要应用在对灵活性和扩展性要求很高的系统框架上，普通程序不建议使用。**

**（2）使用反射会模糊程序内内部逻辑：程序员希望在源代码中看到程序的逻辑，反射等绕过了源代码的技术，因而会带来维护问题。反射代码比相应的直接代码更复杂。**

**那些地方用到了反射？**

**例如: jdbc、Java常用框架、jdk的动态代理、android的加载布局文件**

**java  中有几种类型的流？JDK 为每种类型的流提供了一些抽象类以供继承， 为每种类型的流提供了一些抽象类以供继承，**

**请说出他们分别是哪些类？**

   字节流，字符流。字节流继承于 InputStream OutputStream，字符流继承于

InputStreamReaderOutputStreamWriter。在 java.io 包中还有许多其他的流，主要是为了提

高性能和使用方便。

**多线程部分**

**什么是多线程？**

**在一个应用程序中,同时，有多个不同的执行路径。**

**说一下多线程的好处？**

**提供程序效率。**

**线程和进程有什么区别？**

**线程是进程的一条执行路径，而进程是线程的集合。**

**什么是线程同步、异步？**

**线程同步表示，当前线程执行完后下一个线程接着执行。**

**线程异步表示,  在一个应用程序中,同时，有多个不同的执行路径。例如 javaweb ajax android handler**

**线程之间如何同步**

**线程之间同步使用 synchronized、wait 与 notify**

**什么是线程不安全？如何解决？（重点）**

**就是在多个线程共享同一个数据会受到其他线程的干扰。如何解决:使用线程同步技术， 用上锁(synchronized)。 让一个线程执行完了，在让另一个线程执行。**

**如何创建一个线程？有几种方法？**

**继承thread类， 重写run方法、实现Runnalbe接口，重新run方法 ， 启动一个线程用start();**

**是使用Runnalbe接口好？还是继承Thread类好？**

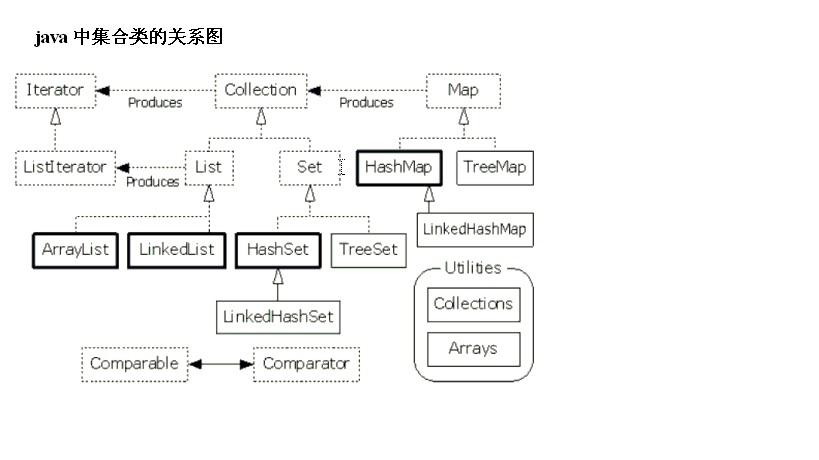
**是实现Runnalbe接口好，因为实现的接口还可以继续继承。如果继承了Thread类不能在继承。**

**sleep()和 wait()有什么区别?**

**a、sleep是让当前线程指定休眠时间，然后继续工作  不释放锁**

**b、让当前线程wait则是等待，直到有线程通知notify（）唤醒他才会重新工作。释放锁**

**集合相关面试题**



**说一下数据结构中的什么是数组？什么是链表？**

**所谓数组，是相同数据类型的元素按一定顺序排列的[集合](http://baike.baidu.com/subview/15216/10703233.htm)**

**数组：存储区间是连续的，占用内存严重，故空间复杂的很大。但数组的二分查找时间复杂度小，为O(1)；数组的特点是：寻址容易，插入和删除困难；**

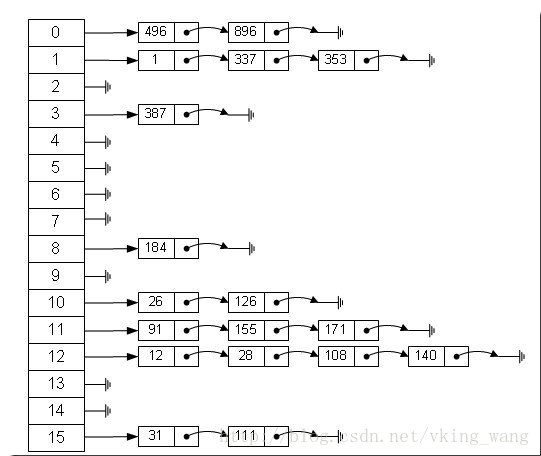
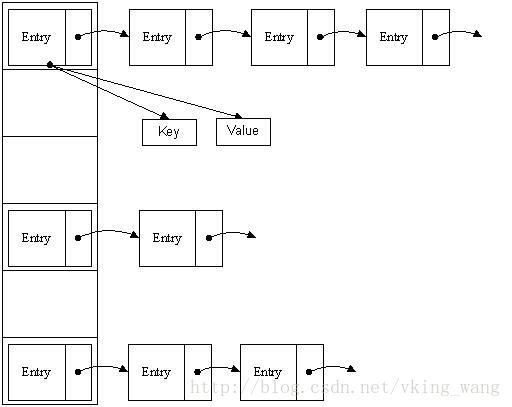
**所谓链表，链表是一种物理[存储单元](http://baike.baidu.com/view/1223079.htm)上非连续、非顺序的[存储结构](http://baike.baidu.com/view/2820182.htm)，[数据元素](http://baike.baidu.com/view/38785.htm)的逻辑顺序是通过链表中的[指针](http://baike.baidu.com/view/159417.htm)链接次序实现的。链表由一系列结点（链表中每一个元素称为结点）组成，结点可以在运行时动态生成。每个结点包括两个部分：一个是存储[数据元素](http://baike.baidu.com/view/38785.htm)的数据域，另一个是存储下一个结点地址的[指针](http://baike.baidu.com/view/159417.htm)域。 相比于[线性表](http://baike.baidu.com/view/178622.htm)[顺序结构](http://baike.baidu.com/view/552878.htm)，操作复杂。由于不必须按顺序存储，链表在插入的时候可以达到O(1)的复杂度，比另一种线性表顺序表快得多，但是查找一个节点或者访问特定编号的节点则需要O(n)的时间，而线性表和顺序表相应的时间复杂度分别是O(logn)和O(1)。**

**链表:链表存储区间离散，占用内存比较宽松，故空间复杂度很小，但时间复杂度很大，达O（N）。链表的特点是：寻址困难，插入和删除容易。**

**说一下什么是哈希表**

**那么我们能不能综合两者的特性，做出一种寻址容易，插入删除也容易的数据结构？答案是肯定的，这就是我们要提起的哈希表。哈希表（(Hash table）既满足了数据的查找方便，同时不占用太多的内容空间，使用也十分方便。**

**哈希表有多种不同的实现方法，我接下来解释的是最常用的一种方法—— 拉链法，我们可以理解为“链表的数组” ，如图：**



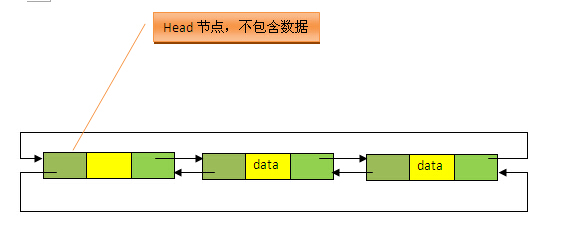
**说一下ArrayList底层实现方式？**

**①ArrayList通过数组实现，一旦我们实例化ArrayList无参数构造函数默认为数组初始化长度为10**

**②add方法底层实现如果增加的元素个数超过了10个，那么ArrayList底层会新生成一个数组，长度为原数组的1.5倍+1，然后将原数组的内容复制到新数组当中，并且后续增加的内容都会放到新数组当中。当新数组无法容纳增加的元素时，重复该过程。是一旦数组超出长度，就开始扩容数组。扩容数组调用的方法 Arrays.copyOf(objArr, objArr.length + 1);**

**说一下LinkedList底层实现方式？**

**LinkedList底层的数据结构是基于双向循环链表的，且头结点中不存放数据,如下：**



**既然是双向链表，那么必定存在一种数据结构——我们可以称之为节点，节点实例保存业务数据，前一个节点的位置信息和后一个节点位置信息，如下图所示：**



**说一下[HashMap](http://blog.csdn.net/vking_wang/article/details/14166593)底层实现方式？**

**HashMap是由数组+链表组成**

**put方法底层实现:**

**通过key的hash值%Entry[].length得到该存储的下标位置，如果多个key的hash值%Entry[].length 值相同话就就会存储到该链表的后面。**

**ArrayList 和 Vector 的区别**

**这两个类都实现了 List 接口(List 接口继承了Collection 接口),他们都是有序集合,即存储在这两个集合中的元素的位置都是有顺序的,相当于一种动态的数组,我们以后可以按位置索引号取出某个元素,并且其中的数据是允许重复的,**

**ArrayList 与 Vector 的区别,这主要包括两个方面:.**

**(1)同步性:**

**Vector 是线程安全的,也就是说是它的方法之间是线程同步的,而 ArrayList 是线程序不安全的,它的方法之间是线程不同步的。如果只有一个线程会访问到集合,那最好是使用 ArrayList,因为它不考虑线程安全,效率会高些;如果有多个线程会访问到集合,那最好是使用 Vector,因为不需要我们自己再去考虑和编写线程安全的代码。**

**(2)数据增长:**

**ArrayList 与 Vector 都有一个初始的容量大小,当存储进它们里面的元素的个数超过了容量时,就需要增加 ArrayList 与 Vector 的存储空间,每次要增加存储空间时,不是只增加一个存储单元,而是增加多个存储单元,每次增加的存储单元的个数在内存空间利用与程序效率之间要取得一定的平衡。Vector 默认增长为原来两倍,而 ArrayList 的增长策略在文档中没有明确规定(从源代码看到的是增长为原来的1.5倍)。**

**ArrayList 与 Vector 都可以设置初始的空间大小,Vector 还可以设置增长的空间大小,而 ArrayList 没有提供设置增长空间的方法。**

**HashMap 和 Hashtable 的区别**

**总结：**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **hashmap** | **线程不安全** | **允许有null的键和值** | **效率高一点、** | **方法不是Synchronize的要提供外同步** | **有containsvalue和containsKey方法** | **HashMap 是Java1.2 引进的Map interface 的一个实现** | **HashMap是Hashtable的轻量级实现** |
| **hashtable** | **线程安全** | **不允许有null的键和值** | **效率稍低、** | **方法是是Synchronize的** | **有contains方法方法** | **、Hashtable 继承于Dictionary 类** | **Hashtable 比HashMap 要旧** |

**List 和Set、Map 区别?**

**Java中的集合包括三大类，它们是Set、List和Map，它们都处于[java.util](https://www.baidu.com/s?wd=java.util&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkPHmvPvm3mHP9PvNBnWF90ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnW0LPj0vrHbkrHDLnjfYrHRsr0)包中，Set、List和Map都是接口，它们有各自的实现类。Set的实现类主要有HashSet和TreeSet，List的实现类主要有ArrayList，Map的实现类主要有HashMap和TreeMap。**

**Set中的对象不按特定方式排序，并且没有重复对象。但它的有些实现类能对集合中的对象按特定方式排序，例如TreeSet类，它可以按照默认排序，也可以通过实现[java.util](https://www.baidu.com/s?wd=java.util&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkPHmvPvm3mHP9PvNBnWF90ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnW0LPj0vrHbkrHDLnjfYrHRsr0).Comparator<Type>接口来自定义排序方式。**

**List中的对象按照索引位置排序，可以有重复对象，允许按照对象在集合中的索引位置检索对象，如通过list.get(i)方式来获得List集合中的元素。**

**Map中的每一个元素包含一个键对象和值对象，它们成对出现。键对象不能重复，值对象可以重复。**

**List、Map、Set 三个接口,存取元素时,各有什么特点?**

**list：存储： 有序的 可重复的**

**访问：可以[for循环](https://www.baidu.com/s?wd=for%E5%BE%AA%E7%8E%AF&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkP1-WPjIbP1fYPynsryfv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHnsnWc1P1fzrHTvnj04nWfkrf)，foreach循环，iterator迭代器 迭代。**

**set：存储：无序的 不重复的**

**访问：可以foreach循环，iterator迭代器 迭代**

**map：存储：存储的是一对一对的映射 ”key=value“，key值 是无序，不重复的。value值可重复**

**访问：可以map中key值转为为set存储，然后迭代这个set，用map.get(key)获取value**

**也可以 转换为entry对象 用迭代器迭代**

**说出 ArrayList,Vector, LinkedList 的存储性能和特性**

**ArrayList和Vector都是使用数组方式存储数据，此数组元素数大于实际存储的数据以便增加和插入元素，它们都允许直接按序号索引元素，但是插入元素要涉及数组元素移动等内存操作，所以索引数据快而插入数据慢，Vector由于使用了synchronized方法（[线程安全](https://www.baidu.com/s?wd=%E7%BA%BF%E7%A8%8B%E5%AE%89%E5%85%A8&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YzPADzuj6kmWf4mvN-uWP-0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWbzrjRsPWc4)），通常性能上较ArrayList差，而LinkedList使用双向链表实现存储，按序号索引数据需要进行前向或后向遍历，但是插入数据时只需要记录本项的前后项即可，所以插入速度较快。**

**去掉一个 Vector 集合中重复的元素**

**通过Vector.contains（）方法判断是否包含该元素，如果没有包含就添加到新的集合当中，适用于数据较小的情况下。**

**Collection 和 Collections 的区别。**

**Collection是集合类的上级接口，继承于它的接口主要有Set和List。 Collections是针对集合类的一个帮助类，它提供了一系列静态方法实现了对各种集合的排序，搜索和线程安全等操作。**

**Set 里的元素是不能重复的,那么用什么方法来区分重复与否呢?是用==还是equals()?它们有何区别?**

**set里的元素是不能重复的，用iterator()方法来区分重复与否。**

**equals 方法（是[String类](https://www.baidu.com/s?wd=String%E7%B1%BB&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWPyc3myFBnvwBuyPhuyD0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtznj0vnHmvnWRdrHRsP1DsrjR)从它的超类Object中继承的）被用来检测两个对象是否相等，即两个对象的内容是否相等。**

**==用于比较引用和比较基本数据类型时具有不同的功能：**

**比较基本数据类型，如果两个值相同，则结果为true**

**而在比较引用时，如果引用指向内存中的同一对象，结果为true**

**HashMap面试题**

**HashMap的工作原理是近年来常见的[Java](http://lib.csdn.net/base/javase)面试题。几乎每个Java程序员都知道HashMap，都知道哪里要用HashMap，知道[Hashtable和HashMap之间的区别](http://www.importnew.com/7010.html)，那么为何这道面试题如此特殊呢？是因为这道题考察的深度很深。这题经常出现在高级或中高级面试中。投资银行更喜欢问这个问题，甚至会要求你实现HashMap来考察你的编程能力。ConcurrentHashMap和其它同步集合的引入让这道题变得更加复杂。让我们开始探索的旅程吧！**

**先来些简单的问题**

**“你用过HashMap吗？” “什么是HashMap？你为什么用到它？”**

**几乎每个人都会回答“是的”，然后回答HashMap的一些特性，譬如HashMap可以接受null键值和值，而Hashtable则不能；HashMap是非synchronized;HashMap很快；以及HashMap储存的是键值对等等。这显示出你已经用过HashMap，而且对它相当的熟悉。但是面试官来个急转直下，从此刻开始问出一些刁钻的问题，关于HashMap的更多基础的细节。面试官可能会问出下面的问题：**

**“你知道HashMap的工作原理吗？” “你知道HashMap的get()方法的工作原理吗？”**

**你也许会回答“我没有详查标准的[Java](http://lib.csdn.net/base/java)API，你可以看看Java源代码或者Open JDK。”“我可以用Google找到答案。”**

**但一些面试者可能可以给出答案，“HashMap是基于hashing的原理，我们使用put(key, value)存储对象到HashMap中，使用get(key)从HashMap中获取对象。当我们给put()方法传递键和值时，我们先对键调用hashCode()方法，返回的hashCode用于找到bucket位置来储存Entry对象。”这里关键点在于指出，HashMap是在bucket中储存键对象和值对象，作为Map.Entry。这一点有助于理解获取对象的逻辑。如果你没有意识到这一点，或者错误的认为仅仅只在bucket中存储值的话，你将不会回答如何从HashMap中获取对象的逻辑。这个答案相当的正确，也显示出面试者确实知道hashing以及HashMap的工作原理。但是这仅仅是故事的开始，当面试官加入一些Java程序员每天要碰到的实际场景的时候，错误的答案频现。下个问题可能是关于HashMap中的碰撞探测(collision detection)以及碰撞的解决方法：**

**“当两个对象的hashcode相同会发生什么？” 从这里开始，真正的困惑开始了，一些面试者会回答因为hashcode相同，所以两个对象是相等的，HashMap将会抛出异常，或者不会存储它们。然后面试官可能会提醒他们有equals()和hashCode()两个方法，并告诉他们两个对象就算hashcode相同，但是它们可能并不相等。一些面试者可能就此放弃，而另外一些还能继续挺进，他们回答“因为hashcode相同，所以它们的bucket位置相同，‘碰撞’会发生。因为HashMap使用链表存储对象，这个Entry(包含有键值对的Map.Entry对象)会存储在链表中。”这个答案非常的合理，虽然有很多种处理碰撞的方法，这种方法是最简单的，也正是HashMap的处理方法。但故事还没有完结，面试官会继续问：**

**“如果两个键的hashcode相同，你如何获取值对象？” 面试者会回答：当我们调用get()方法，HashMap会使用键对象的hashcode找到bucket位置，然后获取值对象。面试官提醒他如果有两个值对象储存在同一个bucket，他给出答案:将会遍历链表直到找到值对象。面试官会问因为你并没有值对象去比较，你是如何确定确定找到值对象的？除非面试者直到HashMap在链表中存储的是键值对，否则他们不可能回答出这一题。**

**其中一些记得这个重要知识点的面试者会说，找到bucket位置之后，会调用keys.equals()方法去找到链表中正确的节点，最终找到要找的值对象。完美的答案！**

**许多情况下，面试者会在这个环节中出错，因为他们混淆了hashCode()和equals()方法。因为在此之前hashCode()屡屡出现，而equals()方法仅仅在获取值对象的时候才出现。一些优秀的开发者会指出使用不可变的、声明作final的对象，并且采用合适的equals()和hashCode()方法的话，将会减少碰撞的发生，提高效率。不可变性使得能够缓存不同键的hashcode，这将提高整个获取对象的速度，使用String，Interger这样的wrapper类作为键是非常好的选择。**

**如果你认为到这里已经完结了，那么听到下面这个问题的时候，你会大吃一惊。“如果HashMap的大小超过了负载因子(load factor)定义的容量，怎么办？”除非你真正知道HashMap的工作原理，否则你将回答不出这道题。默认的负载因子大小为0.75，也就是说，当一个map填满了75%的bucket时候，和其它集合类(如ArrayList等)一样，将会创建原来HashMap大小的两倍的bucket数组，来重新调整map的大小，并将原来的对象放入新的bucket数组中。这个过程叫作rehashing，因为它调用hash方法找到新的bucket位置。**

**如果你能够回答这道问题，下面的问题来了：“你了解重新调整HashMap大小存在什么问题吗？”你可能回答不上来，这时面试官会提醒你当多线程的情况下，可能产生条件竞争(race condition)。**

**当重新调整HashMap大小的时候，确实存在条件竞争，因为如果两个线程都发现HashMap需要重新调整大小了，它们会同时试着调整大小。在调整大小的过程中，存储在链表中的元素的次序会反过来，因为移动到新的bucket位置的时候，HashMap并不会将元素放在链表的尾部，而是放在头部，这是为了避免尾部遍历(tail traversing)。如果条件竞争发生了，那么就死循环了。这个时候，你可以质问面试官，为什么这么奇怪，要在多线程的环境下使用HashMap呢？：）**

**热心的读者贡献了更多的关于HashMap的问题：**

**为什么String, Interger这样的wrapper类适合作为键？ String, Interger这样的wrapper类作为HashMap的键是再适合不过了，而且String最为常用。因为String是不可变的，也是final的，而且已经重写了equals()和hashCode()方法了。其他的wrapper类也有这个特点。不可变性是必要的，因为为了要计算hashCode()，就要防止键值改变，如果键值在放入时和获取时返回不同的hashcode的话，那么就不能从HashMap中找到你想要的对象。不可变性还有其他的优点如线程安全。如果你可以仅仅通过将某个field声明成final就能保证hashCode是不变的，那么请这么做吧。因为获取对象的时候要用到equals()和hashCode()方法，那么键对象正确的重写这两个方法是非常重要的。如果两个不相等的对象返回不同的hashcode的话，那么碰撞的几率就会小些，这样就能提高HashMap的性能。**

**我们可以使用自定义的对象作为键吗？ 这是前一个问题的延伸。当然你可能使用任何对象作为键，只要它遵守了equals()和hashCode()方法的定义规则，并且当对象插入到Map中之后将不会再改变了。如果这个自定义对象时不可变的，那么它已经满足了作为键的条件，因为当它创建之后就已经不能改变了。**

**我们可以使用CocurrentHashMap来代替Hashtable吗？这是另外一个很热门的面试题，因为ConcurrentHashMap越来越多人用了。我们知道Hashtable是synchronized的，但是ConcurrentHashMap同步性能更好，因为它仅仅根据同步级别对map的一部分进行上锁。ConcurrentHashMap当然可以代替HashTable，但是HashTable提供更强的线程安全性。看看[这篇博客](http://javarevisited.blogspot.sg/2011/04/difference-between-concurrenthashmap.html)查看Hashtable和ConcurrentHashMap的区别。**

**我个人很喜欢这个问题，因为这个问题的深度和广度，也不直接的涉及到不同的概念。让我们再来看看这些问题设计哪些知识点：**

**hashing的概念**

**HashMap中解决碰撞的方法**

**equals()和hashCode()的应用，以及它们在HashMap中的重要性**

**不可变对象的好处**

**HashMap多线程的条件竞争**

**重新调整HashMap的大小**

**总结**

**HashMap的工作原理**

**HashMap基于hashing原理，我们通过put()和get()方法储存和获取对象。当我们将键值对传递给put()方法时，它调用键对象的hashCode()方法来计算hashcode，让后找到bucket位置来储存值对象。当获取对象时，通过键对象的equals()方法找到正确的键值对，然后返回值对象。HashMap使用链表来解决碰撞问题，当发生碰撞了，对象将会储存在链表的下一个节点中。 HashMap在每个链表节点中储存键值对对象。**

**当两个不同的键对象的hashcode相同时会发生什么？ 它们会储存在同一个bucket位置的链表中。键对象的equals()方法用来找到键值对。**

**因为HashMap的好处非常多，我曾经在电子商务的应用中使用HashMap作为缓存。因为金融领域非常多的运用Java，也出于性能的考虑，我们会经常用到HashMap和ConcurrentHashMap。你可以查看更多的关于HashMap的文章:**

**请讲下[Java](http://lib.csdn.net/base/javase)里面的容器**

**分两大类，Map和Collection。而Collection又有子接口List(数据存储顺序和插入顺序是一样的)、Set(里面的元素具有唯一性)**

**Map是存储键值对的,里面的健不可以重复,但值可以重复**

**a. 对于List主要有ArrayList和LinkedList两种实现。实现的[数据结构](http://lib.csdn.net/base/datastructure)不同，所以主要的区别也都是和数据结构相关的。 ArrayList基于数组，随机访问快，而对于中间元素的插入删除效率比较低，而且需要考虑扩容问题。LinkedList，则 基于链表，和ArrayList提到的正相反，随机访问慢，但对于中间元素的插入和删除更有效率。**

**Set也是一种Collection，和List比起来主要体现在元素唯一性。**

**请说下Iterator的作用**

**迭代器可以实现Collection接口的方法，可以一个一个地获取集合中的元素**

**在遍历集合时 可判断是否有下一个元素**

**说下ArrayList和LinkedList的区别和联系，并说明什么情况下用它们**

**区别:ArrayList用于对象的随机访问速度快,没有顺序**

**LinkedList实现机制是链表式的,和顺序有关,速度比ArrayList慢**

**联系:ArrayList和LinkedList都是List接口的实现类**

**当要快速获取一个值时,用ArrayList,用于顺序插入操作时,用LinkedList.**

**说下List，Set，Map三种集合各有什么特征**

**List集合中的元素可以重复,**

**Set集合中的元素不可以重复**

**Map集合用键-值映射存放对象,Map容器中的键对象不能重复,值对象可以重复**

**HashSet和TreeSet有什么区别，什么时候用它们**

**区别:HashSet中的元素不能重复,没有顺序**

**TreeSet中的元素不能重复,但有顺序**

**当集合中的元素需要排序时,用TreeSet**

**一般情况下用HashSet,因为不需要排序,速度比TreeSet快**

**什么是泛型，怎么使用的，有什么好处？**

**答案**

**定义一个集合时，可以知道里面定义的是什么类型**

**使用:在集合类型后面加< 数据类型 >**

**使用泛型后,从集合中取得元素后就不用再用强转**

**什么是for each循环，它可以循环那些数据类型**

**答案**

**也可以叫增强型循环，通过对象拿到集合里的值，因为扩展性比较强，建议多使用**

**可以用来循环集合和数组**

**比较下集合和数组的优缺点**

**集合是多个对象的容器,可以将不同数据类型的多个对象组织在一起**

**数组类型是有相同数据类型的数据集合,数组是很多语言都支持的底层数据结构,性能上是最高的**

**HashMap与LinkedHashMap,和TreeMap的区别。**

**共同点：HashMap,LinkedHashMap,TreeMap都属于Map的实现类.**

**不同点: 1.HashMap里面存入的键值对在取出的时候是随机的,**

**2.TreeMap取出来的是排序后的键值对。但如果您要按自然顺序或自定义顺序遍历键，那么TreeMap会更好。**

**3. LinkedHashMap 是HashMap的一个子类，如果需要输出的顺序和输入的相同,那么用LinkedHashMap可以实现.**

**在List里面怎么去掉重复的数？**

**通过把List里面的数据放入HashSet可以去除重复**

**HashMap和ArrayList是不是都是线程不安全的？**

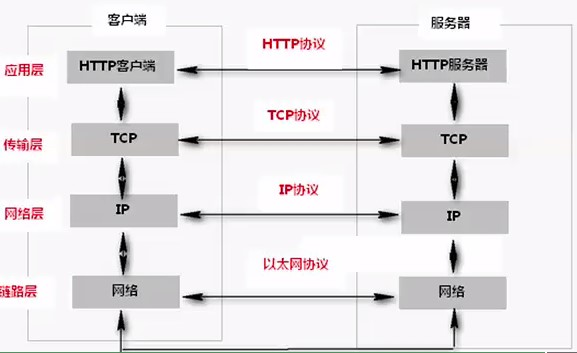
**ArrayList是线程不安全的；HashMap是线程不安全的；还有我们常见的一些JAVA集合都是线程不安全,这样做是为了提高性能**

**在JDK5以后提供了线程安全的并发包java.util.concurrent并发包,譬如里面的类CopyOnWriteArrayList,CopyOnWriteArraySet,ConcurrentHashMap等**

**ArrayList集合加入1万条数据，应该怎么提高效率**

**因为ArrayList的底层是数组实现,并且数组的默认值是10,如果插入10000条要不断的扩容,耗费时间,所以我们调用ArrayList的指定容量的构造器方法ArrayList(int size) 就可以实现不扩容,就提高了性能**

**网路通讯部分**



**Xml与JSON区别**

**数据交换格式**

**区别:**

**xml是重量级、json是轻量级**

**xml比较占带宽、json占带宽小,易于压缩**

**json在webservice 用的比较少、xml用的较多**

**相同：**

**两者都用在项目交互下  例如  移动app接口用的就是json、在web项目中与其他项目对接用xml较多。**

**json常用解析方法 gson、jsonobject、jackson等 xml dom sax pull 解析**

**TCP与UDP区别？**

**udp:  a、是面向无连接, 将数据及源的封装成数据包中,不需要建立建立连接**

**b、每个数据报的大小在限制64k内**

**c、因无连接,是不可靠协议**

**d、不需要建立连接,速度快**

**tcp：  a、建议连接，形成传输数据的通道.**

**b、在连接中进行大数据量传输，以字节流方式**

**c 通过三次握手完成连接,是可靠协议**

**d 必须建立连接m效率会稍低**

**聊天、网络视频会议、桌面共享用的就是  udp**

**说说三次握手?**

**1）第一次握手：建立连接时，客户端A发送SYN包（SYN=j）到服务器B，并进入SYN\_SEND状态，等待服务器B确认。**

**（2）第二次握手：服务器B收到SYN包，必须确认客户A的SYN（ACK=j+1），同时自己也发送一个SYN包（SYN=k），即SYN+ACK包，此时服务器B进入SYN\_RECV状态。**

**（3）第三次握手：客户端A收到服务器B的SYN＋ACK包，向服务器B发送确认包ACK（ACK=k+1），此包发送完毕，客户端A和服务器B进入ESTABLISHED状态，完成三次握手。**

**完成三次握手，客户端与服务器开始传送数据。**

**什么是Webserivce?**

**Webservice就是提供不同的平台相互通讯，基于Soap协议。**

**Web service 就是一个应用程序，它向外界暴露出一个能够通过Web进行调用的API。**

**SOAP是一种简单基于xml的轻量协议，用户web上交换结构化信息和类型信息。**

**soap请求是HTTP POST的一个专用版本，遵循一种特殊的xml消息格式Content-type设置为: text/xml任何数据都可以xml化。**

**WebService实现原理是？**

**HTTP协议+XML**

**说一下什么是Http协议？**

**对器客户端和 服务器端之间数据传输的格式规范，格式简称为“超文本传输协议”。**

**什么是Http协议无状态协议？怎么解决Http协议无状态协议？（曾经去某创业公司问到）**

**1、无状态协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息**

**2、无状态协议解决办法： 通过1、Cookie 2、通过Session会话保存。**

**说一下Http协议中302状态(阿里经常问)**

**http协议中，返回状态码302表示重定向。**

**这种情况下，服务器返回的头部信息中会包含一个 Location 字段，内容是重定向到的url**

**Http协议有什么组成？**

**请求报文包含三部分：**

**a、请求行：包含请求方法、URI、HTTP版本信息**

**b、请求首部字段**

**c、请求内容实体**

**响应报文包含三部分：**

**a、状态行：包含HTTP版本、状态码、状态码的原因短语**

**b、响应首部字段**

**c、响应内容实体**

**Http协议中有那些请求方式？**

**GET： 用于请求访问已经被URI（统一资源标识符）识别的资源，可以通过URL传参给服务器**

**POST：用于传输信息给服务器，主要功能与GET方法类似，但一般推荐使用POST方式。**

**PUT： 传输文件，报文主体中包含文件内容，保存到对应URI位置。**

**HEAD： 获得报文首部，与GET方法类似，只是不返回报文主体，一般用于验证URI是否有效。**

**DELETE：删除文件，与PUT方法相反，删除对应URI位置的文件。**

**OPTIONS：查询相应URI支持的HTTP方法。**

**Http协议中Http1.0与1.1区别？**

**在http1.0中，当建立连接后，客户端发送一个请求，服务器端返回一个信息后就关闭连接，当浏览器下次请求的时候又要建立连接，显然这种不断建立连接的方式，会造成很多问题。**

**5.在http1.1中，引入了持续连接的概念，通过这种连接，浏览器可以建立一个连接之后，发送请求并得到返回信息，然后继续发送请求再次等到返回信息，也就是说客户端可以连续发送多个请求，而不用等待每一个响应的到来。**

**Http协议实现原理机制？**

**2.1、整个流程步骤**



**2.2、域名解析过程**



**2.3、三次握手过程**

**2.4、发起HTTP请求**

**2.5、响应HTTP请求并得到HTML代码**

**2.6、浏览器解析HTML代码**

**2.7、浏览器对页面进行渲染呈现给用户**

**get与post请求区别？（初级程序员必备问题）**

**区别一：**

**get重点在从服务器上获取资源，post重点在向服务器发送数据；**

**区别二：**

**get传输数据是通过URL请求，以field（字段）= value的形式，置于URL后，并用"?"连接，多个请求数据间用"&"连接，如http://127.0.0.1/Test/login.action?name=admin&password=admin，这个过程用户是可见的；**

**post传输数据通过Http的post机制，将字段与对应值封存在请求实体中发送给服务器，这个过程对用户是不可见的；**

**区别三：**

**Get传输的数据量小，因为受URL长度限制，但效率较高；**

**Post可以传输大量数据，所以上传文件时只能用Post方式；**

**区别四：**

**get是不安全的，因为URL是可见的，可能会泄露私密信息，如密码等；**

**post较get安全性较高；**

**区别五：**

**get方式只能支持ASCII字符，向服务器传的中文字符可能会乱码。**

**post支持标准字符集，可以正确传递中文字符。**

**9、Http请求报文与响应报文格式？**

**请求报文包含三部分：**

**a、请求行：包含请求方法、URI、HTTP版本信息**

**b、请求首部字段**

**c、请求内容实体**

**响应报文包含三部分：**

**a、状态行：包含HTTP版本、状态码、状态码的原因短语**

**b、响应首部字段**

**c、响应内容实体**

**10、常见Http协议状态？**

**200：请求被正常处理**

**204：请求被受理但没有资源可以返回**

**206：客户端只是请求资源的一部分，服务器只对请求的部分资源执行GET方法，相应报文中通过Content-Range指定范围的资源。**

**301：永久性重定向**

**302：临时重定向**

**303：与302状态码有相似功能，只是它希望客户端在请求一个URI的时候，能通过GET方法重定向到另一个URI上**

**304：发送附带条件的请求时，条件不满足时返回，与重定向无关**

**307：临时重定向，与302类似，只是强制要求使用POST方法**

**400：请求报文语法有误，服务器无法识别**

**401：请求需要认证**

**403：请求的对应资源禁止被访问**

**404：服务器无法找到对应资源**

**500：服务器内部错误**

**503：服务器正忙**

**Http协议首部字段？**

**a、通用首部字段（请求报文与响应报文都会使用的首部字段）**

**Date：创建报文时间**

**Connection：连接的管理**

**Cache-Control：缓存的控制**

**Transfer-Encoding：报文主体的传输编码方式**

**b、请求首部字段（请求报文会使用的首部字段）**

**Host：请求资源所在服务器**

**Accept：可处理的媒体类型**

**Accept-Charset：可接收的字符集**

**Accept-Encoding：可接受的内容编码**

**Accept-Language：可接受的自然语言**

**c、响应首部字段（响应报文会使用的首部字段）**

**Accept-Ranges：可接受的字节范围**

**Location：令客户端重新定向到的URI**

**Server：HTTP服务器的安装信息**

**d、实体首部字段（请求报文与响应报文的的实体部分使用的首部字段）**

**Allow：资源可支持的HTTP方法**

**Content-Type：实体主类的类型**

**Content-Encoding：实体主体适用的编码方式**

**Content-Language：实体主体的自然语言**

**Content-Length：实体主体的的字节数**

**Content-Range：实体主体的位置范围，一般用于发出部分请求时使用**

**Http与Https优缺点？**

**a、通信使用明文不加密，内容可能被窃听，也就是被抓包分析。**

**b、不验证通信方身份，可能遭到伪装**

**c、无法验证报文完整性，可能被篡改**

**HTTPS就是HTTP加上加密处理（一般是SSL安全通信线路）+认证+完整性保护**

**Http优化**

**[利用负载均衡优化和加速HTTP应用](http://virtualadc.blog.51cto.com/3027116/580832/)**

**[利用HTTP Cache来优化网站](http://www.cnblogs.com/cocowool/archive/2011/08/22/2149929.html)**

**Http协议有那些特征？**

**1、支持客户/服务器模式；2、简单快速；3、灵活；4、无连接；5、无状态；**

**如果你还对Http协议不熟悉的话，请参考Http协议<http://www.itmayiedu.com/front/articleinfo/49.html>文章**

**JavaWeb基础部分**

**讲下Servlet的执行流程。doGet和doPost的区别**

**Servlet的执行流程也就是servlet的生命周期，当服务器启动的时候生命周期开始，然后通过init()《启动顺序根据web.xml里的startup-on-load来确定加载顺序》方法初始化servlet，再根据不同请求调用doGet或doPost方法，最后再通过destroy()方法进行销毁。**

**doGet和doPost都是接受用户请求的方法，doGet处理get请求，doPost处理post请求，doGet用于地址栏提交，doPost用于表单提交，在页面提交数据时，get的数据大小有限制4k，post没有限制，get请求提交的数据会在地址栏显示，post不显示，所以post比get安全.**

**当service有一个实例变量，doGet和doPost去调用这个变量，会出现什么问题，你是如何解决的。**

**会出现线程不安全问题。无论是doGet还是doPost去调用，服务器端处理的过程都是一样的，那么我们可以把处理过程单独写在另外一个方法handle里，让两个方法都去调用handle，根据不同请求去调用不同的方法。**

**如何处理servlet的线程不安全问题**

**线程安全就是多线程操作同一个对象不会有问题，线程同步一般来保护线程安全，所以可以在Servlet的线程里面加上同步方法或同步块。（Synchronized）可以保证在同一时间只有一个线程访问，（使用同步块会导致性能变差，最好不去使用实例变量）**

**Jsp的重定向和转发的流程有什么区别**

**重定向是客户端行为，转发是服务器端行为**

**重定向时服务器产生两次请求，转发产生一次请求，重定向时可以转发到项目以外的任何网址，转发只能在当前项目里转发**

**重定向会导致request对象信息丢失。转发则不会**

**转发的url不会变,request.getRequestDispatch（）。forward()**

**重定向的url会改变,response.getRedirect();**

**Jsp和servlet的区别**

**jsp的可读性强，容易维护，并且jsp在最后会编译成servlet**

**servlet容易调试**

**Jsp的九大内置对象，三大指令，七大动作的具体功能**

**JSP九大内置对象:**

**pageContext ：只对当前jsp页面有效，里面封装了基本的request和session的对象  
Request ：对当前请求进行封装  
Session ：浏览器会话对象，浏览器范围内有效  
Application ：应用程序对象，对整个web工程都有效  
Out ：页面打印对象，在jsp页面打印字符串  
Response ：返回服务器端信息给用户  
Config ：单个servlet的配置对象，相当于servletConfig对象  
Page ：当前页面对象，也就是this  
Exception ：错误页面的exception对象，如果指定的是错误页面，这个就是异常对象**

**三大指令：**

**Page ：指令是针对当前页面的指令  
Include ：用于指定如何包含另一个页面  
Taglib ：用于定义和指定自定义标签**

**七大动作：**

**Forward，执行页面跳转，将请求的处理转发到另一个页面  
Param ：用于传递参数  
Include ：用于动态引入一个jsp页面  
Plugin ：用于下载javaBean或applet到客户端执行  
useBean ：使用javaBean  
setProperty ：修改javaBean实例的属性值  
getProperty ：获取javaBean实例的属性值**

**获取页面的元素和值有几种方式，分别说一下**

**request.getParameter() 返回客户端的请求参数与值  
  
request.getParameterNames() 返回所有可用属性名的枚举  
  
request.getParameterValues() 返回包含参数的所有值的数组**

**servlet和javaScript的区别，他们分别是什么作用**

**一个是服务端，一个是客户端**

**Servlet是独立于平台和协议的服务器端的java应用程序，可以动态生成web页面，并采用响应--请求的模式提供web服务**

**javaScript是一种解释性语言，用于向html页面提供交互行为，通常被直接嵌入在html页面中**

**servlet是java语言编写的web应用**

**js是基于html上的一种解释语言**

**会话跟踪有哪些，他们的区别是什么**

**Cookie，session和application，  
Cookie是http对象，客户端与服务端都可以操纵**

**cookie是在客户端保持状态，session是在服务器端保持状态，由于cookie是保存在客户端本地的，所以数据很容易被窃取，当访问量很多时，使用session则会降低服务器的性能，application的作用域是整个工程里只有一个，可以在不同浏览器之间共享数据，所有人都可以共享，因此application也是不安全的**

**说说jsp的隐藏对象有哪些**

**Request,out,response , pageContext , session , application , config , page , exception，也即jsp的九大内置对象**

**request ，response，session 和 application是怎么用的**

**Request是客户端向服务端发送请求**

**Response是服务端对客户端请求做出响应**

**Session在servlet中不能直接使用，需要通过getSession()创建，如果没有设定它的生命周期，或者通过invildate()方法销毁，关闭浏览器session就会消失**

**Application不能直接创建，存在于服务器的内存中，由服务器创建和销毁**

**jsp页面跳转**

**Jsp页面跳转有两种方式，forward和redirect（转发和重定向）**

**Forward只能在当前项目里跳转，只产生一次请求，request保存的变量不会丢失，url地址不会改变**

**Redirect可跳转到项目以外的任何页面，产生两次请求，request保存的变量会全部丢失，url地址会发生改变，变化为第二个请求的地址**

话跟踪

**如果创建servlet实例不用构造方法，怎么创建一个servlet实例**

**Web容器会自动为servlet写一个无参的构造器，它使用class.forName("").newInstance()反射来创建servlet实例的**

**Servlet是安全的吗？当service有一个实例变量，doGet和doPost去调用这个变量，会出现什么问题，你是如何解决的**

**是线程不安全的，因为servlet是单例模式，当多个客户端共同访问的时候线程不安全。**

**尽量用局部变量，同步块，如果当前字段是不会改变的，用final修饰**

**Java框架部分**

**说说Spring？**

**Spring的核心是控制反转、依赖注入,Aop（面向切面）相当于把每个bean与bean之间的关系交给第 三方容器进行管理.**

**说SpringIOC、SpringAOP?**

**SpringIOC ,其实就是依赖注入、控制反转。相当于把每个bean与bean之间的关系交给第三方容器管理。而这个容器就是spring**

**SpringAOP 面向切面的编程，或AOP，是一种编程技术，允许程序模块化横向切割关注点，或横切典型的责任划分，如日志和事务管理。 SpringAop 就是用 Javva的动态代理**

**Spring的底层实现机制是什么？**

**使用Demo4j（解析XML）+Java反射机制**

**Demo4j 其实就是解析XML。使用反射机制实例化bean。**

**SpringAOP用到了什么代理？**

**JDK动态代理:对实现了接口的类生成代理**

**CGLib代理机制:对类生成代理**

**动态代理与静态代理区别？**

**静态代理：由程序员创建或特定工具自动生成源代码，再对其编译。在程序运行前，代理类的.class文件就已经存在了。   
动态代理：在程序运行时，运用反射机制动态创建而成。**

**Spring注入有那些方式？**

**Set注入**

**构造器注入**

**静态工厂的方法注入**

**实例工厂的方法注入**

**Spring有那些注解？**

**@Autowired(按类型注入)**

**@Service(标示为注入为服务层)**

**@Resource(按名称注入)**

**@Controller(标识控制器bean id)**

**@RequestMapping（表示映射URL路径）**

**简述Spring的优缺点？**

**Spring 的优点？?**

**1.降低了组件之间的耦合性 ，实现了软件各层之间的解耦**

**2.可以使用容易提供的众多服务，如事务管理，消息服务等**

**3.容器提供单例模式支持**

**4.容器提供了AOP技术，利用它很容易实现如权限拦截，运行期监控等功能**

**5.容器提供了众多的辅助类，能加快应用的开发**

**6.spring对于主流的应用框架提供了集成支持，如hibernate，JPA，Struts等**

**7.spring属于低侵入式设计，代码的污染极低**

**8.独立于各种应用服务器**

**9.spring的DI机制降低了业务对象替换的复杂性**

**10.Spring的高度开放性，并不强制应用完全依赖于Spring，开发者可以自由选择spring的部分或全部**

**缺点:**

**使用到了大量反射机制。反射机制非常占内存，**

**SpringMVC工程流程**

**1. 用户向服务器发送请求，请求被Spring 前端控制Servelt DispatcherServlet捕获；**

**2. DispatcherServlet对请求URL进行解析，得到请求资源标识符（URI）。然后根据该URI，调用HandlerMapping获得该Handler配置的所有相关的对象（包括Handler对象以及Handler对象对应的拦截器），最后以HandlerExecutionChain对象的形式返回；**

**3. DispatcherServlet 根据获得的Handler，选择一个合适的HandlerAdapter。（附注：如果成功获得HandlerAdapter后，此时将开始执行拦截器的preHandler(...)方法）**

**4.  提取Request中的模型数据，填充Handler入参，开始执行Handler（Controller)。 在填充Handler的入参过程中，根据你的配置，Spring将帮你做一些额外的工作：**

**HttpMessageConveter： 将请求消息（如Json、xml等数据）转换成一个对象，将对象转换为指定的响应信息**

**数据转换：对请求消息进行数据转换。如String转换成Integer、Double等**

**数据根式化：对请求消息进行数据格式化。 如将字符串转换成格式化数字或格式化日期等**

**数据验证： 验证数据的有效性（长度、格式等），验证结果存储到BindingResult或Error中**

**5.  Handler执行完成后，向DispatcherServlet 返回一个ModelAndView对象；**

**6.  根据返回的ModelAndView，选择一个适合的ViewResolver（必须是已经注册到Spring容器中的ViewResolver)返回给DispatcherServlet ；**

**7. ViewResolver 结合Model和View，来渲染视图**

**8. 将渲染结果返回给客户端。**

**SpringMVC工作流程描述**

**为什么SpringMVC只使用一个Servlet(DispatcherServlet)来处理所有请求？**

**详细见J2EE设计模式-前端控制模式**

**Spring为什么要结合使用HandlerMapping以及HandlerAdapter来处理Handler?**

**符合面向对象中的单一职责原则，代码架构清晰，便于维护，最重要的是代码可复用性高。如HandlerAdapter可能会被用于处理多种Handler。**

**Hibernate面试题**

**什么是Hibernate？**

**hibernate是一个基于ORM持久框架,可以让程序员以面向对象的思想操作数据库,提高生产效率.**

**什么是ORM？**

**orm不过是一种思想，对象关系映射。是对象关系模型，如hibernate,让你以面向对象的方式去编程。封装了JDBC.**

**说一下orm与jdbc的区别？**

**jdbc只是一个java操作数据库的规范接口而已**

**orm不过是一种思想，对象关系映射。**

**ORM：是对象关系模型，如hibernate,让你以面向对象的方式去编程。封装了JDBC.**

**JDBC：是从底层访问数据库服务器。一般银行，金融行业为了安全起见，直接用JDBC访问**

**Hibernate中get和load有什么不同之处?**

**load ：找不到数据的话会抛出org.hibernate.ObjectNotFoundException异常。此时hibernate会使用延迟加载加载机制**

**get找不到的话会返回null。**

**如果查询不到数据，get 会返回 null,但是不会报错, load 如果查询不到数据，则报错ObjectNotFoundException**

**使用get 去查询数据，(先到一级/二级)会立即向db发出查询请求(select ...), 如果你使用的是 load查询数据，(先到一级、二级))即使查询到对象，返回的是一个代理对象,如果后面没有使用查询结果，它不会真的向数据库发select ,当程序员使用查询结果的时候才真的发出select ,这个现象我们称为懒加载(lazy)**

**hibernate的三种状态？**

**在Hibernate中，对象有三种状态：临 时状态(Transient)、持久状态(Persistent)和游离状态(Detached)。**

**处于持久态的对象也称为 PO(PersistenceObject),临时对象和游离对象也称为VO(ValueObject).**

**hibernate的懒加载？ 有几种禁用方法**

**在Hibernate框架中，当我们要访问的数据量过大时，明显用缓存不太合适， 因为内存容量有限 ，为了减少并发量，减少系统资源的消耗，这时Hibernate用懒加载机制来弥补这种缺陷，但是这只是弥补而不是用了懒加载总体性能就提高了。**

**我们所说的懒加载也被称为延迟加载，它在查询的时候不会立刻访问数据库，而是返回代理对象，当真正去使用对象的时候才会访问数据库。**

**1.使用代理对象：Hibernate.initialize("代理对象");**

**2.在需要禁用懒加载的映射文件中显示的加入lazy = "false"**

**3.使用openSessionInView【需要借助于过滤器】 需要在web.xml文件中配置**

**hibernate有几种查询方式？**

**1、 属性查询2、 参数查询、命名参数查询3、 关联查询4、 分页查询5、 统计函数**

**Hibernate的优缺点？**

**1.Hibernate的优缺点：  
优点：1、程序更加面向对象；  
2、提高了生产率；  
3、方便移植（修改配置文件）；  
4、无侵入性。  
缺点：  
1、效率比JDBC略差；  
2、不适合批量操作。**

**Hibernate的缓存机制**

**Hibernate缓存包括两大类：Hibernate一级缓存和Hibernate二级缓存。**

**1.Hibernate一级缓存又称为“Session的缓存”。**

**Session内置不能被卸载，Session的缓存是事务范围的缓存（Session对象的生命周期通常对应一个数据库事务或者一个应用事务）。**

**一级缓存中，持久化类的每个实例都具有唯一的OID。**

**2.Hibernate二级缓存又称为“SessionFactory的缓存”。**

**由于SessionFactory对象的生命周期和应用程序的整个过程对应，因此Hibernate二级缓存是进程范围或者集群范围的缓存，有可能出现并发问题，因此需要采用适当的并发访问策略，该策略为被缓存的数据提供了事务隔离级别。**

**第二级缓存是可选的，是一个可配置的插件，默认下SessionFactory不会启用这个插件。**

**Hibernate提供了org.hibernate.cache.CacheProvider接口,它充当缓存插件与Hibernate之间的适配器。**

**Hibernate延迟加载？**

**1) Hibernate2延迟加载实现：a)实体对象 b)集合（Collection）**

**2) Hibernate3 提供了属性的延迟加载功能 当Hibernate在查询数据的时候，数据并没有存在与内存中，当程序真正对数据的操作时，对象才存在与内存中，就实现了延迟加载，他节省了服务器的内存开销，从而提高了服务器的性能。**

**Hibernate工作原理及为什么要用？**

**原理：**

**1)        读取并解析配置文件**

**2)        读取并解析映射信息**

**3)        创建SessionFactory**

**4)        打开Sesssion**

**5)        创建事务Transation**

**6)        持久化操作**

**7)        提交事务**

**8)        关闭Session**

**9)        关闭SesstionFactory**

**为什么要用：**

**1)        对JDBC访问数据库的代码做了封装，大大简化了数据访问层繁琐的重复性代码。**

**2)        Hibernate是一个基于JDBC的主流持久化框架，是一个优秀的ORM实现。他很大程度的简化DAO层的编码工作**

**3)        hibernate使用Java反射机制，而不是字节码增强程序来实现透明性。**

**4)        hibernate的性能非常好，因为它是个轻量级框架。映射的灵活性很出色。它支持各种关系数据库，从一对一到多对多的各种复杂关系。**

**什么是Mybatis？**

**Mybatis的前生是ibatis，最后升级版本后名称叫mybatis。mybatis是以纯sql操作数据。**

**Mybatis与Hibernate区别？**

**Hibernate是面向对象的思想操作数据生成Sql语句，而mybatis是以纯sql操作数据**

**相对于mybatis容易优化.扩展性好，但是移植性差。**

**设计模式部分**

**你熟悉那些设计模式？**

**总共有23种设计模式**

**总体来说设计模式分为三大类：**

**创建型模式，共五种：工厂方法模式、抽象工厂模式、单例模式、建造者模式、原型模式。**

**结构型模式，共七种：适配器模式、装饰器模式、代理模式、外观模式、桥接模式、组合模式、享元模式。**

**行为型模式，共十一种：策略模式、模板方法模式、观察者模式、迭代子模式、责任链模式、命令模式、备忘录模式、状态模式、访问者模式、中介者模式、解释器模式。**

**企业面试的时候，面试官非常喜欢考单例，而且非常喜欢考手写单例。**

**什么是单例？单例有那些写法？**

**单例分类:懒汉式单例、饿汉式单例**  
**单例模式有以下特点：**  
**1、单例类只能有一个实例。  
2、单例类必须自己创建自己的唯一实例。  
3、单例类必须给所有其他对象提供这一实例。**

**①懒汉式单例**

1. //懒汉式单例类.在第一次调用的时候实例化自己
2. **public** **class** Singleton {
3. **private** Singleton() {}
4. **private** **static** Singleton single=**null**;
5. //静态工厂方法
6. **public** **static** Singleton getInstance () {
7. **if** (single == **null**) {
8. single = **new** Singleton();
9. }
10. **return** single;
11. }
12. }

**②饿汉式单例**

1. **//饿汉式单例类.在类初始化时，已经自行实例化**
2. **public class Singleton1 {**
3. **private Singleton1() {}**
4. **private static final Singleton1 single = new Singleton1();**
5. **//静态工厂方法**
6. **public static Singleton1 getInstance() {**
7. **return single;**
8. **}**
9. **}**

**懒汉式与饿汉式区别？**

**从名字上来说，饿汉和懒汉，**

**饿汉就是类一旦加载，就把单例初始化完成，保证getInstance的时候，单例是已经存在的了，**

**而懒汉比较懒，只有当调用getInstance的时候，才回去初始化这个单例。**

**另外从以下两点再区分以下这两种方式：**

**1、线程安全：**

**饿汉式天生就是线程安全的，可以直接用于多线程而不会出现问题，**

**懒汉式本身是非线程安全的，为了实现线程安全有几种写法，分别是上面的1、2、3，这三种实现在资源加载和性能方面有些区别。**

**2、资源加载和性能：**

**饿汉式在类创建的同时就实例化一个静态对象出来，不管之后会不会使用这个单例，都会占据一定的内存，但是相应的，在第一次调用时速度也会更快，因为其资源已经初始化完成，**

**而懒汉式顾名思义，会延迟加载，在第一次使用该单例的时候才会实例化对象出来，第一次调用时要做初始化，如果要做的工作比较多，性能上会有些延迟，之后就和饿汉式一样了。**

**项目相关面试题**

**你说说你做的最好的项目？**

**要重点介绍到①项目是做什么？②用到那些技术？③整个项目中最大的亮点是？核心部分④遇到bug是怎么解决的？**

**例如： 我认为我做的最好的项目是**

**创办了蚂蚁课堂it在线教学网站，蚂蚁课堂在线教育网站，类似（慕课网、我要自学网）是一家为IT爱好者提供免费教学服务网站，分为首页、课程、博客、问答四大专区，自己也录制一些Java教学视频发布在蚂蚁课堂网站上。目前国内各大搜索引擎都可以通过搜索“蚂蚁课堂”关键字找到蚂蚁课堂网站，目前日均PV5000-7000等**

**使用技术SpringMVC+Mybatis+Spring+Maven，视频接口才第三方乐视云接口，使用反向代理nginx+tomcat实现负载均衡及集群减轻单台服务器压力、使用CDN加速提高网络分发。………**

**你项目遇到bug？怎么查问题？**

**例如:首先遇到了bug，会查询日志，通过日志定位到某个类的行数，判断是否有代码问题。**

**你遇到了什么bug？你是怎么解决？**

**例如我自己项目中，查询量非常大。通过日志发现了堆内存溢出，最后通过优化代码，减轻new和加大堆内存。**

**你们项目人员是怎么分配的？**

**项目分配为:**

**产品经理(负责提需求)**

**UI设计师（负责设计样式文件）**

**Web前端（只做Web页面前端静态文件）**

**Java工程师（写业务逻辑）**

**测试人员(负责测试bug)**

**非技术项目经理（负责项目管理、人员分配）**

**项目架构师（负责架构项目）**

**你们项目是怎么发布的？**

**企业当中项目都是发布在linux环境上**

**小公司:使用maven编译好通过打war包,放入到tomcat的webapps文件下**

**大公司:使用自动部署系统jenkins jenkins直接关联svn地址自动打包、自动部署等**

**说说linux常用命令?**

**Cat 查看某个文件**

**CP 拷贝**

**Ls 查看所有列表**

**Pwd 查看路径**

**Tail 查看日志**

**Grep 搜索日志**

**以后大家用不懂的，可以上蚂蚁课堂网站，提供更多免费的IT在线教育网站。**

**[http://www.itmayiedu.com](http://www.itmayiedu.com/)**