Java 求职面试指南

目录

第一部分　应聘求职

什么时候适合找工作？在网上搜索的话，会有很多人回复说九月是毕业生进入社会的高峰期，金九银十也是换工作的的黄金时期。貌似现在什么东西都能和金九银十扯上关系，房产、旅游、找工作等等；我感觉毕业生如果等到金九银十再去找工作，估计很难找到合适的了。现在的大学生貌似六七月毕业，而很多学校最后半学期就是让学生做毕业设计和找工作的时间，春节到校招生的企业也是最多的。并不能否定九月十月不适合找工作，但建议找工作要趁早，如果你准备好了，什么时候都可以找工作，而且越早越好。

对于社会有工作经验的朋友来说，好多人可能认为年初最适合找工作，因为拿了年终，而且好多企业会在年初提供很多职位；但我感觉年初不适合找工作，而适合入职，最好在年底把新的工作搞定。

去年年末对于 51JOB,智联，猎聘等专业的招聘网站提供的职位进行了分析，年底的职位可能没有年初多，但从待遇福利和公司规模来看，都很不错。而且从周围的朋友来看，过一个春节，回家放松得很彻底，回来两三周才找到感觉，这个时候去面试，成功率低很多。当然上面多是个人建议，工作不爽了，随时可以找；尤其是ＩＴ行业，在一家公司呆的越久，与新技术脱离的越远，找一个满意的工作对自己和对公司来说，都是有利的。

第 01 章　简历篇

如果你是一个刚毕业的大学生，建议可以到 51job 或其它网络上了解一下简历的注意事项，上面讲的很细致，这里就懒得把那么多细节拉下来占用篇幅，请大家一定要慎重对待简历，好的简历就是一次面试机会，如果你的简历没有写好，那么接下来的内容看了也无用武之地了，这里简单介绍一下简历的注意事项，在附录中，我会放上一个简历模板供大家参考使用。

１、基本信息简单明确

像**姓名、性别、学校专业、学历、电话、电子邮箱、应聘职位**这些是必须的，绝对不能被简洁的，至于出生年月，政治面貌之类的可以选择；薪资和地址之类的个人建议没有必要写，如果写上薪资要求的话，一定要加上左右两字或者给出一个范围，给自己留一些谈判空间。

2、信息要真实

诚信是最基本的沟通前提，无论对于用人企业还是以后与同事相处，如果简历中没有太多引人注意的地方，可以通过自荐信之类的弥补（后面附录中会提到）；不要想通过制造虚假的简历信息来获得职位，IT 这个行业求职时要提供毕业证，学位证，有经验的还要提供离职证明，很多公司会对你的简历信息进行验证（阿里就有求职者的档案库，对于不良求职者可能会进入黑名单，永远没有录用的机会），所以信息尽可能的真实，如果刚毕业的学生感觉自己的简历太空，就把实习工作和毕业设计都写进去充实一下。如果简历中的一些硬件信息达不到用人单位的要求，可以变通一下，把这些信息从简历中删除，毕竟 IT 行业更重实力，通过 OFFER 后再和 HR 解释清楚就行了。

３、谦虚适度、切忌张扬

刚毕业的求职者，简历上往往会看到很多精通 CoreJava，精通 SSH……总之简历上很多精通，精通这个词就是属于很张扬的词，可以说实际面试中十个难有一个求职者达到技术要求的精通级别。自信固然不可少，但切忌太张扬，容易让面试人员产生不好的感觉，感觉你有点华而不实，所以除了精通外还可以用一些熟练，这是一个很狡猾的词。虽然这高调的张扬不提昌，但过分谦虚也不行的，比如一个简历上写满了“会一点”、“了解”、“学过”之类的，企业单位虽然喜欢诚实，但付给你钱是让你干活的，这样的简历连ＨＲ一看就知道干不了事，可能都到不了面试官手里。

另外简历上的东西要和你的实际经验符合，别为了显示你的知识面广，经验丰富，乱七八糟的东西全写了，要知道你简历上的东西都是面试官可能会问到你的问题，写简历前要清楚写的每一项技术能不能禁得起面试官的提问，好多求职者都是简历投出后，上面的内容自己都不太清楚了，这点是很忌讳的。

４、项目经验不易过长

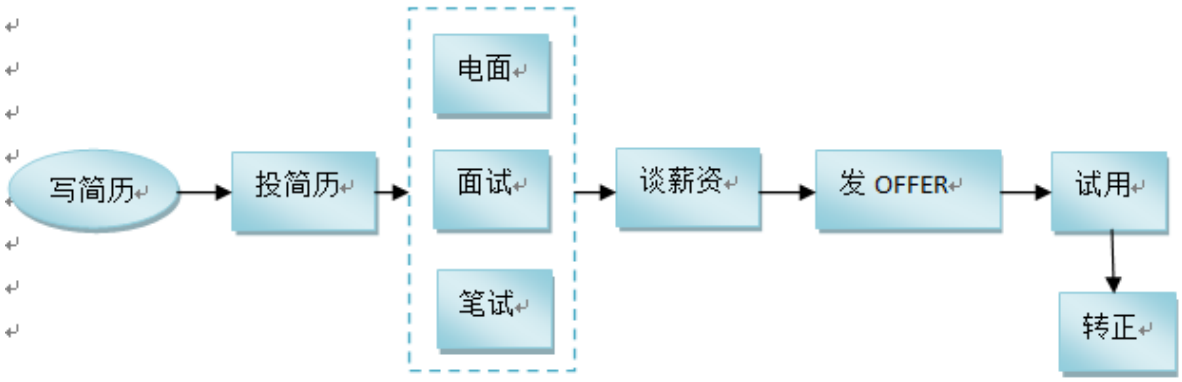
简历项目经验不易太多，一般四到六个为宜，写上你认为最有代表性的，然后描述详细一点就 OK 了；不要把那些参与三两行代码的项目都写上，面试官如果较真起来，可能认为你对项目了解不深入，留下负面印象；如果项目经验实在太多了，也可以针对职位来修改简历中的项目经验，这样命中率更高。其实优秀的员工一般都有自己钟情的公司，希望大家以后也都有这样的公司。

５、求职信

这个主要针对刚毕业的求职者，实际工作经验可能还没有，简历写出来有时候会很空白，在海量的简历中很容易被埋没；写一封求职信将有助于你获取更大的面试机会。求职信的目的是让对方了解自己、相信自己、录用自己，它是一种私人对公并有求于公的信函。但格式有一定的要求，内容要求简练、明确，切忌模糊、笼统、面面俱到。

第 02 章　应聘流程

１、应聘流程图



２、面试前的工作

I、写简历，除了要有一份电子档的简历外，还要在各大主要的招聘网上写一下简历，如 51job,智联，猎聘，100%OFFER，CSDN 招聘专栏等。对于刚毕业的学生，主要推荐前两个，职位多；对于有一定工作经验，重点推荐后三种，含金量高，待遇一般都不错。

II、 投简历约面试，写简历的目的就是投给用人单位以获得面试机会；切忌海投，盲目投，甚至乱投，可能会被 HR 打入黑名单；投简历一定要有针对性，一定要看清公司招聘的职位信息。约面试时，要问清时间，如果是当天要求面试的，最好协商一下是否可以晚一天，以便有时间了解一些公司信息和确认一些职位信息，这个很有必要；面试时出于礼貌最好早到十来分钟，如果因为意外可能迟到，最好给 HR 电话说明一下。

III、面试前一天根据招聘的职位信息，看一下此"Java 面试指南"，如果没有包含的网上查一下，做到有备无患，知己知彼。

第 03 章　福利如何谈

１、薪资多少合适

首先薪水福利是不能比的，同一年同一班出来的同学，工作几年后薪资差别是相当大的，不同的城市和地区也差别较大。对于有工作经验的朋友跳槽的话，可以参考自己的同学，和自己工作经验工作强度相似的朋友（不要盲目比较，毕竟技术能力，行业背景不同，而且工作强度、工作压力也不同），一般跳槽薪资比上家会浮动 30%左右；如果实在没法参考，可以参考网络上传的一个比较牛的计算公式：**3k+工作年限＊２**。对于刚毕业的求职者来说，经验比薪资更重要，但作为 IT 的同行建议坚决不能干那种白出力的活，一般实习本科大概 4k+，专科一般少 0.5K。

２、福利如何谈

面试成功后，和ＨＲ谈福利待遇时，千万不要只谈工资不谈福利，尤其是刚毕业的求职者没什么经验。谈薪资福利可以从以下几方面入手：第一，谈薪资的时候要问清是税前还是税后，是试用期工资还是转正后的工资，试用期是付全额还是付全额的 80%.第二，公司有没有年终奖，发多少，什么时候发？第三，公司福利怎么缴，是五险一金还是三险一金，缴费基数是多少？记住千万别忘记问一金（公积金，对以后买房可能很有帮助），然后再问清试用期给不给缴。第四，公司加班制度，有没有加班补助，好多人受不了加班太多，所以入职时最好问清楚。第五，像其它的旅游了，体检了，餐补了，高温补贴，过节礼品等其它的一些东西可以都问清楚。

第二部分　CoreJava 基础

第 04 章　解读 API

4.1　Object 常用的方法

解析：面试刚开始都很简单，但也容易给面试官留下印象，就像这个题，如果你有三年以上经验，回答的少于六个，那么估计你下面将很危险了。Object 中的这些方法都很精典，务必记住。

参考答案：object 常用的方法有 clone(),equals(),hashCode(),notify(),notifyAll(), toString(),wait(),finalize()。

4.2　String 与 StringBuffer 区别

解析：这是一个很基础的题，String 字符串的问题几乎很多面试官都喜欢问，因为在实际项目开发中用到的最多的变量就是 String，所以读者要重点对待，熟练掌握。

参考答案：String 与 StringBuffer 在 java 中都是用来进行字符串操作的。

String 是一个很别的类，它被 final 修饰，意味着 String 类不能被继承使用，一旦声明是一个不可变的类。

StringBuffer 是一个长度可变的，通过追加的方式扩充，并且是线程安全；尤其在迭代循环中对字符串的操作，性能一般优于 String。但线程安全意味着开销加大，性能下降，所以在线程安全的情况下一般用 StringBuilder 来代替 StringBuffer 使用，StringBuilder 的速度要优于 StringBuffer。

拓展：javaAPI 中哪些类是 final 的？String 的"+"号操作性能是不是一定慢？

4.3　Vector,ArrayList, LinkedList 的区别

解析：这里主要考察集合类的熟练程度，考察深度可深可浅；浅的主要是考察使用，深点可能考察底层的数据结构，算法实现等，感兴趣的朋友可以深究一下。

参考答案：

1、Vector、ArrayList 都是以类似数组的形式存储在内存中，LinkedList 则以链

表的形式进行存储。

2、Vector 线程同步，ArrayList、LinkedList 线程不同步；后两者一般用在线程安全的地方，也可以通过 Collections.synchronizedList(……);实现线程同步。

3、LinkedList 适合指定位置插入、删除操作，不适合查找；ArrayList、Vector 适合查找，不适合指定位置的插入、删除操作。

4、ArrayList 在元素填满容器时会自动扩充容器大小的 50%，而 Vector 则是 100%，因此 ArrayList 更节省空间。

5、LinkedList 还实现了 Queue 接口,该接口比 List 提供了更多的方法,包括 offer(),peek(),poll()等,多与一些线程池一起使用.

补充：以上都实现了序列化接口，这点在对象远程传输时很重要；而 Vector,ArrayList 实现了 AccessRandom 接口，这是一个标记接口，此接口的主要目的是允许一般的算法更改其行为，从而在将其应用到随机或连续访问列表时能提供良好的性能。

注意: 默认情况下 ArrayList 的初始容量非常小,所以如果可以预估数据量的话,分配一个较大的初始值属于最佳实践,这样可以减少调整大小的开销。

拓展：线性列表和双链表的数据结构实现？

4.4 HashTable, HashMap,ConcurrentHashMap 区别

解析：这里还有一个 TreeMap 没有列出，TreeMap 能够把它保存的记录根据 Key 键排序，所以 Key 值不能为空，默认是按升序排序，也可以写自己的比较器（TreeMap 排序是根据红黑树实现的，有兴趣的可以研究一下它的算法实现）。而 ConcurrentHashMap 是从 jdk1.5 新增的，是 HashMap 的一种线程安全的实现类，经常在多线程的环境中使用，由于底层的实现使用的是局部锁技术，在线程安全的前提下比 HashTable 的性能要好，推荐使用（ConcurrentHashMap 局部锁技术实际上就是把 Map 分成了 N 个 Segment，put 和 get 的时候，都是现根据 key.hashCode()算出放到哪个 Segment 中，而这里的每个 segment 都相当于一个小的 HashTable,有兴趣的朋友可以读一下源代码），此题经常在中高级职位面试中问到，所以请慎重对待。

参考答案：

1、HashTable 线程同步，key,value 值均不允许为空；HashMap 非线程同步，key,value 均可为空,HashTable 因为线程安全的额外开销会造成性能下降，而 HashMap 由于线程不安全，在多线程的情况下，一般要加锁，或者使用 Collections.synchronizedMap()来创建线程安全的对象。

2、HashTable 继承的父类是 Dictionary，而 HashMap 继承的父类是 AbstractMap，而且又都实现了 Map 接口，所以 Map 接口中提供的方法，他们都可以使用；HashTable 的遍历是通过 Enumeration，而 HashMap 通常使用 Iterator 实现。

3、HashTable 有一个 contains(Object value)的方法，其功能和 HashMap 的 containsValue(Object value)一样，而 HashMap 提供的更加细致还有一个取 Key 值的方法为 containsKey(Object key)。

4、哈希值的使用不同，HashTable 直接使用对象的 hashCode，代码是这样的：

```
int hash = key.hashCode();
int index = (hash & 0x7FFFFFFF) % tab.length;
```

而 HashMap 重新计算 hash 值，而且用与代替求模：

```
   int hash = hash(k);          int i = indexFor(hash, table.length);
static int hash(Object x) {       int h = x.hashCode();
       h += ~(h << 9);
       h ^= (h >>> 14);
       h += (h << 4);
       h ^= (h >>> 10);
       return h;}
static int indexFor(int h, int length) {return h & (length-1);}
```

//网络提供，面试时不追问，可以不说出具体的实现代码。

5、ConcurrentHashMap 是 HashMap 线程安全的实现，并且逐渐取代 Hashtable 的使用，因为 Hashtable 锁的机制是对整个对象加锁，而 ConcurrentHashMap 使用的是局部锁技术，实际上就是把 Map 分成了 N 个 Segment，put 和 get 的时候，都是现根据 key.hashCode()算出放到哪个 Segment 中，而这里的每个 segment 都相当于一个小的 Hashtable，性能将高于 HashTable。

4.5　Equals()和 HashCode()作用

解析：这个问题每个面试求职者或许都被问到过，属于常见题型，但也是比较难回答的面试题之一；包括很多面试官本身也有时候并不能解释得很清楚，如果要回答好这个小问题，首先要了解这两个方法的实现原理（在第 I 版中不会太深入讨论）。在《Effective java》中有这样一句话：***在每个覆盖了 equals 方法的类中，也必须覆盖 hashCode 方法***，如果不这样做的话，就会违反 Object.hashCode 的通用约定，从而导致该类无法结合所有基于散列的集合一起正常动作，这样的集合包括 HashMap,Hashtable,HashSet。这里列出一下 equals 重写的规则：

1 自反性：对任意引用值 X，x.equals(x)的返回值一定为 true.

2 对称性：对于任何引用值 x,y,当且仅当 y.equals(x)返回值为 true 时，x.equals(y)的返回值一定为 true;

3 传递性：如果 x.equals(y)=true, y.equals(z)=true,则 x.equals(z)=true

4 一致性：如果参与比较的对象没任何改变，则对象比较的结果也不应该有任何改变

5 非空性：任何非空的引用值 X，x.equals(null)的返回值一定为 false。

参考答案：这两个方法都是 Object 类的方法，equals 通过用来判断两个对象是否相等。HashCode 就是一个散列码，用来在散列存储结构中确定对象的存储地址。HashMap，HashSet，他们在将对象存储时，需要确定它们的地址，而 HashCode 就是做这个用的。在默认情况下，由 Object 类定义的 hashCode 方法会针对不同的对象返回不同的整数，这一般是通过将该对象的内部地址转换成一个整数来实现的。

在 java 中 equals 和 hashCode 之间有一种契约关系：

1. 如果两个对象相等的话，它们的 hashcode 必须相等。

2. 但如果两个对象的 hashcode 相等的话，这两个对象不一定相等。

由于 java.lang.Object 的规范，如果两个对象根据 equals()方法是相等的，那么这两个对象中的每一个对象调用 hashCode 方法都必须生成相同的整数结果。(下面作为解释内容可以不必回答，理解使用)

举例说，在 HashSet 中，通过被存入对象的 hashCode()来确定对象在 HashSet 中的存储地址，通过 equals()来确定存入的对象是否重复，hashCode()和 equals()都需要重新定义，因为 hashCode()默认是由对象在内存中的存储地址计算返回一个整数得到，而 equals()默认是比较对象的引用，如果不同时重写他们的话，那么同一个类产生的两个完全相同的对象就都可以存入 Set，因为他们是通过 equals()来确定的，这就是 HashSet 失去了意义。面试时简单的说，HashSet 的存储是先比较 HashCode，如果 HashCode 相同再比较 equals，这样做的目的是提交储存效率。所以换句方式问，如果用对象来做为 Key 值的话，要满足什么条件呢？答要重写 equals 和 HashCode 方法。

4.6　Sleep()和 Wait()区别

解析：多线程可以说是 java 面试中的重点难点之一，也是最能筛选面试者的分水领，这个只是属于最基本的考点，尽可能的回答周全，面试的朋友可以对多线程这块多花点时间。

参考答案：

1、wait()是 Object 类的方法，在每个类中都可以被调用；而 sleep()是线程类 Thread 中的一个静态方法，无论 New 成多少对象，它都属于调用的类的。

2、sleep 方法在同步对象中调用时，会持有对象锁，其它线程必须等待其执行结束，如果时间不到只能调用 interrupt()强行打断；在 sleep 时间结束后重新参与 cpu 时间抢夺，不一定会立刻被执行。

3、wait()方法在同步中调用时，会让出对象锁。通常与 notify,notifyAll 一起使用。

4.7　IO 与 NIO 的区别

解析：这是现在面试中常被问到的一个问题，尤其做后台开发的职位；NIO 是从 jdk1.4 就引入了，但是如果没有实际开发中使用过，不是很容易说的很清楚，这里从概念的层面介绍一下，希望有兴趣的朋友能更深层次的研究一下。

参考答案：IO 是早期的输入输出流，而 NIO 全名是 New IO 在 IO 的基础上新增了许多新的特性。提供的新特性包括：非阻塞 I/O，字符转换，缓冲以及通道。

1、IO 是面向流的，NIO 是面向缓冲区的。 Java IO 面向流意味着每次从流中读一个或多个字节，直至读取所有字节，它们没有被缓存在任何地方。如果需要前后移动从流中读取的数据，需要先将它缓存到一个缓冲区。 Java NIO 的缓冲导向方法略有不同。数据读取到一个它稍后处理的缓冲区，需要时可在缓冲区中前后移动。

2、Java IO 的各种流是阻塞的。这意味着，当一个线程调用 read() 或 write()时，该线程被阻塞，直到有一些数据被读取，或数据完全写入。该线程在此期间

不能再干任何事情了。 Java NIO 的非阻塞模式，使一个线程从某通道发送请求读取数据，但是它仅能得到目前可用的数据，如果目前没有数据可用时，就什么都不会获取。线程通常将非阻塞 IO 的空闲时间用于在其它通道上执行 IO 操作，所以一个单独的线程现在可以管理多个输入和输出通道（channel）。虽然是非阻塞的，但也会遇到一个问题就是服务器对最大连接的支持，但在线用户连接数大于系统支持数时，NIO 的默认实现是并不管是否还有足够的可用连接数，而是直接打开连接。在 netty 框架中经常会看到一个 open too many files 异常就是由此引起的，所以要灵活使用，合适配置。
3、Java NIO 的选择器允许一个单独的线程来监视多个输入通道，你可以注册多个通道使用一个选择器，然后使用一个单独的线程来“选择”通道：这些通道里已经有可以处理的输入，或者选择已准备写入的通道。这种选择机制，使得一个单独的线程很容易来管理多个通道。

扩展：Netty 是 NIO 实现的一个经典框架，好多职位中都要求熟悉 netty，大家有时间可以研究一下。

4.8 Synchronized 和 Lock 区别和用法

解析：这里相对于上面的 wait 与 sleep 的区别来说，难度明显增加，也是常见的面试题。

参考答案：
首先他们都是用来实现线程的同步操作的，synchronized 是 jdk1.2 以来一直存在，而且 Lock 是 jdk1.5 的新增；具体区别如下：

1、Lock 是 JDK1.5 才出现的，而在 jdk1.5 及之前的 synchronized 存在很大的性能问题，尤其在资源竞争激烈的条件下，性能下降十多倍，而此时的 Lock 还基本能维持常态。（synchronized 在 jdk1.5 后的版本，作了很大的性能优化）

2、使用 synchronized 时对象一定会被阻塞，其它线程必须等待锁释放后才能执行，在高并发的情况下，很容易降低性能，而且控制不当还容易造成死锁，所以要合适的利用 Synchronized，并且尽可能的精确到最小的业务逻辑块。而 Lock 提供了更细致的操作，其常用的实现类有读写锁，这里以 ReentrantLock 为例，它拥有 Synchronized 相同的并发性和内存语义，此外还多了锁投票，定时锁等候和中断锁等候等很多详细的方法操作。 ReentrantLock 获取锁定与三种方式：a) lock(), 如果获取了锁立即返回，如果别的线程持有锁，当前线程则一直处于休眠状态，直到获取锁 b) tryLock(), 如果获取了锁立即返回 true，如果别的线程正持有锁，立即返回 false；c)tryLock(long timeout,TimeUnit unit)，如果获取了锁定立即返回 true，如果别的线程正持有锁，会等待参数给定的时间，在等待的过程中，如果获取了锁定，就返回 true，如果等待超时，返回 false；d) lockInterruptibly: 如果获取了锁定立即返回，如果没有获取锁定，当前线程处于休眠状态，直到或者锁定，或者当前线程被别的线程中断。

3、synchronized 是在 JVM 层面上实现的，不但可以通过一些监控工具监控 synchronized 的锁定，而且在代码执行时出现异常，JVM 会自动释放锁定，但是使用 Lock 则不行，lock 是通过代码实现的，要保证锁定一定会被释放，就必须将 unLock()放到 finally{}中。

补充：

A．无论 synchronized 关键字加在方法上还是对象上，他取得的锁都是对象，而不是把一段代码或函数当作锁——而且同步方法很可能还会被其他线程的对象访问。 B．每个对象只有一个锁（lock）和之相关联。
C．实现同步是要很大的系统开销作为代价的，甚至可能造成死锁，所以尽量避免无谓的同步控制。synchronized 可以加在方法上，也可以加在对象上，通常理解为，只有持有了锁才可以进行对应代码块的执行。
java.util.concurrent.locks 包下面提供了一些锁的实现，有读写锁，公平锁等。
将 synchronized 替换成 lock 的实现可以提升性能：
1. 大部分应用场景是读写互斥，写和写互斥，读和读不互斥。而 synchronized 则是都互斥。
可以利用读写锁来优化性能，读锁锁住读的代码块，写锁锁住写的代码块。
2. 要确保你在理解原来利用到 synchronized 的代码逻辑，避免一概而论地把 synchronized 替换成锁。
PS：被 synchronized 修饰的方法所在的对象，其它线程可以访问该对象的非 synchronized 修饰的方法，不能访问 synchronized 的任何方法。

说明：里面有一部分看网上总结的比较好，就直接拿过来使用了，当然面试时也不是说必须要把上面的点回答完，最重要要通过自己的理解表达出来。

4.9 Final、Finally、Finalize 的区别

解析：这是一个很老很老就存在的面试题，但还是经常被问到，越简单的题在面式中越不允许答错。

参考答案：
Final 是 Java 中一个特别的关键字（修饰符），final 修饰的类不能被继承，final 修饰的变量不能被修改（是常量）,final 修饰的方法不能被重写，所以使用 final 要考虑清楚使用场景。
Finally 是 Java 异常处理 try{}catch(..){}finally{}中的一部分，finally 语句块中的语句一定会被执行到（Down 机除外），通常用于进行资源的关闭操作，如数据连接关闭等。
Finalize 是 Object 类中的一个方法，finalize() 方法是在垃圾收集器删除对象之前对这个对象调用的，设计的初终是可以进行一些资源的关闭，但这个方法很少被重写，因为垃圾回收器执行时并不一定会调用，比如抛异常时，而且垃圾回收本身就是随意的，不可控制的。

4.10 OverLoad 与 Override 的区别

解析：这里其实是考的是 Java 面向对象的三（四，如果算上抽象就是四大特征）大特征，封装、继承、多态、（抽象）中的多态，属于基础类型常考题之一。

参考答案：overload 和 override 是多态的两种表现形式；
overload 重载规则是***同名不同参***,方法名相同，参数类型、个数、顺序至少有一个不相同（与返回值、参数名、抛出的异常都没有关系）；

overload 可以发生在父类、同类、子类中。override 重写的规则是***方法名、参数、返回值相同***，发生在父子类之间，而且子类不能抛出比父类更多的异常，被重写的方法一定不能定义成 final。
拓展：static 方法能不能被 overload,override?

4.11　Collection 与 Collections 的区别

参考答案：Collection 是一个集合的超级接口，它定义了一些集合常用的操作方法，如 add,size,remove,clear 等，我们经常用到的 List（可重复集合）,Set（不可重复集合）接口就是它的子接口。

Collections 此类完全由在 collection 上进行操作或返回 collection 的静态方法组成。它包含在 collection 上操作的多态算法，即“包装器”，包装器返回由指定 collection 支持的新 collection，以及少数其他内容。如果为此类的方法所提供的 collection 或类对象为 null，则这些方法都将抛出 NullPointerException。常用到的方法如 copy,sort,reverse,replaceall 等。

第 05 章　Java 基础知识

5.1“==”与“equals”区别

解析：这是一个很简单的题，很多人都能答上来“==”通常用来比较地址，而且 equals 则是比较内容；如果仅仅这样回答，通常面试只能得到 60 分，可以通过下面的两个例子来看一下。

```
String s1 = "s";
String s2 ="s" ;
String s3 = new String("s");
String s4 = new String("s");
StringBuffer sb1 = new StringBuffer("s");
StringBuffer sb2 = new StringBuffer("s");
System.out.println(s1==s2) ;
System.out.println(s3==s4) ;
System.out.println(sb1==sb2) ;
System.out.println(s1.equals(s2)) ;
System.out.println(s3.equals(s4))
System.out.println(sb1.equals(sb2)) ;
```

读者可以自己判断一下结果;对于 s1,s2 会指向同一个地址，因为这里的 s1 会保存在常量池中，s2 会先读取常量池，发现里面已经存在”s”了，不会再进行重新开辟空间。对于 s3,s4 是在堆中分配两块空间，它们的地址是存放在栈中。sb1 与 sb2 的内在分配和 s3,s4 一样，但注意 StringBuffer 没有重写 equal 方法，而调

用的是 Object 中的 equal 方法，看一下源码可以知道 Object 中的 equal 的实现 return (this == obj)其实就是在比较对象的地址。

参考答案：“==”对于基本数据类型，通常用来比较值（基本数据类型是在栈中进行分配的）相等，对于对象通常用来比较对象的地址。equals 是 Object 中的方法，可以根据业务需要进行重写，比如 String 中重写了 equals 的方法，常用来比较两个字符串内容是否相等。
说明：重写 equal 的风险也很大的，上面也提到过重写的规则，所以根据需要进行重写。

5.2 接口和抽象类的区别

解析：这是一个很简单的题，面试时一定要想清楚再回答，这样的题答错，估计机会也就失去了。

参考答案：
含有 abstract 修饰符的 class 即为抽象类，abstract 类不能创建实例对象，含有 abstract 的方法的类必须定义为 abstract class ，abstract class 里的方法不必是抽象的，抽象类中定义抽象方法必须放在具体子类中实现，所以，不能有抽象的构造方法或抽象的静态方法，如果子类没有实现抽象父类中的所有方法，那么，子类也必须定义为抽象类。
接口（interface）可以说成是抽象类的特例。接口中的所有方法都必须是抽象的，接口中的方法定义默认为 public abstract 。接口中的变量是全局常量，即 public static final 修饰的。概括如下：
1、抽象类里可以有构造方法，而接口内不能有构造方法。
2、抽象类中可以有普通成员变量，而接口中不能有普通成员变量。
3、抽象类中可以包含非抽象的普通方法，而接口中所有的方法必须是抽象的，不能有非抽象的普通方法。
4、抽象类中的抽象方法的访问类型可以是 public ，protected 和默认类型，但接口中的抽象方法只能是 public 类型的，并且默认即为 public abstract 类型。
5、抽象类中可以包含静态方法，接口内不能包含静态方法。
6、抽象类和接口中都可以包含静态成员变量，抽象类中的静态成员变量的访问类型可以任意，但接口中定义的变量只能是 public static 类型，并且默认为 public static 类型。
7、一个类可以实现多个接口，但只能继承一个抽象类。
以上答案网上都是可以找的到的，也很有条理，如果能回答出五条左右就可以满分了。除此外，下面就是加分项了，在设计上接口的作用侧重于模块划分，比如：用户管理接口，信息管理接口，里面都相关用户的增删改查操作。而抽象类则是侧重于功能的划分，比如事先知道需要实现某个功能，但设计时却还没有想到如何实现则先抽象出一个方法，或者把子类中重复实现的功能放到抽象中去实现等。

5.3　运行时异常和一般异常的区别

解析：异常的优点是可以提高程序的健壮性，可读性，可维护性，但异常的设计初衷是用于不正常的情形，一般的 JVM 是不会试图对它进行优化，所以异常使用不当也会给系统带来负面影响。在谈异常之前，往往还会谈一下 Error，它们都是基于 Throwable 实现；但 Error 的出现通常是 JVM 层次无法自动恢复的严重例外，比如 JVM 内存泄露，一般都会造成系统荡机，所以对于 Error 进行继承设计的没有意义的。而 Exception 是可以处理的，合理的使用可以使用系统更健壮。

参考答案：运行异常（RuntimeException）和一般异常（CheckedException）都是从 Exception 继续来的，都是可以捕捉到，可以恢复的例外。

一般异常（CheckedException）：是 JVM 强制让我们处理的异常，在现在通常的开发 IDE 会提示我们必须用 try{}catch(..){}捕捉，常见的文件读写的 IO 导常，Socket 通信时的网络异常，数据操作的 JDBC 异常等。

运行异常（RuntimeException）：JVM 不强制我们去捕捉，异常一旦发生，会有 JVM 接管，一直往上抛，直到遇到异常处理的代码；如果没有处理块，到最上层，如果是多线程就由 Thread.run()抛出，如果是单线程就被 main()抛出。抛出之后，如果是线程，这个线程也就退出了。如果是主程序抛出的异常，那么这整个程序也就退出了。

异常处理的目的就是使程序从异常中恢复，不会中止线程退出系统，所以对于常的 Exception 要做一些合理的处理，但由于异常机制比较复杂而且往往得不到 JVM 的优化，会造成性能下降下，所以异常的使用最好对可能出现的异常位置进行捕捉，最好同时进行异常日志记录，便于系统维护。

5.4　序列化和反序列化

解析：对于经常接口开发的的来说，序列化这个词一定不会默生，对象本身是不能直接在网络中传输的，传递的都是一些字节流，而将对象转换成字节流的过程就叫做对象的序列化。这个考点的命中率属于一般，一般多出现于后台接口开发相关的职位面试中，但序列化与反序列化的使用却在开发中无处不在，需要深入了解。

参考答案：

1、序列化与反序列化的定义：把对象转换为字节序列的过程称为对象的序列化，把字节序列恢复为对象的过程称为对象的反序列化。

2、对象序列化的作用，比如远程接口调用，对象网络传输；深度克隆时，复杂对象的的复制；缓存对象的硬盘存储；Session 的硬盘序列化等。至于反序列化的作用，就是把序列化的对象重新加载到内存中使用。

3、序列化的实现，首先必须实现 Serializable 或者 Externalizable，而 Serializable 是超级接口，Externalizable 是其实现，Serializable 接口中什么也没有定义，它只是对对象生成一个唯一的 ID 标识，告诉 JVM 该对象是可以序列化的，如果不实现序列化接口，在序列化的过程中会抛出异常。

4、JavaAPI 提供的序列化方式，java.io.ObjectOutputStream 代表对象输出流，

它的 writeObject(Object obj)方法可对参数指定的 obj 对象进行序列化，把得到的字节序列写到一个目标输出流中。java.io.ObjectInputStream 代表对象输入流，它的 readObject()方法从一个源输入流中读取字节序列，再把它们反序列化为一个对象，并将其返回。对于实现了 Externalnalizable 接口的类，且类必须实现 readExternal(ObjectInput in)和 writeExternal(ObjectOutput out)方法，按照以下方式进行序列化与反序列化。ObjectOutputStream 调用该类生成对象的 writeExternal(ObjectOutput out))的方法进行序列化。ObjectInputStream 会调用对象的 readExternal(ObjectInput in)的方法进行反序列化。
扩展：Java 实现序列化带来了很多好处，但实际开发中要根据项目类型和业务场景慎重实现，因为实现序列化也会付出一定的“代价”，首先会降低类的灵活性，其实会增加出现 BUG 和漏洞的可能。

5.5 Java 实现浅克隆与深克隆

解析：这个面试命中的概率不算太高，此题目是笔者在某家公司的面试的原题，而笔者之前的工作中研究过克隆的使用，在这里总结一下供大家参考（里面用到序列化的一些知识，参考上面）。

参考答案：
所谓的浅克隆，顾名思义就是很表面的很表层的克隆，如果我们要克隆 Administrator 对象，只需要克隆他自身以及他包含的所有对象的引用地址。而深克隆，就是非浅克隆。克隆除自身以外所有的对象，包括自身所包含的所有对象实例。至于深克隆的层次，由具体的需求决定，也有“N 层克隆”一说。但是，所有的基本（primitive）类型数据，无论是浅克隆还是深克隆，都会进行原值克隆。毕竟他们都不是对象，不是存储在堆中。（Java 中所有的对象都是保存在堆中，而堆是供全局共享的。也就是说，如果同一个 Java 程序的不同方法，只要能拿到某个对象的引用，引用者就可以随意的修改对象的内部数据。）
潜克隆实现：1. 让该类实现 java.lang.Cloneable 接口；2. 重写（override）Object 类的 clone()方法。（并且在方法内部调用父对象的 clone()方法）。
对于复杂对象，比如数组、集合通过浅克隆是不能修改对象的引用的，这里要彩用深度克隆，一般是通过序列和反序列的方式来实现浓度克隆：

```
try
{
    if (src != null)
    {
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        ObjectOutputStream oos = new ObjectOutputStream(baos);
        oos.writeObject(src);
        oos.close();

        ByteArrayInputStream bais = new ByteArrayInputStream(baos
                .toByteArray());
        ObjectInputStream ois = new ObjectInputStream(bais);

        o = ois.readObject();
        ois.close();
    }

} catch (IOException e)
{
    e.printStackTrace();
} catch (ClassNotFoundException e)
{
    e.printStackTrace();
}
return o;
```

如上面的代码，对序列化的对象写到内存或者研盘上，然后在需要的地方再从内存或者硬盘上反序列化成对象。

5.6　枚举可以序列化吗

解析：这个是笔者参加某一个著名的互联网公司的面试题，冷不丁上来这个小问题，如果你平时没太关注，很容易认为是陷阱，大江大河都淌过的人很容易死在小水塘里；而且在两家的互联网开发设计规范中都提到多使用 Enum，所以像这样的小知识点平时还是要多留意。

参考答案：因为 Enum 实现了序列化接口 Serializable，所以枚举是可以被序列化的。

5.7　Java 创建对象的方式

解析：这个没什么深意，面试官多是想了解一下你的知识面，一般人都应该至少能回答出来两种。

参考答案：

(1) 用 new 语句创建对象，这是最常见的创建对象的方法。

(2) 运用反射手段,调用 java.lang.Class 或者 java.lang.reflect.Constructor 类的 newInstance()实例方法。

(3) 调用对象的 clone()方法。

(4) 运用反序列化手段，调用 java.io.ObjectInputStream 对象的 readObject()方法。

5.8　能否自己定义一个 java.lang.String 类

解析：这个是考察面试人员对类加载机制的了解，问题不难，但需要真正了解 JVM 对类的加载原理，这个在下面 JVM 相关部分还会详细提及，这里就简答一下。

参考答案：
一般情况下是不可以的，类的 APP 加载器会根据双亲代理机制委托父类加载器去加载此类，而在父亲加载器中已经加载过此类(因为 java.lang.String 类是 JVM 定义的类)，会报该类已经存。处理方法，自定义加载器加载，指定一个其它路径。

5.9　Java 线程池的了解

解析：这里还是多线程考察的延续，在互联网职位、高并发分布式相关的职位中几乎必然会面试多线程题目，而且有资格面试这块的面试官一般都是设计、架构级别的资深人士，难度中偏上，是面试有效筛选题目类型；线程池在 jdk1.5 之前的版本并不是特别成熟，但在 Jdk1.5 的版本中作了很大的优化，在 jdk1.6 的版本中更是引入了 java.util.concurrent 包，有兴趣的读者可以详细研究一下这个包；而回答此问题可以从线程池的作用、为什么要使用线程池及线程池的优越性、线程池的实现原理和使用等几方面回答。

参考答案：

1、线程池的作用：在没用使用线程池的情况下，开发人员通常自己来开发管理线程，很容易将线程开启过多或者过少；过多将造成系统拥堵，过少又浪费系统资源。用线程池控制线程数量，其他线程排队等候。一个任务执行完毕，再从队列的中取最前面的任务开始执行。若队列中没有等待进程，线程池的这一资源处于等待。当一个新任务需要运行时，如果线程池中有等待的工作线程，就可以开始运行了；否则进入等待队列。

2、由于手动维护线程成本很高（大家都知道多线程的开发难度大，对程序员要求高，而且代码不易调试维护），所以我们应尽可能用 JDK 提供的比较成熟的线程池技术,它减少了创建和销毁线程的次数，每个工作线程都可以被重复利用，可执行多个任务。可以根据系统的承受能力，调整线程池中工作线线程的数目，以达到系统性能最优。

3、线程池的父接口是 Executor，它只定义了一个 Execute 方法，而通过使用的是它的扩展接口 ExecutorService，它有几个常用的实现类 AbstractExecutorService, ScheduledThreadPoolExecutor, ThreadPoolExecutor（详见 API）。而在实际开发中，通过使用的是一个 java.util.concurrent 中的一个工具类 Executors，里面提供了一些静态工厂，生成常用的线程池(下面这些作为了解，说出一两个就可以了)。

i. newSingleThreadExecutor

创建一个单线程的线程池。这个线程池只有一个线程在工作，也就是相当于单线程串行执行所有任务。如果这个唯一的线程因为异常结束，那么会有一个新的线程来替代它。此线程池保证所有任务的执行顺序按照任务的提交顺序执行。

ii.newFixedThreadPool

创建固定大小的线程池。每次提交一个任务就创建一个线程，直到线程达到线程池的最大大小。线程池的大小一旦达到最大值就会保持不变，如果某个线程因为执行异常而结束，那么线程池会补充一个新线程。

iii. newCachedThreadPool

创建一个可缓存的线程池。如果线程池的大小超过了处理任务所需要的线程，那么就会回收部分空闲（60 秒不执行任务）的线程，当任务数增加时，此线程池又可以智能的添加新线程来处理任务。此线程池不会对线程池大小做限制，线程池大小完全依赖于操作系统（或者说 JVM）能够创建的最大线程大小。

iv.newScheduledThreadPool

创建一个大小无限的线程池。此线程池支持定时以及周期性执行任务的需求。

5.10 Return 和 finally 语句块的顺序

解析：这个面试题就是考查基本功的，而且出现频率较高，面试官大部分也不会深问，但希望读者关注一下拓展中的问题，在面试笔试中都有很高的命中率。

参考答案：Finally 是在 return 语句之前被执行的,但是 finally 与 return 是相互独立的；return 语句位置会影响到返回值的结果，finally 中的计算将影响到其后 return 结果,不会影响到 try 里面 return 的结果。

扩展：Finally 做为 try{}catch(..){}finally{}一部分，如果 finally 删掉可以吗？catch()删掉呢？只留 try{}可以吗？try{}finally{}连用呢？请诸位试一下，这里也是一个考点，这里就不详细说明了。

5.11 Java 静态变量与实例变量的区别

解析：这个属于 java 面向对象中语法级别的考查，是一种基本的面试题，也是不容许失分的题目。

参考答案：Java 成员变量中静态变量用 static 修饰，而没有 static 修饰的变量称为实例变量。

Java 静态变量也叫类变量，在类加载时被分配空间（从 java 内存分配上可以知道，它是存放在静态域中），由于它是属于类的，为所有实例共享，它的特性是一改全改（任何一个实例中修改了这个属性，其它实例都能看到修改的结果），调用方式通过类名.变量名。

Java 实例变量属于对象的，在对象被实例化的时候（new 的时候）分配空间，每个实例化的对象都有一套自己的实例变量，调用方法通常通过生成的 set/get 方法，或者通过 this.实例变量名。

5.12 同步与异步区别及使用场景

解析：同步一定是发生在多线程条件下，对共同资源不安全的访问进行控制一种手段；异步是要求减少请求方时间的等待。

参考答案：如果数据将在线程间共享.例如正在写的数据以后可能被另一个线程读到,或者正在读的数据可能已经被另一个线程写过了,那么这些数据就是共享数据,必须进行同步存取。
当应用程序在对象上调用了一个需要花费很长时间来执行的方法,并且不希望让程序等待方法的返回时,就应该使用异步编程,在很多情况下采用异步途径往往更有效。
使用场景：同步，最简单最常见的就是火车票销售，不可能多个窗口下允许出现卖两张完全一样的票。异步，在注册页面，当我们写其它信息时，我希望后台已经完成了对我姓名是否重复的验证，而不是我完全写完提交后再去验证。

5.13 Java 堆和栈的区别

解析：要想详细了解堆和栈的区别，就要了解 JavaVM 的内存分配机制，一般会涉及以下几个区域寄存器、堆、栈、静态域、常量池、非 RAM 存储（这些概念性的东西，理解就好），静态域主要是存放一些 static 静态成员、常量池存放常量，下面主要说一下堆、栈。

参考答案：栈与堆都是 Java 用来在 Ram 中存放数据的地方，与 C 和 C++不同 Java 自动管理栈和堆，程序员不能直接地设置栈或堆。
栈中存放局部变量（基本类型的变量）和对象的引用地址。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；**栈是跟随线程的，有线程就有栈**。
堆中存放对象，包括对象变量以及对象方法（大学时老师可能都讲过一个类似的笑话，一 New 一堆，说的就是堆是用来存放对象的）。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，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。**堆是跟随 JVM 的，有 JVM 就有堆内存**。
http://www.csdn123.com/html/exception/723/723513_723516_723514.htm

5.14 Java 多线程快速问答

解析：多线程中有很多知识点，面试经常问到，而且会被连续问到，这里简单介绍一下。

1、Java 实现多线程的方式有哪些？
答：Java 实现多线程的方式通常有实现 Runnable 接口，或者继承 Thread 类，或者从线程 ExecutorService 中获取。

2、你通常用哪种方式来实现线程，为什么？

答：通常使用实现 Runnable 接口或者从线程池中来获得经程；实现接口是因为接口是支持多实现，我们知道“一个类只能有一个父类，但是却能实现多个接口”，因此 Runnable 具有更好的扩展性，而且 Thread 也是实现了 Runnable 接口，所以推荐通过接口实现来实现线程。另一种方式就是从线程池中获取线程，优点是将线程交给线程池去管理，可以减轻大量的精力去维持线程状态的管理，而且手动干预的越多，后期的维护成本越高。

3、run()和 start()哪个是线程的启动方法？

答：run()方法，在实现一个线程类时必须重写的方法，但它本身只是一个普通的方法，如果通过对象.run()去调用，它只是一个属于调用线程中的普通方法，可以调用多次，而且是按顺序执行。而 start()方法是线程的启动方法，当他把调用时，线程的状态就变成了就绪，一旦获取了时间片，就会立刻执行（执行 run()方法，这里的 run 是线程体，执行完成表示线程执行结束）。

4、多线程状态有哪些？

答：Java 线程有五种状态**创建、就绪、运行、阻塞和死亡**。创建：可以理解我们 new 了一线程对象；就绪：new 的线程对象调用了 start()方法，但并没有立即抢到 CPU 时间片；运行：线程启动后，线程体 run 方法在执行；阻塞：阻塞状态是指线程因为某些原因放弃 CPU，暂时停止运行。当线程处于阻塞状态时，Java 虚拟机不会给线程分配 CPU，直到线程重新进入就绪状态，它才会有机会获得运行状态；死亡：当线程执行完 run()方法中的代码或者调用了 stop()方法，又或者遇到了未捕获的异常，就会退出 run()方法，此时就进入死亡状态，该线程结束生命周期。

5、多线程的同步方法有哪些？

答：**wait():**使一个线程处于等待状态，并且释放所持有的对象的 lock。**sleep():**使一个正在运行的线程处于睡眠状态，是一个静态方法，调用此方法要捕捉 InterruptedException 异常。**notify():**唤醒一个处于等待状态的线程，注意的是在调用此方法的时候，并不能确切的唤醒某一个等待状态的线程，而是由 JVM 确定唤醒哪个线程，而且不是按优先级。**notifyAll():**唤醒所有处入等待状态的线程。实现线程同步一个是 synchronized 关键字，一个是通过对象 Lock。

6、你了解死锁吗（如果是笔试可能让你写一段死锁代码）？

答：死锁可以这样认为：多个线程同时被阻塞，它们中的一个或者全部都在等待某个资源被释放（例如有 a,b 两个线程，a 等待 b 让出了某个资源才可以执行；而 b 同样等待 a 让出了某个资源才可以继续，ab 都在等待对方就是线程同时阻塞，结局就是死锁）。

7、当一个线程进入到一个对象的 synchronized 方法，那么其他线程是否可以进入该对象的其它方法？

答：当一个线程进入到一个对象的 synchronized 方法，那么其他线程是可以进入这个对象的非 synchronized 方法，但不可能进入 synchronized 方法

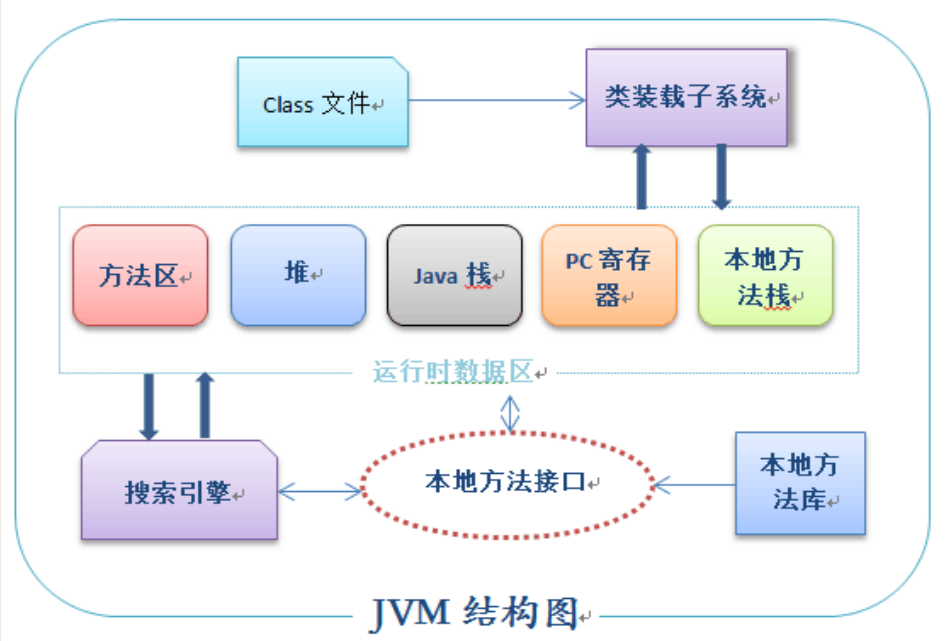
扩展：多线程面试是 Java 面试中不可缺少的一部分，也是属于面试中的难点，绝对是面试中的淘汰型问题，需要多花点时间准备。

第 06 章　JVM 相关

6.1　JVM 的工作原理

解析：很多大的互联网公司笔试时，会让你画一下 JVM 的结构图，然后让你解释一下各部分的意义，所以这个面试命中率还是很高的，作为基础扎实的求职者是应该要会的。纠错：搜索引擎--->执行引擎。

参考答案：下面是我画的一个 JVM 的结构图



1. 类装载器子系统：
负责查找并装载 Class 文件到内存，最终形成可以被虚拟机直接使用的 Java 类型。

2. 方法区：
在类装载器加载 class 文件到内存的过程中，虚拟机会提取其中的类型信息，并将这些信息存储到方法区。方法区用于存储已被虚拟机加载的类信息、常量、静态变量、即时编译器编译后的代码等数据。由于所有线程都共享方法区，因此它们对方法区数据的访问必须被设计为是线程安全的。

3. 堆：
存储 Java 程序创建的类实例。所有线程共享，因此设计程序时也要考虑到多线程访问对象(堆数据)的同步问题。

4. Java 栈：
Java 栈是线程私有的。每当启动一个新线程时，Java 虚拟机都会为它分配一个 Java 栈。Java 栈以帧为单位保存线程的运行状态。虚拟机只会直接对 Java 栈执行两种操作：以帧为单位的压栈或出栈。当线程调用 java 方法时，虚拟机压入一个新的栈帧到该线程的 java 栈中。当方法返回时，这个栈帧被从 java 栈中弹出并抛弃。一个栈帧包含一个 java 方法的调用状态，它存储有局部变量表、操作栈、动态链接、方法出口等信息。

5. 程序计数器：

一个运行中的 Java 程序，每当启动一个新线程时，都会为这个新线程创建一个自己的 PC(程序计数器)寄存器。程序计数器的作用可以看做是当前线程所执行的字节码的行号指示器。字节码解释器工作时就是通过改变这个计数器的值来选取下一条需要执行的字节码指令，分支、循环、跳转、异常处理、线程恢复等基础功能都需要依赖这个计数器来完成。如果线程正在执行的是一个 Java 方法，这个计数器记录的是正在执行的虚拟机字节码指令的地址；如果正在执行的是 Natvie 方法，这个计数器值则为空(Undefined)。

6.本地方法栈：

本地方法栈与虚拟机栈所发挥的作用是非常相似的，其区别不过是虚拟机栈为虚拟机执行 Java 方法（也就是字节码）服务，而本地方法栈则是为虚拟机使用到的 Native 方法服务。任何本地方法接口都会使用某种本地方法栈。当线程调用 Java 方法时，虚拟机会创建一个新的栈帧并压入 Java 栈。然而当它调用的是本地方法时，虚拟机会保持 Java 栈不变，不再在线程的 Java 栈中压入新的帧，虚拟机只是简单地动态链接并直接调用指定的本地方法。如果某个虚拟机实现的本地方法接口是使用 C 连接模型的话，那么它的本地方法栈就是 C 栈。

7.执行引擎：

负责执行字节码。方法的字节码是由 Java 虚拟机的指令序列构成的。每一条指令包含一个单字节的操作码，后面跟随 0 个或多个操作数。执行引擎执行字节码时，首先取得一个操作码，如果操作码有操作数，取得它的操作数。它执行操作码和跟随的操作数规定的动作，然后再取得下一个操作码。这个执行字节码的过程在线程完成前将一直持续。

6.2　类的加载机制

解析：这个题目看上去不太容易，但实际上在学习 Java 第一节课的时候，这些知识老师都已经告诉我们了，第一节课就是安装 JDK 配置环境变量，PATH,JAVA_HOME 大家肯定不默生，但随着 IDE 的使用，CLASSPATH 环境变量被慢慢遗忘了，CLASSPATH 就是配置我们自己开发的项目路径，JVM 在加载我们写的类时，会根据 classpath 从左到右各个项目中依次查找是否有这个包，包下是否有这个类，找到类后会验证其中的 packeage 包的路径是否与参数中的路径一致，是则加载，不是会报 NotFoundClassException 的异常。（这个刚毕业的同学可能比有工作经验的朋友印象更深刻）

参考答案：要了解类的加载机制，首先要了解 JVM 使用到的类的加载器；这里主要有 bootstrap classloader、extension cassLoader、AppClassLoader 三个类的加载器，第一个是 C++写的，两后两个都是 JAVA 写的，JVM 还给我们开放一个 ClassLoader 来扩展自己的加载器,当然后两个加载器也是继承了 ClassLoader。bootstrap classloader 在 JVM 运行的时候加载 java 核心 API 以满足 java 程序最基本的需要，其中就包括用户定义的 ClassLoader， ExtClassLoader 是用来加载 java 的扩展 API 的，也就是/lib/ext 中的类。AppClassLoader 是用来加载用户机器上 CLASSPATH 设置目录中的 Class 的，通常在没有指定 ClassLoader 的情况下，程序员自定义的类就由该 AppClassLoader 来进行加载。

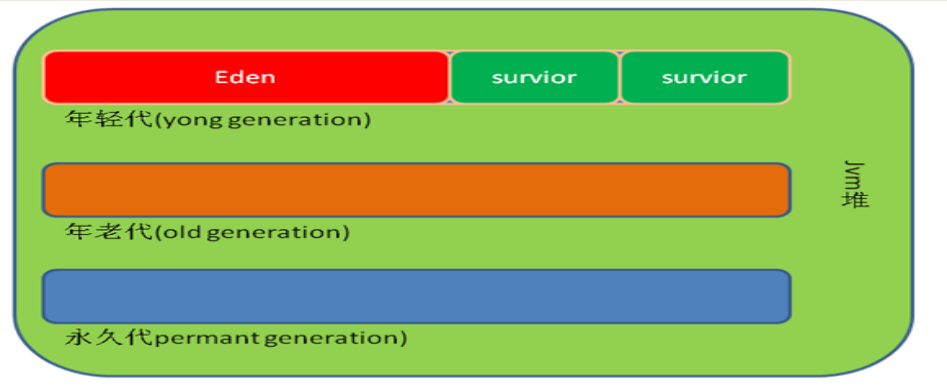
JVM 为了避免类的重复加载，引入了委托加载机制，比如 A 中定义 B 的引用，对于 B 的加载是通过委托 A 的加载器去加载;在上面我们也提到了双亲委托加载

机制，APP 加载器加载一个类时，首先会调 ExtClassLoader, ExtClassLoader 会调用 bootstrap classloader 如果发现类已经加载就不再加载。
　扩展：回答完上面这些应该已经差不多了，求职者可以多关注一下，比如预加载，被动加载等。

6.3　GC 的工作原理

解析：Java 的垃圾回收机制是 java 语言优越性的一大特色，也奠基它的高级语言的地位。这是一个初级、中级、高级、专家级职位都喜欢面试的问题，因为只有熟练了解 JavaGC 工作原理，才能写出质量更高的代码，才能更好的在代码级别做性能优化

参考答案：现在市场上 JVM 的实现厂家很多，不同的厂家对垃圾回收的算法实现也不同，目前主流上常的算法有**标记清除、分代、复制、引用记数、增量回收**等，而目前最常用的 JVM 是 SUN 公司(现被 Oracle 收购)的 HotSport，它采用的算法为分代回收。
　HotSport 将 jvm 中堆空间划分为三个代：年轻代（Young Generation）、年老代（Old Generation）和永久代（Permanent Generation）。年轻代和年老代是存储动态产生的对象。永久带主要是存储的是 java 的类信息，包括解析得到的方法、属性、字段等等。永久带基本不参与垃圾回收。我们这里讨论的垃圾回收主要是针对年轻代和年老代。具体如下图（网络提供）



　年轻代又分成 3 个部分，一个 eden 区和两个相同的 survior 区。刚开始创建的对象都是放置在 eden 区的。分成这样 3 个部分，主要是为了生命周期短的对象尽量留在年轻代。当 eden 区申请不到空间的时候，进行 minorGC，把存活的对象拷贝到 survior。年老代主要存放生命周期比较长的对象，比如缓存对象。具体 jvm 内存回收过程描述如下（可以结合上图）：
1、对象在 Eden 区完成内存分配
2、当 Eden 区满了，再创建对象，会因为申请不到空间，触发 minorGC，进行 young(eden+1survivor)区的垃圾回收
3、minorGC 时，Eden 不能被回收的对象被放入到空的 survivor（Eden 肯定会被清空），另一个 survivor 里不能被 GC 回收的对象也会被放入这个 survivor，始终保证一个 survivor 是空的
4、当做第 3 步的时候，如果发现 survivor 满了，则这些对象被 copy 到 old 区，

或者 survivor 并没有满，但是有些对象已经足够 Old，也被放入 Old 区 XX:MaxTenuringThreshold
5、当 Old 区被放满的之后，进行 fullGC。
由于 fullGC 会造成很大的系统资源开销，所以一般情况下通过代码规范和 JVM 参数设置来减少 fullGC 的调用，提高性能。
上面只是从算法实现上来说明的，目前主流的垃圾收集器主要有三种：串行收集器、并行收集器、并发收集器。 这里有兴趣的朋友可以查一下，绝对是加分点，如果在串行的垃圾回收器进行 fullGC 会造成应用的短暂中断，这是一个很危险的事情，所以一定要了解清楚各种实现方式的区别。

6.4 Java 的反射机制

解析：这个概念定义起来比较简单，但是使用却不容易，现在很多主流的开源框架中 DI,IOC 的实现很多都是用 Java 反射机制来做的，可以适当深入研究一下。

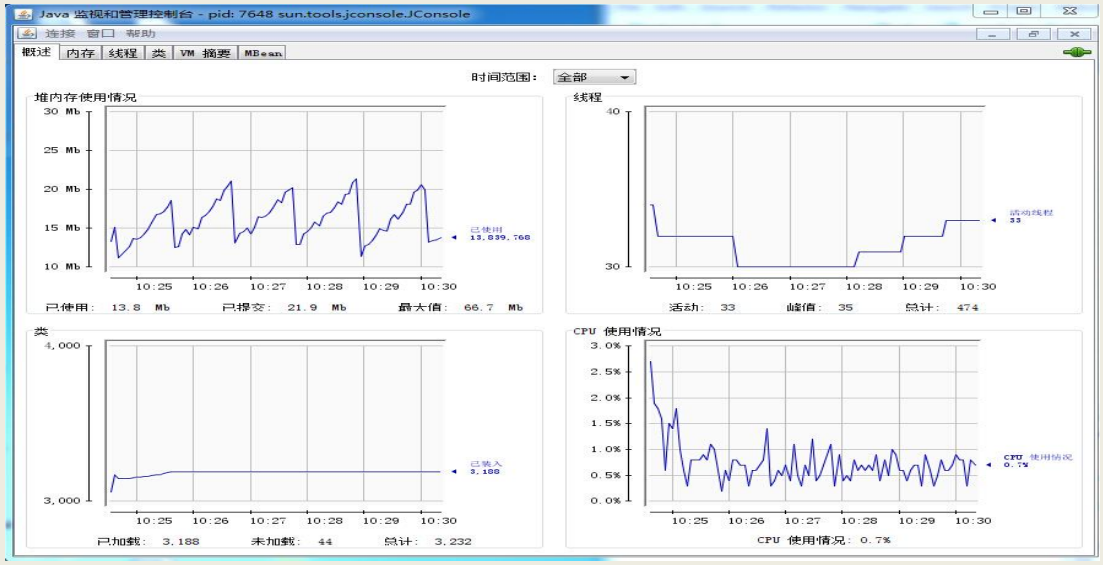
参考答案：Java 发射机制可以让我们在运行时加载，探知，使用编译期间完全未知的 classes。换句话说就是 Java 程序可以加载一个在运行时才得知名称的 class，获悉其完整构造，并生成其对象实体，或对其 fields 设值，或调用其 methods。
反射的作用：在运行的时判定任意一个对象所属的类；运行时，构造任意一个类的对象；运行时，判定一个类所属的成员变量和方法；在运行时调用任意的一个方法；生成动态代理。
反射的实现步骤（不问不需要答），1、获取类的常用方式有三种：a) Class.forName("包名.类名")，最常用、推荐；b) 包名.类名.class 最简捷；c) 对象.getClass 的方式获得。2、对象的实例化,上面已经获取了类，只需要调用类的实例化方法,类.newInstance()便可。3、获取属性和构造等，可以参考 JavaApi 的调用，类. getDeclaredFields,类. getConstructor(..)等。

6.5 JDK 性能分析工具及应用

解析：这里主要针对一定工作经验的人，而且工作经验越久可能问的越详细，主要考察对 JDK 性能的监控，用以性能调优。

参考答案：JDK 性能分析工具很多，像 **jprofiler** 及 **yourkit 是**性能十分优越，但使用费用也不低，对于开发人员的对性能这块的监控 JDK 本身就给提供了很多优秀的免费的监控工具，如 Visualvm（如果 jdk 自带版本低，可以单独下载）是一款堪比上面商业性能的监控工具、**jconsole**（下面是我的一个应用截图）能够监视和管理查看堆内存，线程，类，CPU 状况。直接双击就可以启动了，然后选择连接本地 local 还是远程 remote，而且支持控制台管理；**jstack** 主要用于线程死锁的监控；**jmap** 主要用于监控内存泄露时候对象占用的字节数；**jstat** 主要用于监控 jvm 的 gc 使用情况；**jhat** 主要用于分析 jmap 产生的 dump 并提供 web 页

面查看分析结果。



第三部分　J2EE 框架相关

第 07 章　Servlet&JSP

7.1　Servlet 的生命周期

解析：这是一个最常见的面试题目，越是大的公司越喜欢问这些基本的问题，不要轻视这些小问题，我一个资深的开发朋友就把这种张口就来的问题答错了，当时还是面试一家很大的公司，所以小的知识点也有必要梳理一下。

参考答案：Servlet 的生命周期主要为三个步骤初始化 init()、业务处理 service()、销毁 destory()。

1、init 阶段：init()方法在 Servlet 实例化的时候调用，而且只调用一次（对于 Servlet 的初始化，一般是第一次被请求时；或者在在 web.xml 中没有配置<servlet/>标签的映射<servlet-mapping/>这个时候，我们可以配置<servlet>元素中指定了<load-on-startup>子元素时，容器在启动的时候自动加载这些 Servlet 并调用 init()方法），init()方法的作用是完成一些全局性的比较花费时间的初始化工作。

2、service()阶段：Servlet 继承了父类的 service（）方法，那么前端 URL 发出的请求不管是以 get 方式或者 post 方式，都将直接走重写的 service()方法，而不再走 doGet()和 goPost()方法。

3、终止阶段调用 destroy()方法：Servlet 容器关闭时调用，用来关闭 Servlet 占用的一些资源。

7.2　Servlet3.0 有哪些新特性

解析：如果你面试一些创业型公司或者一些互联网行业公司时，这个问题很容易被问到，技术更新的快，总会涉及一些新版的特性面试，还有 JDK8 的新特性等，选择 IT 就要选择与最新技术时时同步，闲时多逛逛技术论坛还是有必要的；这里的 servlet3.0 新增的特性给我们的开发带来了不少的简便，所以更有必要去了解了。

参考答案：Servlet 3.0 作为 Java EE 6 规范体系中一员，随着 Java EE 6 规范一起发布。该版本在前一版本（Servlet 2.5）的基础上提供了若干新特性用于简化 Web 应用的开发和部署。其中有几项特性的引入让开发者感到非常兴奋：

1、异步处理支持：有了该特性，Servlet 线程不再需要一直阻塞，直到业务处理完毕才能再输出响应，最后才结束该 Servlet 线程。在接收到请求之后，Servlet 线程可以将耗时的操作委派给另一个线程来完成，自己在不生成响应的情况下返回至容器。针对业务处理较耗时的情况，这将大大减少服务器资源的占用，并且提高并发处理速度。

2、新增的注解支持：该版本新增了若干注解，用于简化 Servlet、过滤器（Filter）和监听器（Listener）的声明，这使得 web.xml 部署描述文件从该版本开始不再是必选的了。

3、可插性支持：熟悉 Struts2 的开发者一定会对其通过插件的方式与包括 Spring 在内的各种常用框架的整合特性记忆犹新。将相应的插件封装成 JAR 包并放在类路径下，Struts2 运行时便能自动加载这些插件。现在 Servlet 3.0 提供了类似的特性，开发者可以通过插件的方式很方便的扩充已有 Web 应用的功能，而不需要修改原有的应用。

4、原本文件上传时通过 common-fileupload 或者 SmartUpload，上传比较麻烦，在 Servlet 3.0 中不需要导入任何第三方 jar 包，并且提供了很方便进行文件上传的功能；

5、ServletContext 的性能增强除了以上的新特性之外，ServletContext 对象的功能在新版本中也得到了增强。现在，该对象支持在运行时动态部署 Servlet、过滤器、监听器，以及为 Servlet 和过滤器增加 URL 映射等。以 Servlet 为例，过滤器与监听器与之类似。

补充：这里只是对新增特性的简单描述，如果运用还请参阅官方资料。

7.3　Servlet 监听器的作用

解析：这是一个小命中率的面试题，往往出现率不高的问题，一旦出现时有点措手不及，回答这个问题的时候，要先了解一下 Servlet 提供了哪些常用的监听器，然后再了解它们的使用。

参考答案：

1、监控器的作用就是监听一些重要事件的发生，监听器对象可以在事情发生前、发生后可以做一些必要的处理。Servlet 提供的常用的监听器有 ServletContextListener、HttpSessionListener、ServletRequestListener 等等，这些只

是接口，如果要实现一个监听器只需要实现这些接口，同时在 web.xml 配置一下 listener 就可以了。

2、**ServletContextListener**：Servlet 的上下文监听，它主要实现监听 ServletContext 的创建和删除。该接口提供了两种方法 i.contextInitialized(ServletContextEvent event); 通知正在收听的对象，应用程序已经被加载和初始化。ii.contextDestroyed(ServletCotextEvent event); 通知正在收听的对象，应用程序已经被载出，即关闭。**HttpSessionListener**：HTTP 会话监听，该接口实现监听 HTTP 会话创建、销毁。该接口提供了一下两种方法 i.sessionCreated(HttpSessionEvent event); 通知正在收听的对象，session 已经被加载及初始化 ii. sessionDestoryed(HttpSessionEvent event)l 通知正在收听的对象，session 已经被载出（HttpSessionEvent 类的主要方法是 getSession(),可以使用该方法回传一个 session 对象）**ServletRequestListener**：该接口提供了以下两个方法。i.requestInitalized(ServletRequestEvent event) 通知正在收听的对象，ServletRequest 已经被加载及初始化 ii.requestDestroyed(ServletRequestEvent event) 通知正在收听的对象，ServletRequest 已经被载出，即关闭

3、一些使用场景可以用来统计在线人数和在线用户、统计网站访问量、系统启动时初始化信息等。

7.4 forward 与 redirect 区别

解析：这是一个基本知识的考察，是一定要回答好的题目。

参考答案：

Forward：转发，是服务器内部的一种转向行为，客户端并不能察觉，URL 显示的依然是转发前的地址;它属于一次 Request 请求，转发目标页依然可以使用 Request 范围内的数据。使用场景：多用户多角色的系统根据登录用户进行模块的跳转。

Redirect：重定向，服务器会首先响应请求端一个状态码，请求端根据状态码再次发生的请求，URL 的地址会换成后一次请求的地址；它属于两次 Request 请求，所以第一次 Request 请求范围内的数据将丢失，不能再从 Request 中获取数据。使用场景：Session 过期（或未登录时）跳转到登录页，系统异常跳转到异常页。

7.5 Session 与 Cookie 的区别

解析： Session 和 Cookie 的区别是属于基本概念的考察，难度一般，一般在 Web 应用相关职位中问到的较多。

参考答案：

Session:运行在服务器端，默认是保存在内存中，安全性高，可以存放对象，可以设置生命周期，当服务器端维护 Session 对象过多的时候，会影响到服务器的性能，可以将一部分 Session 序列化到硬盘上存储。当用户第一次请求生成 Session 对象时会生成 sessionID 用来标识此对象，SessionID 将会返回给用户保存

在 Cookie 中。

Cookie:是保存在客户端，一般不超过 4k，用户客户端对 cookie 数据量也有限制，好像 20 个左右，Cookie 安全性低，可以被改写，而且容易被浏览器禁用，但如果 Cookie 完全被禁用，Session 的会话功能也将失效。

7.6　如何实现一个自己的 session

解析：其实这是一个不难的面试题，命中率中等，因为网上相关资源不多，所以很多工作经验不多的朋友可能感觉无从下手，要答这样的题首先要了解 Session 的特点，然后就容易实现了。

参考答案：要写一个自己的 session，首先要知道 session 的几个特点：第一，能够进行对象的保存；第二，有一个唯一的识别码 sessionID 可以通过 cookie 中的 sessionID 来找到 session 对象；第三，就是可以设置 session 的有效期；解决这三点就可以来实现我们自己的 session 了；上面三个条件中最容易实现的就是 sessinID，在 Java 中生成一个不重复的 ID 太容易了，这里用 UUID 生成一个 32 位序列作为唯一识别码 token（token 就是上面的 sessionID，自己写时就不要用原来的名称了）。下面推荐三种方案，面试时根据自己的理解选择。

第一种Map 方案：

1、定义一个全局的静态的Map 对象（最好用线程安全的实现类）；

2、用户首次访问时生成一个 token 作为 Map 中的 key 值，Map 中的 value 可以根据需要定义成对象，此对象里面一定要有一个时间字段，来记录用户最近一次的访问时间；

3、定义一个后台线程，用来监控 Map 中对象的日期与系统日期时间的差值，当大于设定的时间时，就把对象从 Map 中删除(模拟 session 过期清理)。

第二种 SQL 方案：

1、建一张 t_session 表，里面的主键为 token，至少有一个日期字段，其余的字段根据保存的对象需要建立；

2、用户首次访问时生成一个 token 作为主键，同时插入一个当前日期；后续只要用户对 Session 要操作的地方，就要更新 t_session 中的日期字段；

3、定义一个数据 JOB，用来监控 t_session 中对象的日期与系统日期时间的差值，当大于设定的时间时，就把对象从 t_session 中删除(模拟 session 过期清理)。

第三种缓存方案：

1、引入一个缓存 Encache 对象；

2、用户首次访问时生成一个 token 作为 Encache 中的 key 值，Value 值可以根据需要定义对象，最好是实现了序列化。

3、在缓存配置中声明一个过期日期。

综上：第一种方案有一个问题，就是当 Map 存和的对象足够多的时候，后台线程在扫描的时候会不会造成前台用户操作 Map 对象的阻塞，从原理上是有这存情况发生，总之随着 Map 存放的东西越多，性能下降的越厉害。第二种方案不多说了，每次操作都可能引起后台数据表的操作，而且在线用户多的时候，只是

这块就会占用很多的连接数，有点浪费系统资源。第三种方案是我比较推崇的方试，不用再担心过期时间的管理，缓存本身就有过期时间管理的机制，有人担心 Encache 是与应用绑定发布的，不容易做分布式，其实多虑了，Encache 本身支持分布式的，即使有问题，我们还有 memcached 等其它缓存呢。总之根据业务场景需要来选择就行了。

7.7 Http 请求中 Get 和 Post 区别

解析：这是一个基本考察题，如果结合 http 协议和 restful 一起就会变得很复杂，在本手册中我将三部分拆开来分别说明，便于理解。

参考答案：get,post 是前台与后台交互时两种请求方式。

Get：从 URL 上看它是以明文的方式展现（一般要对参数需要加密处理）在地址栏中，而且它对提交的内容长度有限制，不能超过 1024Btye;Get 一般用于向服务器中请求数据（查询时）。

Post:是一种自动加密的请求方式，而且理论请求的内容没有长度限制，一般用于表单提交，向服务器进行数据添加或者更新的时候使用。

7.8 JSP 中动态 INCLUDE 与静态 INCLUDE 的区别

参考答案：

1、写法不同，动态包含<jsp:include page="目标页"></jsp:include>，而静态包含<%@ include file=”目标文件”%>;动态包含中一般是同样的.jsp 页，而静态包含可以是其它.txt,.html 等文件。

2、动态包含是两个独立的文件，分别编译，但它总会动态检查被引入页中的内容变化，在执行到 jsp:include 时动态引入被包含的文件，而且可以向被引入的页面中传递参数。静态包含相当于在编译前将被包含的文件插入到<%@ include file="">位置，不会动态的检查被包含文件的变化，不能传递参数，而且两部分文件会编译到一个文件中去。

3、jsp 动态包含，侧重于功能引用，比如定义一个分页组件；而静态包含侧重于页面结构布局，比如用来划分页面版块。

7.9 Servlet 是否线程安全并解释原因

解析：这个问题回答起来可深可潜，Servlet 和 Struts1 都是单实例支持多线程请求的，处理多个请求同时访问。servlet 依赖于一个线程池来服务请求。线程池实际上是一系列的工作者线程集合，Servlet 使用一个调度线程来管理工作者线程，所以大家要详细深程度了解就要先了解一下工作线程，调度线程等概念（这里不详细介绍了）。

参考答案：

Servlet 工作机制是单实例提供多线程的服务，它是通过一个调度线程来维护多个工作线程的有序执行；但由于多个线程共享一份实例，在对公共资源进行访问时一定会出现线程不同步的问题，即 Servlet 是线程不安全的。

回答完上面几乎就可以了，如果想多表示一下你的能力，可以多说一下，比如单例的好处？Servlet 容器默认采用单实例多线程的方式来处理请求，这样减少产生 Servlet 实例的开销，提升了对请求的响应时间。

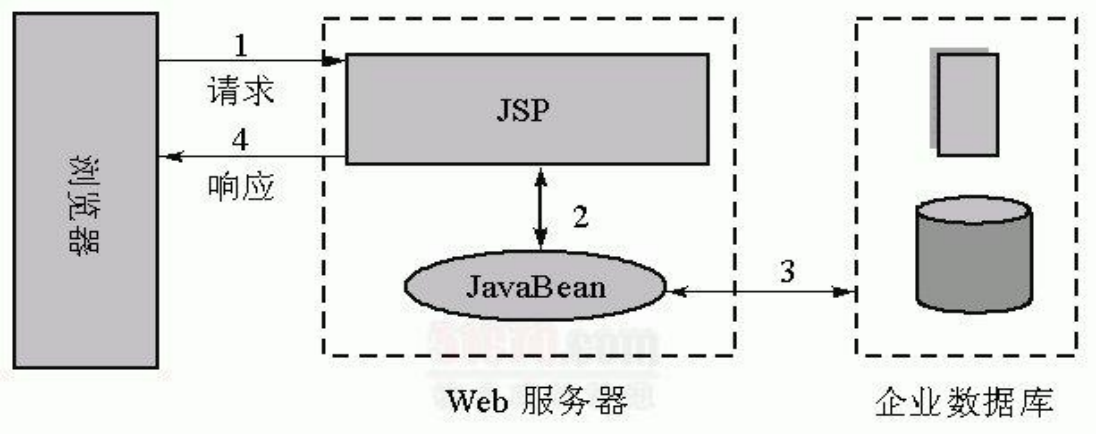
如果继续问，如何避免产生线程不安全问题呢？其实方案很多，这里说一下通常用的方式；第一，避免使用全局变量或者静态变量，这里全局变量或者静态变量在多线程情况下可能就是被争抢的公共资源。第二，采用同步锁来保证线程安全，尽可能减小锁控制，这种方式可能会使性能下降不少。第三，把 Servlet 改成单例单线程执行，这样一来，servlet 可能没什么性能而言了。建议第一种方式。

第 08 章　Struts1.x 框架

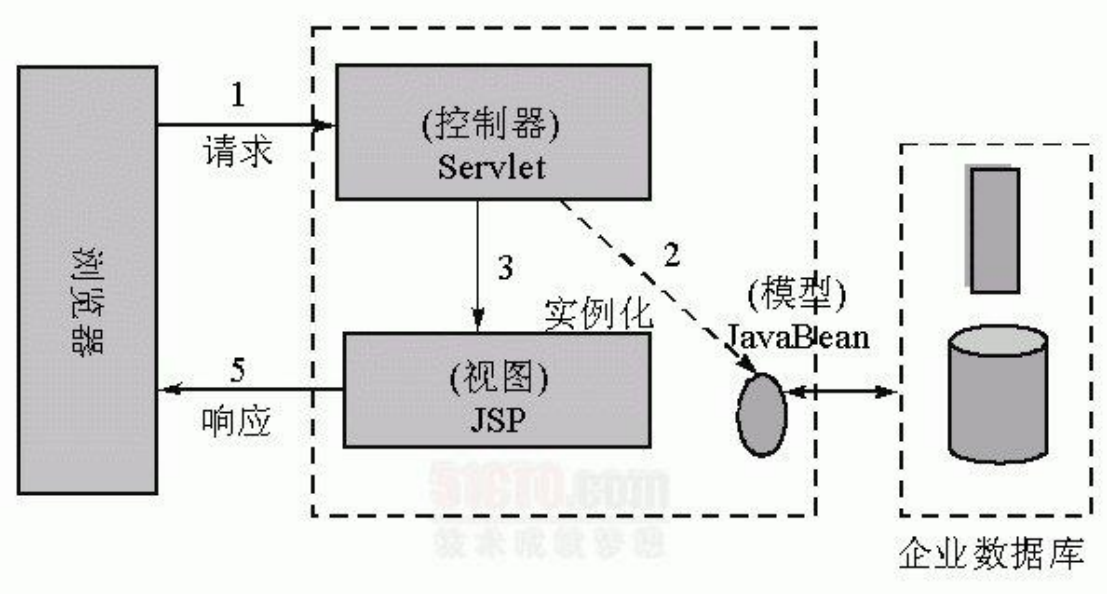
随着 Java 技术的飞速发展，Struts1.x 框架逐渐在成为历史，现在的新的项目一般不会再选用 struts1.x 去搭建框架，培训学校也不再给 struts1.x 的框架课程；但是这并不代表 struts1.x 框架就消失了，现在很多面试都会提及，尤其是一些遗留的老项目还在维护，所以你也不确定你以后会负责哪块工作，面试时还是有必要多准备一下。

8.1　Struts1 的 MVC 实现

解析：MVC 模式是现在开发中搭建框架主要设计模式，如果是有过几年工作经验的求职者还可能接触过 JspModel1，JspModel2 的设计模式（老项目维护还有可能遇到），现在看来它有不少缺点，层次不明确，业务逻辑与视图很难分离，给维护带来很大的成本，而且不易管理和分工协作。从网上看到了很具代表的两个图，分享大家了解一下，无论是面试还是以后维护到，相信对大家都会有帮助。



（JspModel1）



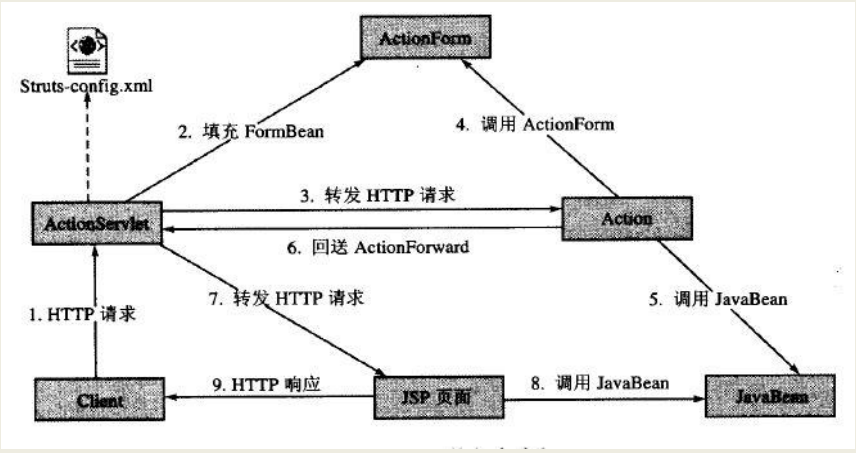
（JspModel 2）

参考答案：Struts1.x 的 MVC(Model View Controller)的实现和上面的 JspModel2 很相似的。Struts1.x 的 Model 层也是由 JavaBean 充当的(PO 层)；View 层主要是由一些前端显示来充当，如 JSP（这里把 ActionForm 有时也划到 View）层；控制层 Controller 在 Struts1.x 中也有两部分，一是核心控制器是 ActionServlet，另一个是逻辑流程控制器是 Action 类。

8.2　Struts1 的工作原理

解析：Struts1 的工作原理应该包括 Struts 框架的初始化+Struts 交互原理两部分来说明，而根据笔者的经验大多数的面试官想问的是 Struts 的交互原理；本版本中暂时不讨论 Struts 底层初始化这块，其实它的底层就是一个 Servelt，有兴趣的朋友可以多研究一下，或者关注第二个版本的指导手册。

参考答案：首先看一张官方提供的流程交互的图，如果你面试中不能用文字清晰的表述你的观点，可以要求纸笔画图，而且如果你画的很清晰的话，对面试成绩很有加分。



1、在 web 应用启动时，加载并初始化 ActionServlet，ActionServlet 从 struts-config.xml 文件中读取配置信息，将它们存放到各个配置对象中。

2、当 ActionServlet 接收到一个 Client 客户请求时，首先检索和用户请求相匹配的 ActionMapping 实例，如果不存在，就返回用户请求路径无效信息。

3、如果 ActionForm 实例不存在，就创建一个 ActionForm 对象，把客户提交的表单数据保存到 ActionForm 对象中。

4、根据配置信息决定是否需要验证表单，如果需要，就调用 ActionForm 的 validate()方法，如果 ActionForm 的 validate()方法返回 null 或返回一个不包含 ActionMessage 的 ActionErrors 对象，就表示表单验证成功。

5、ActionServlet 根据 ActionMapping 实例包含的映射信息决定请求转发给哪个 Action，如果相应的 Action 实例不存在，就先创建一个实例，然后调用 Action 的 execute()方法，与后台的 JavaBean 交互。

6、Action 的 execute()方法返回一个 ActionForward 对象，ActionServlet 再把客户请求转发给 ActionForward 对象指向的 JSP 组件。

7、ActionForward 对象指向的 JSP 组件生成动态网页，返回给客户。

8.3 为什么要用 Struts1

解析：这个问题其实对于五年往上的朋友来说，几乎很少有人问了，但对于五年以下的求职者来说，问的还是比较多的，这样都属于基础题型，回答失败的话很影响面试成绩，下面的每一个开源框架笔者都尽可能带着大家一起梳理一下为什么要选择用。

参考答案：Struts 是 Apache 基金会 Jakarta 项目组的一个 Open Source 项目，它采用 MVC 模式，能够很好地帮助 java 开发者利用 J2EE 开发 Web 应用。和其他的 java 架构一样，Struts 也是面向对象设计，将 MVC 模式"分离显示逻辑和业务逻辑"的能力发挥得淋漓尽致。Struts 框架的核心是一个弹性 struts 的控制层，基于如 Java Servlets，JavaBeans，ResourceBundles 与 XML 等标准技术，以及 Jakarta Commons 的一些类库。Struts 由一组相互协作的类（组件）、Servlet 以及 jsp tag lib 组成。基于 struts 构架的 web 应用程序基本上符合 JSP Model2 的设计标准，可以说是一个传统 MVC 设计模式的一种变化类型。而且它对改进了传统模式中存在的很多缺点，比如 JSPModel1 的耦合性，层次不清晰；JSPModel2 的

代码重复量大，维护成本高。Struts 在传承 MVC 优点的同时，而且还定义了很多常用的开发组件，比如参数验证、国际化处理等。还有一个很重要的原因，就是 Struts1 积累了大量的用户，对于各种复杂的情况都有成熟的解决方案，可以降低项目的研发成本。

8.4 Struts1 的优缺点

解析：对于刚刚毕业的同学来说，可能对于 struts1 已经很默生了，现在貌似上来就学习 struts2 了，但你的面试官可能是一名 IT 老人，你要负责的项目可能是十年前的遗留项目（那时候还是 struts1 风靡的时代），所以还是要理解一些。

参考答案：虽然 strtus1 已经渐渐老去了，但回顾一下，发现 struts1 当年还真是风靡了 N 多年，给 struts2 奠定良好的基础。

Struts 优点：

1、在 JSP+Servlet+Bean 这种 model2 设计模式基础上，Struts1 是一个基于 Sun J2EE 平台的更加完善的 MVC 框架，而且它是 Apache 基金会 Jakarta 项目组的一个 Open Source 项目。

2、Strtus1 是基于 jsp+servlet 的基础上开发的，学习成本低，易学易懂，敏捷迅速；而且引用 struts-config.xml 配置来维护 MVC 各层次之间的关系，使关系更加明确，而且更容易进行团队的分工开发。

3、struts 本身自带的 validator 框架,tiles 和 jstl 标记库标记，更能为编程人员提供方便，提高编程效率。更能使整个项目结构性良好、清晰，便于维护。

4、提供了强大的标签库，对 EL,JSTL 友好支持，同时允许用户根据需要定义自己的标签库。

5、提供了国际化支持，而且能够友好的与 hibernate,spring 友好的整合。

Struts 缺点：

1、struts 的核心控制器是 ActionServlet，它是一个标准的 Servlet，HttpServletRequest 和 HttpServletResponse 是由 Servlet 容器负责实例化的，过度的依赖于容器，这将导致 Struts 的 action 类很难时行单元测试。

2、struts 中的 actionForm 组件，通常是划分在 View 层用来对前面参数封装验证，但 ActionForm 这个组件与视图中的 VO，业务中 PO 都是一样的定义，造成重复定义，冗余。

3、struts 的配置虽然带来了很多优点，但过多的配置也会给维护带来很高的成本。

8.5 Struts1 常用的标签库

参考答案：Struts 提供了五个标签库，即：HTML、Bean、Logic、Template 和 Nested。

HTML 标签：用来创建能够和 Struts 框架和其他相应的 HTML 标签交互的 HTML 输入表单。

Bean 标签：在访问 JavaBeans 及其属性，以及定义一个新的 bean 时使。

Logic 标签：管理条件产生的输出和对象集产生的循。

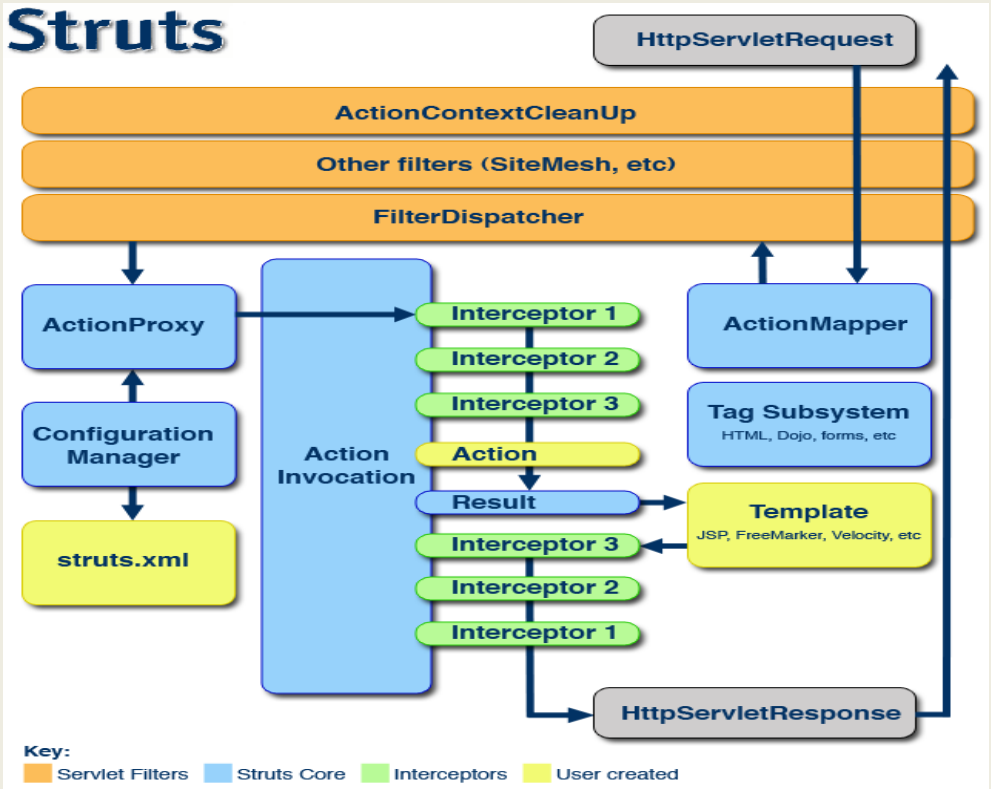
Template 标签：随着 Tiles 框架包的出现，此标记已开始减少使用。
Nested 标签：增强对其他的 Struts 标签的嵌套使用的能力。

第 09 章　Struts2.x 框架

9.1　Struts2 的工作原理

解析：Struts2 可谓站在巨人肩膀上超越，不是简单的对 struts1 的升级，而是 Struts 的下一代产品，是在 struts 1 和 WebWork 的技术基础上进行了合并的全新的 Struts 2 框架。Struts 2 以 WebWork 为核心，采用拦截器的机制来处理用户的请求，得业务逻辑控制器能够与 ServletAPI 完全脱离开，所以 Struts 2 可以理解为 WebWork 的更新产品。虽然从 Struts 1 到 Struts 2 有着太大的变化，但是相对于 WebWork，Struts 2 的变化很小。

参考答案：回答这个问题之前先来看一下 struts 官方提供的一个原理图，结合图比较容易理解，面试之前读者也可以先自己草画一下（注意图下面各种颜色对应的功能描述，这样就很容易看懂了）。



1、客户端初始化一个指向 Servlet 容器（例如 Tomcat）的请求；
2、这个请求经过一系列的过滤器（Filter）（这些过滤器中有一个叫做 ActionContextCleanUp 的可选过滤器，ActionContextCleanUp 延长 action 中属性的生命周期，包括自定义属性，而且对于 Struts2 和其他框架的集成很有帮助）；
3、接着 FilterDispatcher 被调用，FilterDispatcher 询问 ActionMapper 来决定这个请是否需要调用某个 Action ；

4、如果 ActionMapper 决定需要调用某个 Action，FilterDispatcher 把请求的处理交给 ActionProxy ；
5、 ActionProxy 通过 Configuration Manager 询问框架的配置文件，找到需要调用的 Action 类；
6、ActionProxy 创建一个 ActionInvocation 的实例。
7、 ActionInvocation 实例使用命名模式来调用，在调用 Action 的过程前后，涉及到相关拦截器（Intercepter）的调用。
8、一旦 Action 执行完毕，ActionInvocation 负责根据 struts.xml 中的配置找到对应的返回结果。返回结果通常是（但不总是，也可能是另外的一个 Action 链）一个需要被表示的 JSP 或者 FreeMarker 的模版。在表示的过程中可以使用 Struts2 框架中继承的标签。

9.2 Struts2 有哪些优点

解析：struts2 是 struts 1 和 WebWork 技术的整合，对它们是择优而取，可以说 struts1 与 webwork 的优点在 strtus2 中都得到了体现。

参考答案：struts2 是 struts 1 和 WebWork 技术的整合，拥有两大框架的精髓，细节如下；
1、struts2 依然是基于 J2EE 标准的 MVC 的框架，开发流程清晰，开发人员易学习，培训成本低。
2、struts2 与 struts1 相比解禁了与底层容器的强偶合，控制层可以是一个纯净的 JOPO 类，使得测试成本降低，可维护性增强；而且对前台的参数封装使用强大的 DI 注入方式
3、提供了 struts-default 默认的配置，最大成度上降低了开发人员的配置工作，而且支持多配置并行开发，使得大型项目分工开发更加容易。
4、引入了 OGNL 进行参数传递，OGNL 提供了在 Struts2 里访问各种作用域中的数据的简单方式，你可以方便的获取 Request，Attribute，Application，Session，Parameters 中的数据。大大简化了开发人员在获取这些数据时的代码量。
5、强大的拦截机制，struts2 中有 18 层拦截过滤设置，对于用户的登录验证、权限验证、文件上传、异常处理等有很好的处理机制，合理使用可以大大的减化工作量。
6、另外还有很多如模块化配置、多视图支持、全局结果与声明式异常配置、可扩充插件机制、对其它开源框架整合的友好支持，等等。

9.3 为什么要用 Struts2

解析：这样问其实和上面问“Struts2 的优点”提问相似，你完全可以按照上面的回答，当然也可以从设计的角度考虑一下，它们考虑使用一个框架的时候还要考虑组内人员是否熟悉，框架本身的解决方案是不是成熟（非研究性机构是不

愿意冒风险花成本去使用不成熟的东西）。总之这两个问题在面试的时候是可以互补着回答。

参考答案：先根据上面 struts2 的优点回答几点（这里省略）。
1、struts2 发展很快，积累了大量的资料和解决方案，这将降低开发中使用其它框架造成的未知风险和开发成本。
2、strtus2 支持多配置，可以对大项目进行很好的拆分，并行开发，减少开发周期。
3、struts2 是 J2EE 标准的的 MVC 框架，培训成本低，可以快速度的组建开发团队，而且由于技术通用，在后期的维护中成本也会相对低很多。

9.4 struts2 如何获取 request(session)

解析：Strtus2 默认是一个纯净的 POJO 类，是不与底层窗口偶合的，但 Struts2 中有时候也需要用到 HttpServletRequest、HttpSession、ServletContext 对象，这里便是考察 Struts2 的基本知识，如果从容器中获取这样对象？

参考答案：对于获取 HttpServletRequest、HttpSession、ServletContext 对象，Strtus2 给提供了两种方式。第一种，从 ServletActionContext 中获取,以取 Request 为例，ServletActionContext.getRequest()，返回的是一个 HttpServletRequest 对象。第二种，实现接口；Struts2 提供了 ServletRequestAware,ServletResponseAware, SessionAware,ServletContextAware 接口，重写后会提供重写方法，然后便可以在方法中获取上面对象。

9.5 Struts2 中拦截器与过滤器的区别

解析：这是一个低命中率的面试，很多工作经验丰富的朋友也很难说清楚，因为这个知识点太不引人注意了，这里网上有总结很好的回答，在这里分享一下（我们在二期版本中详细的探讨）。

参考答案：
1、拦截器是基于 java 反射机制的，而过滤器是基于函数回调的。
2、过滤器依赖于 servlet 容器，而拦截器不依赖于 servlet 容器。
3、拦截器只能对 Action 请求起作用，而过滤器则可以对几乎所有请求起作用。
4、拦截器可以访问 Action 上下文、值栈里的对象，而过滤器不能。
5、在 Action 的生命周期中，拦截器可以多次调用，而过滤器只能在容器初始化时被调用一次。
org.apache.struts2.dispatcher.FilterDispatcher 是 Struts2 的主要的 Filter，负责四个方面的功能：
(1)执行 Actions
(2)清除 ActionContext
(3)维护静态内容
(4)清除 request 生命周期内的 XWork 的 interceptors

9.6 什么是 ValueStack 和 OGNL

解析：低命中率面试，笔者参与的面试中并没有被问到过，但作为 struts2 的特色部分，还是要慎重对待。

参考答案：ValueStack（值栈）是用来处理客户端请求数据的存储区，数据一般是存放在 ActionContext 中，通过 ThreadLocal 为每个线程创建的（一个对象会对应一个 ValueStack），生命周期是请求级别。
OGNL（Object-Graph Navigation Language）：是一种功能强大的表达式语言（Expression Language，简称为 EL），通过它简单一致的表达式语法，可以存取对象的任意属性，调用对象的方法，遍历整个对象的结构图，实现字段类型转化等功能。它使用相同的表达式去存取对象的属性。

第 10 章　Hibernate 框架

10.1　Hibernate 工作原理

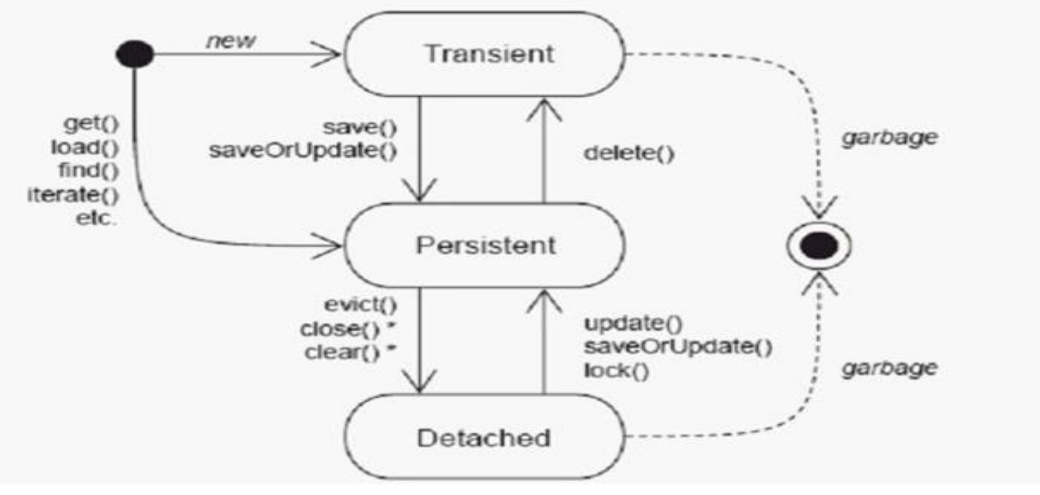
解析：Java 中所有框架的底层工作原理都很相似的，上面也解读过几个了，读者可以自己总结一下，一般系统启动时，都会先根据配置加载配置里的信息，实例化一些容器。

参考答案：
1、通过 Configuration().configure();读取并解析 hibernate.cfg.xml 配置文件(读取并解析配置文件)。
2、由 hibernate.cfg.xml 中的<mappingresource="com/xx/User.hbm.xml"/>读取解析映射信息。
3、通过 config.buildSessionFactory();（创建 SessionFactory）。
4、sessionFactory.openSession();//得到 session（打开 Session）。
5、session.beginTransaction();//开启事务（创建事务 Transation）。
6、persistent operate;
7、session.getTransaction().commit();//提交事务
8、关闭 session;
9、关闭 sessionFactory;
说明：如果记不住代码，就回答后面的注释或者括号里面的内容就行了。

10.2　简述 Hibernate 中对象的状态

解析：Hibernate 中的对象状态命中率中等，这个问题不仅仅面试中是一个考点，而且在实际开发中也是很重要的知识点，属于必懂必会的知识点。

参考答案：结合下面一官方的图来说一下 Hibernate 的三种状态，临时态（瞬时态）Transient、持久态 Persistent、游离态（脱管态）Deatched。



1、临时状态（transient）：刚刚用 new 语句创建，还没有被持久化，不处于 Session 的缓存中。处于临时状态的 Java 对象被称为临时对象。（***和 Session 没有关联，数据库中没有对应记录***）

2、持久化状态（persistent）：已经被持久化，加入到 Session 的缓存中。处于持久化状态的 Java 对象被称为持久化对象。（***和 Session 有关联，数据库中有对应记录***）

3、游离状态（detached）：已经被持久化，但不再处于 Session 的缓存中。处于游离状态的 Java 对象被称为游离对象。（***和 Session 没有关联，数据库中有对应记录***）

10.3　Load()与 Get()的区别

参考答案：Load 和 get 都是根据指定的实体类和 id 从数据库读取记录，并返回与之对应的实体对象，主要区别如下：

1、Load 支持延迟加载，而 Get 不支持延时加载。

2、对于 Get 方法，hibernate 会确认一下该 id 对应的数据是否存在，首先在 session 缓存中查找，然后在二级缓存中查找，还没有就查询数据库，数据库中没有就返回 null。

3、对于 Load 方法，load 方法加载实体对象的时候，根据映射文件上类级别的 lazy 属性的配置(默认为 true) (a)若为 true,则首先在 Session 缓存中查找，如果不存在则使用延迟加载，返回实体的代理类对象(该代理类为实体类的子类，由 CGLIB 动态生成)。等到具体使用该对象(除获取 OID 以外)的时候，再查询二级缓

存和数据库，若仍没发现符合条件的记录，则会抛出一个异常。(b)若为 false,就跟 get 方法查找顺序一样，只是最终若没发现符合条件的记录，则会抛出一个 ObjectNotFoundException。

10.4 Hibernate 缓存机制

解析：Hibernate 中缓存机制是一大特色，也是面试中的必考点；Hibernate 的缓存有一级缓存 Session 是必须知道的；而面试人员一般想让你回答的是二级缓存 SessinFactory（通常使用的是第三方缓存），它的实现比较复杂，应该了解如何配置、如何使用。

参考答案：

一级缓存：一级缓存也叫 session 级缓存或事务级缓存，一级缓存的生命周期和 Session 的生命周期一致，也就是当 session 关闭时缓存即被清除，一级缓存在 Hibernate 中是不可配置的，即不能被卸载。(Session 为应用程序提供了两个管理缓存的方法： evict(Object obj),从缓存中清除参数指定的持久化对象; clear(), 清空缓存中所有持久化对象。)

二级缓存：二级缓存也称进程级缓存或 SessionFactory 级缓存，二级缓存可以被所有的 session 共享，二级缓存的生命周期和 SessionFactory 的生命周期一致。二级缓存在 Hibernate 中是可以配置的，可以通过 class-cache 配置类粒度级别的缓存(class-cache 在 class 中数据发生任何变化的情况下自动更新)，同时也可通过 collection-cache 配置集合粒度级别的缓存(collection-cache 仅在 collection 中增加了元素或者删除了元素的情况下才自动更新，也就是当 collection 中元素发生值的变化的情况下它是不会自动更新的)。

查询缓存：查询缓存，查询的结果集也可以被缓存。只有当经常使用同样的参数进行查询时，这才会有些用处。要使用查询缓存，首先你必须打开它：hibernate.cache.use_query_cache true。该设置将会创建两个缓存区域 - 一个用于保存查询结果集(org.hibernate.cache.StandardQueryCache)；另一个则用于保存最近查询的一系列表的时间戳(org.hibernate.cache.UpdateTimestampsCache)。请注意：在查询缓存中，它并不缓存结果集中所包含的实体的确切状态；它只缓存这些实体的标识符属性的值、以及各值类型的结果。所以查询缓存通常会和二级缓存一起使用。绝大多数的查询并不能从查询缓存中受益，所以 Hibernate 默认是不进行查询缓存的。如若需要进行缓存，请调用 Query.setCacheable(true)方法。这个调用会让查询在执行过程中时先从缓存中查找结果，并将自己的结果集放到缓存中去。查询缓存在 Hibernate 同样是可配置的，默认是关闭的。

扩展:二级缓存适合放哪些数据，不适合放哪些数据？

适合放二级缓存中的数据:

1) 很少被修改的数据
2) 不是很重要的数据，允许出现偶尔并发的数据
3) 不会被并发访问的数据
4) 常量数据

不适合存放到第二级缓存的数据:

1) 经常被修改的数据

2) 绝对不允许出现并发访问的数据，如财务数据，绝对不允许出现并发
3) 与其他应用共享的数据。

10.5 Hibernate 悲观锁与乐观锁的区别

解析：悲观锁与乐观锁都是 Hibernate 控制多线程情况下数据资源安全的机制，悲观锁采用的是数据库自带锁的机制（如果数据库不支持锁机制，那悲观锁就没有意义了），而乐观锁则是 Hiberante 层实现的锁机制。

参考答案：hibernate 中经常用到当多个线程对同一数据同时进行修改的时候，会发生脏数据，造成数据的不一致性，解决办法是可以通过悲观锁和乐观锁来实现。

悲观锁：采用数据库自带的锁的机制（也只有数据库层提供的锁机制才能真正保证数据访问的排他性，否则，即使在本系统中实现了加锁机制，也无法保证外部系统不会修改数据），系统谨慎的认为每个线程的操作都可能导致数据的不一致性，于是当前的线程进行加锁处理，其它线程阻塞，直到当前线程结束释放锁（数据库执行的语句应该是 select * from table where id.. for update 操作）。

乐观锁：悲观锁虽然能够很好的保证数据一致性，但由于会造成阻塞，系统性能很低；而乐观锁是 Hibernate 级别的锁，它是乐观派通常认为多个事务同时操纵同一数据的情况是很少的，因为根本不做数据库层次上的锁定，只是基于数据的版本标识实现应用程序级别上的锁定机制，能够更好的提高系统性能。

10.6 Update()和 SaveOrUpdate()的区别

解析：如果从上面一直看下来，想必这个问题这个时候已经很容易回答了，参考（10.2 简述 Hibernate 中对象状态的图）上面 hibernate 状态的图，就可以大概了解这个问题的答案了:update 必须数据库中有对应的记录，而 saveOrUpdate 则状态转变的要求。

参考答案：update()和 saveOrUpdate()是用来对 Session 的 PO 进行状态管理的。**update()**方法操作的对象必须是持久化了的对象。也就是说，如果此对象在数据库中不存在的话，就不能使用 update()方法。

saveOrUpdate()方法操作的对象既可以使持久化了的，也可以使没有持久化的对象。如果是持久化了的对象调用 saveOrUpdate()则会更新数据库中的对象；如果是未持久化的对象使用此方法,则 save 到数据库中。

10.7 Hibernate 的优缺点

解析：Hibernate 的优缺点面试命中率很高，另外一种问就是为什么要用 Hibernate，这一问题难度不大，属于必须回答好的题目。

参考答案：

优点：

1、对象/关系数据库映射(ORM)，是一种面向对象数据库操作，将开发人员从自己写 SQL 语句中解放出来，开发人员只要基于 API 对对象的操作，就可以完成数据库层的开发。降低了开发人员的要求，提高了开发和维护的效率。

2、低侵入性设计，Hibernate 的引入没有任何侵入性，不仅可以与 Web 应用简单集成，也可以直接与 APP 应用集成使用，不依赖服务器容器，测试方便，可以随便通过 Junit,main 方法就能完成测试。

3、可移植性好，Hibernate 是基于配置开发，在对于不同的数据库进行切换时，只要简单的修改配置就可以完成，性能很高。

4、Hibernate 实现了透明持久化：当保存一个对象时，这个对象不需要继承 Hibernate 中的任何类、实现任何接口，只是个纯粹的单纯对象—称为 POJO 对象，带有持久化状态的、具有业务功能的单线程对象，此对象生存期很——这些对象唯一特殊的是他们正与（仅仅一个）Session 相关联。一旦这个 Session 被关闭，这些对象就会脱离持久化状态，这样就可被应用程序的任何层自由使用。

5、缓存机制，大大提高了 Hibernate 执行效率。

缺点：

1、Hibernate 的开发成本低，但学习成本高；Hibernate 对开发人员的要求很高，尤其在复杂的关联关系，延迟策略等开发上，很注重开发人员的能力修为。

2、维护成本高，由于 Hibernate 对 SQL 的操作是基于对象的，而 SQL 的生成机制我们很少关心，尤其是复杂对象引用嵌套时，生成 SQL 会很复杂，但运维出现问题时，你会发现跟踪问题很麻烦，有时间想找一条合适的 SQL 都很麻烦。

3、应用层次优化很难，JDBC 开发时我们可以自己来优化 SQL，Hibernate 在这块想提高性能优化是不容易实现的。

4、批量操作不容易，而且调用过程、函数等也不如 JDBC 方便。

第 11 章　Mybatis 框架

11.1　resultType 和 resultMap 的区别

解析：属于小命中率考题，难度不大，但通过率不高，最经常使用的东西往往是最易忽略它，所以有必要好好了解一下。

参考答案：MyBatis 中在查询进行 select 映射的时候，返回类型可以用 resultType，也可以用 resultMap；resultType 是直接表示返回类型的(PO 类)，而 resultMap 则是对外部 ResultMap 的引用(在 po.xml 中配置的映射 key-->value 关系)，但是 resultType 跟 resultMap 不能同时存在。

在 MyBatis 进行查询映射时，其实查询出来的每一个属性都是放在一个对应的 Map 里面的，其中键是属性名，值则是其对应的值。

当提供的返回类型属性是 resultType 时，MyBatis 会将 Map 里面的键值对取出赋给 resultType 所指定的对象对应的属性。所以其实 MyBatis 的每一个查询映射的返回类型都是 ResultMap，只是当提供的返回类型属性是 resultType 的时候，

MyBatis 对自动的给把对应的值赋给 resultType 所指定对象的属性。
当提供的返回类型是 resultMap 时，因为 Map 不能很好表示领域模型，就需要自己再进一步的把它转化为对应的对象。
由于 returnType 中返回的 PO 中的属性默认作为 key 值，所以必须保证 PO 中的属性和数据库表中的字段一致，要不就要配置映射关系。

11.2　Mybatis 中“#”与“$”区别

解析：这个问题笔者曾经的面试中就经历过两次，尤其是阿里出来的人貌似都喜欢问类似的问题（阿里人比较钟情 Spring,Mybatis 技术），所以面试前一定要问清你面试的公司和职位，查一下该公司的技术结构，做到有备而战。

参考答案：#相当于对数据加上双引号，$相当于直接显示数据。
1、#将传入的参数值都当成一个字符串，会对自动传入的数据加一个双引号。如：order by #age#，传入的值是 27,那么解析成 sql 时的值为 order by "27"，如果传入的值是 regdate，则解析成的 sql 为 order by "regdate"，它写入的是你传进去的值。
2、$将传入的属性直接显示生成在 sql 中。如：order by age，如果传入的值是 27，那么解析成 sql 时的值为 order by age，如果传入的值是 name，则解析成的 sql 为 order by regdate，它显示的是你传入的属性片段，而不是属性值。这是一个很危险的做法，如果在验证中加一个" or 1=1" 那就可能恒成立了。
3、#是可以防止语句注入的，但$却不能防止语句注入，比较危险；如果可以用#号的地方就尽量不要用$。

第 12 章　Spring 框架

12.1　为什么要用 Spring

解析：由于 Spring 内容比较多，直接这样被问到，如果没有做过任何准备，很容易让人犯晕，拆分一下其实也不是很难回答。为什么要用？首先可以谈一下作者设计 Spring 的初衷，然后谈一下 Spring 的优点，最后谈一下 Spring 带来的优越性。

参考答案：Rod Johnson 在设计 Spring 目的是，为了解决企业应用开发的复杂性，让 J2EE 开发更容易。
Spring 设计上的优点：
轻量级：相对于 EJB 这种重量级的容器而言，Spring 的 IOC 是完全不依赖底层容器，零侵入性的设计。便于开发测试，相对于 EJB 而言，Spring 部署方便，而且可以运行在任何 J2ee 支持的容器上，或者 APP 中。

控制反转（IOC）：Spring 使用控制反转技术实现了松耦合，依赖被注入到对象，而不是创建或寻找依赖对象。

面向切面（AOP）：解决了面向对象中不足，Spring 支持面向切面编程，同时把应用的业务逻辑与系统的服务分离开来。

事务管理：Spring 强大的事务管理功能，支持声明事务和编程事务，能够处理本地事务(一个数据库)或是全局事务(多个数据，采用 JTA)。

异常处理：由于 Java 的 JDBC，Hibernate 等 API 中有很多方法抛出的是 checked exception，而很多开发者并不能很好的处理异常。Spring 提供了统一的 API 将这些 checked exception 的异常转换成 Spring 的 unchecked exception。

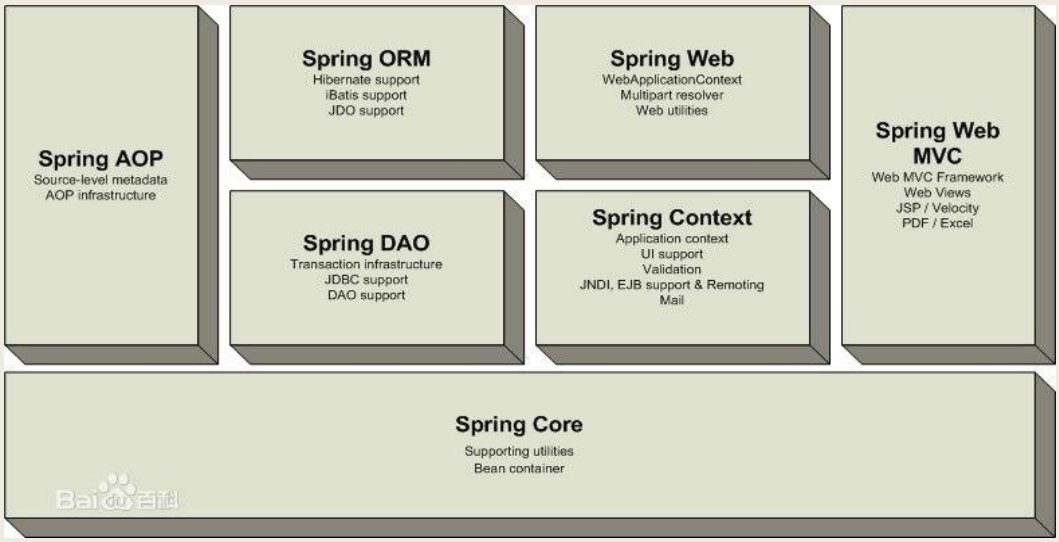
SpringMVC 框架：Spring 实现了一个自己的 MVC 框架，性能优越性很高，可以替换 struts2，而且无需要考虑与 spring 整合问题。

由于 Spring 的优越性，得到了很大的用户青睐，尤其像阿里这样的大公司很注重 Spring 的应用。

12.2　简述一下 Spring 结构

解析：这个问题命中率不是很高，而且难度不算太大，只要了解 Spring 构架结构图，就 OK 了，面试官主要考查你是不是对 Spring 结构了解。

参考答案：这个问题的答案就是用语言把下面这张图描述出来（把下面加黑的回答出来四五了就行了）。



1、核心容器

这是 Spring 框架最基础的部分，它提供了依赖注入（DependencyInjection）特征来实现容器对 Bean 的管理。这里最基本的概念是 BeanFactory，它是任何 Spring 应用的核心。BeanFactory 是工厂模式的一个实现，它使用 IoC 将应用配置和依赖说明从实际的应用代码中分离出来。

2、应用上下文（Context）模块

核心模块的 BeanFactory 使 Spring 成为一个容器，而上下文模块使它成为一个框架。这个模块扩展了 BeanFactory 的概念，增加了对国际化（I18N）消息、

事件传播以及验证的支持。

3、Spring 的 AOP 模块

Spring 在它的 AOP 模块中提供了对面向切面编程的丰富支持。这个模块是在 Spring 应用中实现切面编程的基础。Spring 的 AOP 模块也将元数据编程引入了 Spring。使用 Spring 的元数据支持，你可以为你的源代码增加注释，指示 Spring 在何处以及如何应用切面函数。

4、JDBC 抽象和 DAO 模块

使用 JDBC 经常导致大量的重复代码，取得连接、创建语句、处理结果集，然后关闭连接。Spring 的 JDBC 和 DAO 模块抽取了这些重复代码，因此你可以保持你的数据库访问代码干净简洁，并且可以防止因关闭数据库资源失败而引起的问题。

5、对象/关系映射集成模块

对那些更喜欢使用对象/关系映射工具而不是直接使用 JDBC 的人，Spring 提供了 ORM 模块。Spring 并不试图实现它自己的 ORM 解决方案，而是为几种流行的 ORM 框架提供了集成方案，包括 Hibernate、JDO 和 iBATIS SQL 映射。Spring 的事务管理支持这些 ORM 框架中的每一个也包括 JDBC。

6、Spring 的 Web 模块

Web 上下文模块建立于应用上下文模块之上，提供了一个适合于 Web 应用的上下文。另外，这个模块还提供了一些面向服务支持。例如：实现文件上传的 multipart 请求，它也提供了 Spring 和其它 Web 框架的集成，比如 Struts、WebWork。

7、Spring 的 MVC 框架

Spring 为构建 Web 应用提供了一个功能全面的 MVC 框架。虽然 Spring 可以很容易地与其它 MVC 框架集成，例如 Struts，但 Spring 的 MVC 框架使用 IoC 对控制逻辑和业务对象提供了完全的分离。

12.3　什么是 IOC（DI）

解析：IoC（Inversion of Control，控制倒转）这个是 Spring 核心，面试的必问问题，也是必须会的。

参考答案：IoC（Inversion of Control，控制倒转）是 Spring 核心，它将之前需要在业务中维护的对象依赖关系交给 Spring 容器去管理，开发人员不需要去创建对象，只要在需要调用的地方直接去从容器中取就可以了，而这种原来由自己维护对象关系现在交给 Spring 容器来管理的方式就叫做 IOC。

12.4　AOP 的实现原理

解析：AOP 面向方面编程，是 Java 面向对象编程的一个补充，AOP 使用场景常用在日志记录、事务管理、性能监测、安全认证等方面，而且 AOP 有很多专有名词切点、连接点、织入、切面等都需要好好掌握，这里主要介绍 AOP 的实现原理。

参考答案：AOP 是一种面向过程的编程思想，它的实现原理主要是 Java 的动

态代理，而 Spring 对动态代理的支持有两种方式，一种是默认的 JDK 实现，一种是 CGLIB 实现。

1、JDK 动态代理

Jdk 的动态代理是 Sping 的默认认实现，它要求必要是基于接口，代理类与被代理类必须实现同一个接口。

2、CGLIB 动态代理

Cglib 在 spring 的支持是需要在配置中指定（默认是通过 JDK 的方式实现），它不要求基于接口，它是通过子类继承的方式，利用回调来实现代理的。

12.5 Spring 的生命周期

解析：Spring 的生命周期其实就是 Bean 的生命周期，从 Bean 开始被装载，然后默认的以单例形式实例化，然后是属性注入，然后是被装载到 BeanFactory，然后调用销毁方法，当然这只是大的步骤，里面有一系列的细致化步骤，方法名都很长，很难记得清楚，所以要在理解的基础上大致说一下主要流程就可以了，不要去死记（这里转一个网上的总结，二期版本中我会以图形式画出来）。

参考答案：Spring 的生命周期其实就是 Bean 的生命周期，过程大概如下

1.容器启动，实例化所有实现了 BeanFactoyPostProcessor 接口的类。他会在任何普通 Bean 实例化之前加载.

2.实例化剩下的 Bean，对这些 Bean 进行依赖注入。

3.如果 Bean 有实现 BeanNameAware 的接口那么对这些 Bean 进行调用

4. 如果 Bean 有实现 BeanFactoryAware 接口的那么对这些 Bean 进行调用

5. 如果 Bean 有实现 ApplicationContextAware 接口的那么对这些 Bean 进行调用

6. 如果配置有实现 BeanPostProcessor 的 Bean，那么调用它的 postProcessBeforeInitialization 方法

7.如果 Bean 有实现 InitializingBean 接口那么对这些 Bean 进行调用

8.如果 Bean 配置有 init 属性，那么调用它属性中设置的方法

9. 如果配置有实现 BeanPostProcessor 的 Bean，那么调用它的 postProcessAfterInitialization 方法

10. Bean 正常是使用

11.调用 DisposableBean 接口的 destory 方法

12.调用 Bean 定义的 destory 方法

如果从大体上区分值分只为四个阶段

1. BeanFactoyPostProcessor 实例化

2. Bean 实例化，然后通过某些 BeanFactoyPostProcessor 来进行依赖注入

3. BeanPostProcessor 的调用.Spring 内置的 BeanPostProcessor 负责调用 Bean 实现的接口: BeanNameAware, BeanFactoryAware, ApplicationContextAware 等等，等这些内置的 BeanPostProcessor 调用完后才会调用自己配置的 BeanPostProcessor

4. Bean 销毁阶段

说明，如果感觉参考答案步骤太多难回答，可以只回答里面的下面四条，如果下

面四条你嫌单词太长记不住，就回答我在解析里面的几个小步骤就行了。

12.6　BeanFactory 和 ApplicationContext 区别

参考答案：

BeanFactory 负责读取 bean 配置文档，管理 bean 的加载，实例化，维护 bean 之间的依赖关系，负责 bean 的声明周期。

ApplicationContext 除了提供上述 BeanFactory 所能提供的功能之外，还提供了更完整的框架功能：国际化支持、资源访问、事件传递等。

12.7　Spring 实例化 Bean 的方式

解析：这个属于中等命中率的题目，Spring 实例化 Bean 的方式一般是通过三种试来实现的，一般情况都能说出来一两种，需要在理解的基础上进行掌握。

参考答案：

1、通过构造来实例化 Bean，这也是 Spring 中最常用的，被实例化的 B ean 一定要有默认的构造方法，如果是有参构造，必须在 bean 的配置中指定构造参数，否则 bean 将不能被实例化。

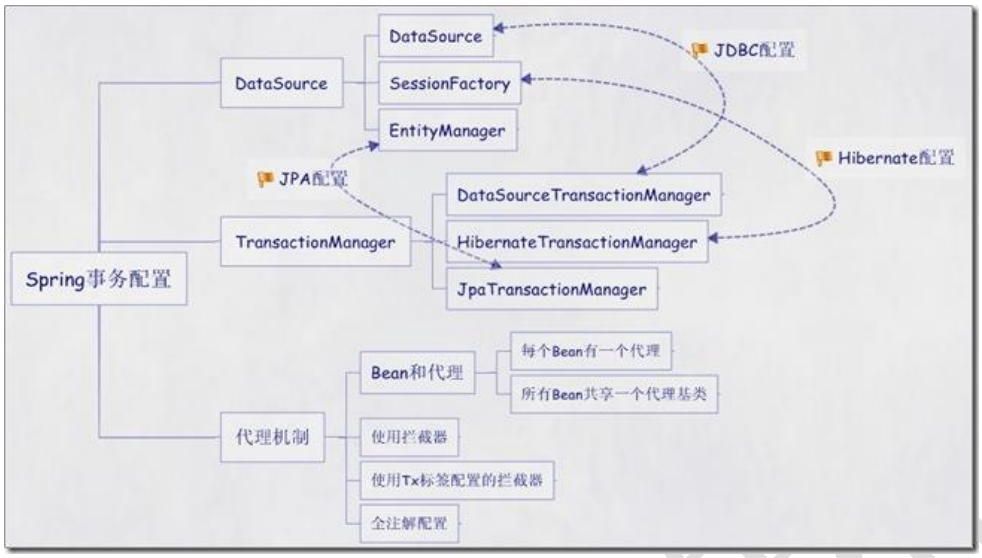
2、静态工厂，这个只需要在 bean 中指定属性 factory-method：参数为静态工厂类的生产对象的静态方法。

3、实例工厂，这个和静态工厂很相似，一般在 bean 中需要批定两个属性 factory-bean：工厂类的实例； factory-method：工厂类生产对象的方法。

说明：你面试时按照上面的回答，肯定没有问题，但是很抽象，最好结合实例先理解然后面试中回答时就会更自然。

12.8　Spring 各种事务实现及区别

解析：这个是 Spring 面试题中命中率相当高的题，而且不好回答，一定要注意听清面试官的问题，然后回答，对于 Spring 事务的实现，可以通过下面的一副图来说明，你能把图说清楚了(其实这个图中并没有完全包括，还有比如 JDO 的支持，还有对 Apache 的 OJB 作支持化时的支持等，不过用作面试应该够了，在版本 II 中还会详细讨论这个问题)，这个问题也就回答好了。



从图中看 Spring 的事务配置包括三部分：DataSource,TransactionManager,代理机制。三种 DataSource 对应三种 TransactionManager 事务管理器，Spring 的五种事务配置其实这里指的是它的代理机制，关于代理机制这里目前不是我们讨论的重点，在面试中可以侧重它的事务支持来回答。

参考答案：Spring 常用事务及实现原理，JDBC DataSource 中的事务是通过 Connnection 来实现的，Hibernate 事务是用 session 来实现的，JTA 的事务一般是通过 JNDI 来实现的。除此外 JDBC 事务和 Hibernate 事务是对单一数据库操作，通常不会同一个 Connection 来操作两个数据库，而 Hibernate 中一般一个 SessionFactory 对应一个数据库，更不会在一个 session 中操作两个数据库了；如果如果是访问不同的资源，上面的事物就不能满足了，这时必须选择 JTA 事务。而分布式部署则要使用 XA 事务才行。

12.9 Spring 事务属性

分析：Spring 事务属性一般会和 Spring 事务一起考，也会有像现在这样分开来问的，Spring 事务属性要从两方面来回答，一个传播特性一个是隔离级别。这个问题回答起来可能很多，如果没有理解靠记忆是不太容易的，不过实际面试中很少有人都能够回答的全面，如果英文单词记不住，就描述一下后面的中文解释。

参考答案：

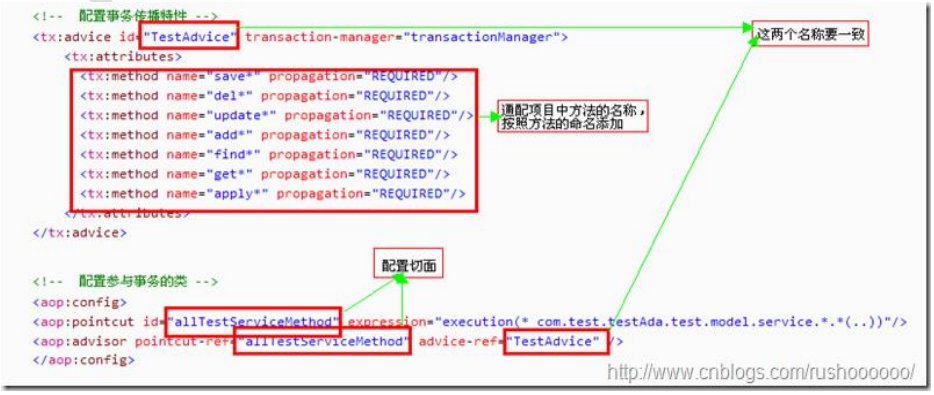
事务的几种传播特性:

1. PROPAGATION_REQUIRED: 如果存在一个事务，则支持当前事务，如果没有事务则开启。
2. PROPAGATION_SUPPORTS: 如果存在一个事务，支持当前事务。如果没有事务，则非事务的。
3. PROPAGATION_MANDATORY: 如果已经存在一个事务，支持当前事务。如果没有一个活动的事务，则抛出异常。
4. PROPAGATION_REQUIRES_NEW: 总是开启一个新的事务。如果一个事务已经存

在，则将这个存在的事务挂起。
5. PROPAGATION_NOT_SUPPORTED: 总是非事务地执行，并挂起任何存在的事务。
6. PROPAGATION_NEVER: 总是非事务地执行，如果存在一个活动事务，则抛出异。
7. PROPAGATION_NESTED：如果一个活动的事务存在，则运行在一个嵌套的事务中. 如果没有活动事务, 则按 TransactionDefinition.PROPAGATION_REQUIRED 属性执行。
Spring 事务的隔离级别:
1. ISOLATION_DEFAULT：这是一个 PlatfromTransactionManager 默认的隔离级别，使用数据库默认的事务隔离级别.
另外四个与 JDBC 的隔离级别相对应
2. ISOLATION_READ_UNCOMMITTED：这是事务最低的隔离级别，它充许令外一个事务可以看到这个事务未提交的数据。
这种隔离级别会产生脏读，不可重复读和幻像读。
3. ISOLATION_READ_COMMITTED：保证一个事务修改的数据提交后才能被另外一个事务读取。另外一个事务不能读取该事务未提交的数。
4. ISOLATION_REPEATABLE_READ：这种事务隔离级别可以防止脏读，不可重复读。但是可能出现幻像读。
它除了保证一个事务不能读取另一个事务未提交的数据外，还保证了避免下面的情况产生(不可重复读)。
5. ISOLATION_SERIALIZABLE 这是花费最高代价但是最可靠的事务隔离级别。事务被处理为顺序执行,除了防止脏读，不可重复读外，还避免了幻像读。

12.10 Spring 编程事务与声明事务的区别

分析：Spring 中的面试题难度一般都比较大，这也是因为 Spring 优秀的原因之一，底层帮你实现的东西越多，用着越方便，越能体现其优越性，但偶尔还是要多做一下底层的研究，以便更好去使用 Spring 框架。关于这个问题，面试时命中率属于中等，可以结合着“12.8Spring 各种事务实现及区别”来回答，上面其实已经在解释声明式事务在 Spring 中的使用，它需要配置数据源(jdbc,sessionfactory,jndi 的等)，需要一个事务管理类，需要代理机制对业务 bean 进行事务管理（回答这些其实声明事务就已经 Ok 了），下面提供一个网络上的图来说明常用的声明事务的配置方式（之一）。



参考答案：

编程式事务一般的步骤是，开启手动事务、处理业务、提交事务、异常中回滚事务、finally 中关闭事务，在代码中需要手动完成；但我们这里提到的是 Spring 中的编程事务，所以一般的做法是在 bean.xml 中声明一个事务类型、声明一个事务模板（注入声明的事务）、在业务 Service 中使用我们的事务模板。编程事务使用场景一般是事务比较简单且业务不复杂的的应用中推荐。**编程事务的优点：容易控制边界，粒度细；缺点是，侵入性强。**

声明式事务方式一（通过使用代理增加的方式配置）：创建一个代理的工厂 Bean(如 TransactionProxyFactoryBean)，然后设置业务类接口属性，然后注入接口的实现类，然后注入事务管理器，然后注入事务属性（在上面已经提到，配置事务的隔离级别和传播特性）；

方式二（常用的），声明一个数据源，声明一个事务管理类，声明一个 Advice（设置一些 Spring 事务属性）,定义切面指定事务的作用范围。**声明事务的优点是，侵入性性配置方便，缺点是边界不容易控制**。

12.11　SpringMVC 的工作原理

解析：SpringMVC 现在可以和 Struts2 并列成为当前最常用的 Web 项目框架，所以面试提到也将会很多，高命中率，中等难度，而且它的原理和使用都比较简单，它的核心是 DispatcherServlet。

参考答案：

1. 用户向服务器发送请求，请求被 Spring 前端控 DispatcherServlet 捕获。
2. DispatcherServlet 对请求 URL 进行解析，得到请求资源标识符（URI）。查询一个或者多个 HandlerMapping，找到处理请求的 Controller。
3. 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，数据转换、数据格式化、数据验证等工作，然后提交到相应的 Controller。
4. Handler 执行完成后，向 DispatcherServlet 返回一个 ModelAndView 对象。
5. 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。
6. ViewResolver 结合 Model 和 View，来渲染视图。
7. 将渲染结果返回给客户端。

说明：这里只是简略的回答，如果细致起来，可能很长。比如 SpringMVC 的核心是 DispatcherServlet，它除了截取 URL 请求外，还初始化上下文 WebApplicationContext，并将其与业务层和持久化层建立关联，并且装配 SpringMVC 中的各个组件等，有兴趣可以多研究点。

第 13 章　WebService 技术

13.1　什么是 WebService

解析：看到这个问题时，估计大家都会想到另外一个问题“什么是 J2EE”，这类的问题都不容易说清楚，官方定义往往都很简单，我们往往借助一下它的作用来给它做一些解释，或者根据自己的理解来解释，只要你能说清楚它是什么就行了。

参考答案：百度百科的定义是，Web service 是一个平台独立的，低耦合的，自包含的、基于可编程的 web 的应用程序，可使用开放的 XML（标准通用标记语言下的一个子集）标准来描述、发布、发现、协调和配置这些应用程序，用于开发分布式的互操作的应用程序。

我们也可以简单总结一下，**Web service 它是一种特殊的 Web 应用程序，它基于 Saop 协议，用于不同平台之间相互通信的数据接口**。

W3C 组织对其的定义如下，它是一个软件系统，为了支持跨网络的机器间相互操作交互而设计。Web Service 服务通常被定义为一组模块化的 API，它们可以通过网络进行调用，来执行远程系统的请求服务。

拓展：很多著名论坛说以后的程序员将有 80%的人是做接口服务的开发工作，这点是不夸张的，现在的应用注重前后端分离，所以以后很多人是会转型到后台写接口开发工作，而 WebService 是后台接口开发非常成熟的标准（它本身还不是框架），虽然还未完全通过 w3c 的认证，但有微软，IBM，Borland 这些大公司的支持，现在发展已经很快，已经用到了各个领域之中，所以多花点时间研究这块是很有必要的。

WebService 的优缺点：优点是跨平台，跨语言，跨系统。但缺点也很明显，它是基于 Saop 协议=http+xml，将数据转换成 XML 封装，由于 xml 用到很多标签来规范格式，这将造成传输入的数据包很大，性能很低。另外就是 WebService 并未完全经过 w3c 的认证，很多规范细则没有制定，也使不同的实现平台会出现一些不兼容问题。

13.2　WebService 常用实现引擎并说明

解析：不要被面试官的官方语言问晕了，所谓实现引擎就是问实现框架，可以把 WebService 看作 J2ee 一样的概念，它只是一个技术或者说平台规范，真正的项目应用时，是使用基于它规范实现的框架，即所谓引擎。WebService 常用的实现引擎为 Xfire,Axis,Axis2,Cxf,Jdk 自带的支持，笔者以前就各种引擎的实现专门写了一份文档，有时间会分享给大家。下面的分析，可以有助于理解，读者可以选择性看。

XFIRE

是 codeHaus 组织提供的一个开源框架，webservice 引擎中比较简单的一个，配置非常简单，而且很容易很容易和 Spring 作集成，开发效率高。

适合场景：XFIRE 框架从 2007 年后开发团队就不再对这个项目进行维护，它的升级版本就是现在比较热的 CXF 框架；但 XFIRE 还是比较成熟的经典框架，对于业务量不大，项目不大而且要求快速度开发的项目还是可以使用。

缺点：该框架不再更新和维护、仅对 Java 项目支持。

AXIS2

Axis2 是下一代 Apache Axis。Axis2 虽然由 Axis 1.x 处理程序模型提供支持，但它具有更强的灵活性并可扩展到新的体系结构。Axis2 基于新的体系结构进行了全新编写，而且没有采用 Axis 1.x 的常用代码。支持开发 Axis2 的动力是探寻模块化更强、灵活性更高和更有效的体系结构，这种体系结构可以很容易地插入到其他相关 Web 服务标准和协议（如 WS-Security、WS-ReliableMessaging 等）的实现中。Apache Axis2 是 Axis 的后续版本，是新一代的 SOAP 引擎。
主要特点：
一、采用名为 AXIOM（AXIs Object Model）的新核心 XML（标准通用标记语言的子集）处理模型，利用新的 XML 解析器提供的灵活性按需构造对象模型。
二、支持不同的消息交换模式。目前 Axis2 支持三种模式：In-Only、Robust-In 和 In-Out。In-Only 消息交换模式只有 SOAP 请求，而不需要应答；Robust-In 消息交换模式发送 SOAP 请求，只有在出错的情况下才返回应答；In-Out 消息交换模式总是存在 SOAP 请求和应答。
三、提供阻塞和非阻塞客户端 API。
四、支持内置的 Web 服务寻址 (WS-Addressing) 。
五、灵活的数据绑定，可以选择直接使用 AXIOM，使用与原来的 Axis 相似的简单数据绑定方法，或使用 XMLBeans、JiBX 或 JAXB 2.0 等专用数据绑定框架。
六、新的部署模型，支持热部署。
七、支持 HTTP，SMTP，JMS，TCP 传输协议。
八、支持 REST (Representational State Transfer)。
AXIS2 优缺点：优点是该框架成熟稳定支持多种语言开发，缺点是开发环境部署麻烦，当然这是相对的。

CXF

Apache CXF = Celtix + XFire，开始叫 Apache CeltiXfire，后来更名为 Apache CXF 了，以下简称为 CXF。CXF 继承了 Celtix 和 XFire 两大开源项目的精华，提供了对 JAX-WS 全面的支持，并且提供了多种 Binding 、DataBinding、Transport 以及各种 Format 的支持，并且可以根据实际项目的需要，采用代码优先（Code First）或者 WSDL 优先（WSDL First）来轻松地实现 Web Services 的发布和使用。Apache CXF 已经是一个正式的 Apache 顶级项目。
功能特性：

CXF 包含了大量的功能特性，但是主要集中在以下几个方面：
支持 Web Services 标准：CXF 支持多种 Web Services 标准，包含 SOAP、Basic Profile 、 WS-Addressing 、 WS-Policy 、 WS-ReliableMessaging 和 WS-Security 。
Frontends：CXF 支持多种“Frontend”编程模型，CXF 实现了 JAX-WS API （遵循 JAX-WS 2.0 TCK 版本），它也包含一个“simple frontend”允许客户端和 EndPoint 的创建，而不需要 Annotation 注解。CXF 既支持 WSDL 优先开发，也支持从 Java 的代码优先开发模式。容易使用： CXF 设计得更加直观与容易使用。有大量简单的 API 用来快速地构建代码优先的 Services，各种 Maven 的插件也使集成更加容易，支持 JAX-WS API ，支持 Spring 2.0 更加简化的 XML 配置方式，等等。支持二进制和遗留协议：CXF 的设计是一种可插拔的架构，既可以支持 XML ，也可以支持非 XML 的类型绑定，比如：JSON 和 CORBA。
JDK6.0+

JDK6+以上的版本对 WebService 的开发，被认为是最简单的开发方式（在发

布时只有 WebLogic 直接支持 JAX-WS2 的直接发布环境，tomcat,jboss 都需要引入这个环境才能发布，下面会仔细介绍），而且客户端开发也非常简单。

参考答案：**WebService 常用的实现引擎有 Xfire,Axis,Axis2,Cxf,Jdk 自带的支持。**

XFIRE

是一个小巧灵活，配置简单，容易集成的框架，但官方已经停止了对它的维护。适用场景，对于业务量不大，项目不大而且要求快速度开发的项目还是可以使用。

AXIS2

Axis2 是 axis1 的升级版本，是一款历史久版本稳定的非常优秀、支持语言最多的框架。对于 Java 开发也有很好的支持，也是 WebService 开发者选用最多的框架。但对于 Java 开发者来说，环境搭建比较麻烦（当然这只是相对的）。

CXF

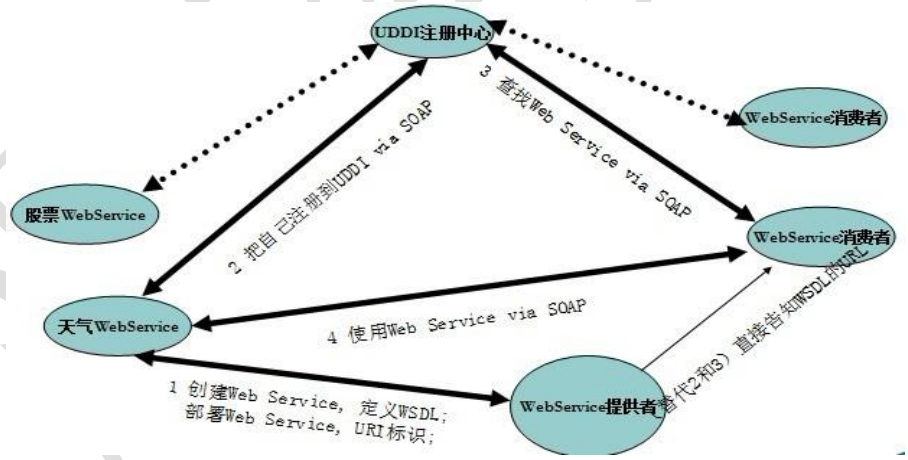
Apache CXF = Celtix + XFire，虽然 XFire 不再维护升级了，但 CXF 继承了 Celtix 和 XFire 两大开源项目的精华，是一款完全支持 Java 的 WS 框架，而且与 Spring 无缝集成，功能很强大，Apache CXF 已经是一个正式的 Apache 顶级项目。

JDK6.0+

JDK6+以上的版本对 WebService 的开发，被认为是最简单的开发方式，而且不需要额外打包，客户端开发也非常简单。

13.3　WebService 工作原理

解析：属于低命中率的面试题，一般做后台开发的职位才会问到你，了不了解 WebService 的实现原理，从网上找了一个图，大家看一下很容易就了解了。



参考答案：WebService 的工作原理应该分为以下几个步骤：

1、服务发布方创建、发布 WebService 服务，并且定义 WSDL 接口；

2、服务发布方将 WSDL 注册到 UDDI 注册中心；

3、WebService 消费者从 UDDI 注册中心找到自己所需要的服务接口，通过 SOAP 协议进行请求服务。

4、对于普通的 B2B 企业之间的 WebService 服务调用，就不需要 UDDI 注册这一步了，服务发布方会直接把 WSDL 提供给消费方。

13.4　可以谈一下什么是 SOA 吗

分析：上面在什么是 WebService 中，把它与"J2EE"概念作了一个类比，有点对 WebService 夸大，毕竟 WebService 只是 SOA 的一种实现。而 SOA 和"J2EE"还是可以类比一下的，SOA 面向服务的体系结构(Service-Oriented ArchITecture，SOA，也叫面向服务架构)。

参考答案：SOA 面向服务的体系结构(Service-Oriented ArchITecture，SOA，也叫面向服务架构)是指为了解决在 Internet 环境下业务集成的需要，通过连接能完成特定任务的独立功能实体实现的一种软件系统架构。

SOA 是一个组件模型，它将应用程序的不同功能单元(称为服务)通过这些服务之间定义良好的接口和契约联系起来。接口是采用中立的方式进行定义的，它应该独立于实现服务的硬件平台、操作系统和编程语言。使得构建在各种这样的系统中的服务可以以一种统一和通用的方式进行交互。

第 14 章　综合部分

14.1　常用哪些设计模式及使用场景

解析：曾经的面试中出现过的问题，原问是“开发中常用到哪些设计模式，结合场景说一下，回答完后紧结着问六个设计原则是什么（这个不单独介绍了，会在附件文档中有一专门介绍 24 种设计模式和六大设计原则的文档）,分别对应哪些设计模式”，这是一个让人容易崩溃的问题，如果你碰到这样的面试，回答的还不错的话，一定要比你期望的薪水再高一点 (和 Spring 面试相似，要回答很多东西才能解释清楚)，所以要在理解的基础上去回答。

参考答案：所谓设计模式，是被多数人验证过的、可重用性强、可靠性强，容易被人理解的经验的总结。话说 Java 常用的设计模式有 23 种（也有说 24 种）之多，而我们普通的工作中最经常用到的有：单例模式、工厂模式、代理模式、策略模式等。

单例模式：一般有常用的有“懒汉式”和“饥饿式”两种写法（写法在笔试部分会提到），但无论哪种想法都不要忽略一点，线程安全问题（单例模式也很容易线程不安全，而且不容易重易错误）；使用场景：通常用于系统启动时一些配置文件的初始化管理，或者实例化一个工厂。

工厂模式：工厂模式一般分为静态工厂和动态工厂或者分为简单工厂和复杂工厂甚至是抽象工厂（详见附属文档），也是我们开发中最常用的设计模式之一，使用场景：创建一系列相互依赖的结构相似的对象（比如 Hibernate 中的 Sesson 就是工厂类生产出来的）。

代理模式：代理模式为静态代理，动态代理；而动代理又分为 JDK 动态代理和 CGLIB 动态代理。除 cglib 实现的动态代理外，都需要基于接口实现。使用场景：业务逻辑的接口不希望直接暴露给客户端调用，采用委托的方式来隐藏真正的实现，可以使用代理模式。

策略模式：使用场景，多个类只区别在表现行为不同，在运行时动态选择具体

要执行的行为，隐藏具体策略(算法)的实现细节，彼此完全独立等。

补充：会发一个通俗易懂的 24 种设计模式介绍，有兴趣的可以自己多研究一下。

14.2　XML 常用的解析方式

解析：这是一个早几年非常热的面试题，现在属于中等命中率，面试被问到的不多，XML 这种文本虽然存储数据有很多优点，但在数据传输时由于文本较大，网络传输时会损失很多性能，对响应较高的系统都是采用 JSON 格式来封装数据了。当然这并不能说明 XML 不重要了，上面介绍 WebService 就说过，WS 现在技术现在被使用很广泛，而 XML 作为 SOAP 协议的一部分，肯定还会经常用到，所以面试还是要好好准备。

参考答案：XML 的解析方式通常有 JOM,DOM4J,SAX，这几中方式也各有优缺点。

JOM:解析器读入整个文档，以树的结构加载到内存中，然后代码就可以使用 DOM 接口来操作这个树结构。

优点：1、允许应用程序对数据和结构做出更改。

2、访问是双向的，可以在任何时候在树中上下导航，获取和操作任意部分的数据。

缺点：将整个文档调入内存（包括无用的节点），浪费时间和空间，如果文件足够大时，超过 10M 加载速度会很慢，甚至会抛出异常。

DOM4J:在文件加载上是和 JOM 一样，也是属于一致性加载到内存中，所以也会继承 JOM 的优缺点，除此外 DOM4J 还有自己独特的地方。

优点：1、使用了很多 JavaApi 的集合类，便于对象的操作。

2、支持 Xpath，性能很好，Hibernate 对 XML 的解析默认的方式就是 dom4j，毕竟在我们自己的开发中，一般很少出现超过 10M 的单个 xml 文件。

缺点：有时候优点过了就是缺点，API 过多也使用开发的复杂性增加。

SAX:SAX 是一个用于处理 XML 事件驱动的“推”模型，它对大文件的处理性能很高，支持对节点分块加载。

优点：是一种解析速度快并且占用内存少的 xml 解析器,它需要哪些数据再加载和解析哪些内容。

缺点：是它不会记录标签的关系,而要让你的应用程序自己处理,这样就增加了你程序的负担。

14.3　Apache、Tomcat、JBoss、WebLogic 的区别

解析：这个问题面试的时候一般不会这样问，如果真被这样问了，只能说面试官一定是一个老人精了。一般的面试会问工作或者学习中使用过哪些服务器，这些服务器有什么区别？之类的问题，这里放到一块只是做一个比较，读者可以对比了解，把自己用到过的掌握就好了。

参考答案：

Apache：Apache HTTP Server（简称 Apache）是 Apache 软件基金会的一个开放源码的网页服务器，通常称为静态服务器，可以在大多数计算机操作系统中运行，由于其多平台和安全性被广泛使用，是最流行的 Web 服务器端软件之一全球应用最广泛的 http 服务器，免费，也是全球使用最多的服务器。

Tomcat：是 Apache 软件基金会（Apache Software Foundation）的 Jakarta 项目中的一个核心项目，应用也算非常广泛的 web 服务器，支持部分 j2ee 最新标准，免费，默认支持 150 并发，一般普通系统部署情况下可以达到 300 的并发左右。

JBoss：开源的应用服务器，发展最为迅速的服务器，商业性能较好，对 J2EE 支持良好，免费（文档收费）。

Weblogic：应该说算是业界第一的 app server，Java 编写的，全部支持 j2ee 最新标准，对于开发者，有免费使用一年的许可证，用起来比较舒服，出资 BEA 公司，现在应该是归宿 Oracle 公司，并且收费规则也有所改变，但对于 Java 开发者来说是一款非常优秀的服务器，有非常友好强大的控制台管理界面。

14.4　动态代理的原理及实现

解析：在前面的面试题中已经不止一次提到动态代理了，可见它还是面试中的热门问题，动态代理的原理如果光靠文字来说明的话是很难做到的，尤其动态原理的实现肯定要结合接口来说，建议求职者在面试前把动态代理的几种实现写一些 DEOM 对比理解一下。

参考答案：Java 开发中，动态代理模式通常有两种实现方式，一种是 JDK 提供的实现（称 JDK 动态代理），另外一种是引入第三方 Jar 字节码技术的方式实现的称为 CGLIB 动态代理。

JDK 动态代理：必须基于接口实现，它是通过对目标类的实现接口来生成代理类（解决静态代理每个接口都生成代理类所带来的维护成本高，代码量大等问题），jdk 的 java.lang.reflect 包中的提供了两个主要类：Proxy 和 InvocationHandler。InvocationHandler 是一个接口，通过实现该接口来实现需要代理的逻辑，并通过反射机制调用目标类的代码，动态将代理逻辑和业务逻辑整合。

Proxy 利用 InvocationHandler 动态创建一个符合某一接口的实例，生成目标类的代理对象。

CGLIB 动态代理：对于没有基于接口开发的类提供的一种代理方式，CGLIB 是在 asm 包上进行的封装，可以动态的对没有接口的业务生成子类，覆盖被代理类（目标类）中所有的业务方法，所以 CGLIB 实现的目标类不能有 final 关键字修饰。

实现过程，声明 Enhancer 类，通过回调来生成代理的子类对象；实现 MethodInterceptor 接口，重写 intercept 方法完成代理逻辑与业务逻辑的整合。另外 Spring 默认使用的是 JDK 动态代理，如果在 Spring 中使用 cglib 需要打开配置，<aop:aspectj-autoproxy proxy-target-class="true"/>。

14.5　Struts 与 Spring 的区别

解析：这是当年笔者面试中被问到的一个题，而且当时面试官将 Struts 的首字母发音[x]，当时我反问了三遍才知道问的是什么问题，而且这两个框架在一起比较不知道面试官想从面试中来了解什么？虽然大家在面试中也可能碰到过很多这样的面试官，但作为求职者还是要耐心的回答那些面试的问题。上面经历了 struts 和 spring 的分析，这个题已经很容易回答了，只要简单的组织一下语言就可以了。

参考答案：Struts 是一个典型 MVC 模式的 Web 框架，能够独立完成 Web 项目的开发。Struts 本身集成了核心控制器 ActionServlet 和 View 层（JSP），但对于 Model 层通常依赖第三方框架，或者直接编码实现的（比如自己写 dao 层 jdbc）。

Spring 是一个轻量级的容器，它的初衷就是致力于减化 J2EE 复杂的开发流程，使用各种框架更好更容易的集成。它的核心是 IOC，可以很好来完成 struts 中 Model 层的业务需求。[在此基础之上，Spring 提供了 AOP（Aspect-Oriented Programming 面向层面的编程）的实现，很方便的对声明事务的操作管理(可以把上面 12.2 中 spring 的结构简单介绍几个)]。而且 Spring 本身还实现了一个 Web 框架 SpringMVC，它拥有 Struts 结构上的所有优点。

结合：它们都是非常优秀的开源框架，而且都拥有完善的解决方案和开源社区，具体使用可以根据项目需求和项目人员结构等条件综合考虑。

14.6　Mybatis 与 Hibernate 的区别

参考答案：Hibernate 是比较流行的全自动 O/R mapping 框架，它出身于 sf.net Mybatis 是一种优秀的半自动 O/R mapping 框架，属于 apache 的一个子项目。两者是非常优秀的 ORM 框架，通过 session 开启事务（类似的 sesson 生命周期），都支持 JDBC 和 JTA 事务，也都有第三方的缓存支持等；它们的区别可以通过下面几点比较一下。

开发相关对比

Hibernate 因为封装程度更深，对程序员的要求也越高，尤其是关联关系、抓取策略、延迟加载、性能优化等方便，都要求程序员有很高的技术积累；换句话说培训成本高。但 Hibernate 因为都是基于配置映射，真正的把程序员从 SQL 中解脱出来，可移植性强，不同方言的支持更变数据库容易，开发效率高，开发周期短。

Mybatis 属于半自动化 ORM 框架，入门快，培训成本低，容易上手，比较容易组建团队。但开发人员还是要自己写 SQL 语句（这点其实无所谓的，中国的码农有不会写 SQL 语句的吗），而且灵活性不如 Hibernate，开发周期较长。

缓存策略对比

Hibernat 采用的是两级缓存策略，一级缓存 Session 级别，一般是与线程绑定；二级缓存是 SessionFactory，通常是引用第三方缓存插件，Encache 缓存是 Hibernate 默认的 CacheProvider，二级缓存是与进程绑定的。

Mybatis 缓存比 Hibernate 更细致，也很容易配置实现，默认情况下 Mybatis 是

没有开启缓存的,除了局部的 session 缓存,要开启二级缓存,你需要在你的 SQL 映射文件中添加一行: <cache/>。

项目维护调优对比

Hiberante 虽然开发周期快，但后期调优（配置的关联关系，不易优化）、异常定位（高度的封装，不易看到 SQL 语句）、功能扩展（容易破坏封装的结构，而且关系是在配置中维持的，不易维护）上都会比 Mybatis 上弱些。

Mybatis 上扩展，调优虽然优秀，但是因为维护太多的 SQL 配置，一旦表结构的变动，就会增加一堆工作量。

14.7　Struts1 与 Struts2 的区别

解析：这个问题应该很容易回答了，上面我们已经把 struts1,2 的优缺点都有谈过了，这里只是把上面的东西整合一下。

参考答案：Struts1 是继 JspMode2 后第一个流行最广的 MVC 模式的 Web 框架，它是 Apache 基金会 Jakarta 项目组的一个优秀 Open Source 项目，但虽然技术的发展也逐渐暴露出许多缺陷，比如：与底层容器耦合太高，不易单独测试；actionForm 组件与 PO,VO 的定义完全相同，造成冗余，单实例处理请求不易高并发的设计等。

Struts2 是站在 Struts1 的基础上，加上 WebWork 的核心，继承了 Struts1 中所有优点；而且低耦合、低侵入的设计使得测试变得很容易；另外强大的拦截机制、方便属性读取传送的 ONGL、模块化配置、多视图支持、全局结果与声明式异常配置、可扩充插件机制、对其它开源框架整合的友好支持等都使得 Struts2 成为更加完善优秀 Web 应用框架。

14.8　对 NoSQL 数据库有什么理解

解析：最近两年对 Nosql 的面试越来越多，尤其在追求互联网提速的今天，NoSQL 的产品很多，但根据笔者所工作过的公司以及对招聘职位的了解来看，一般来说常用的 NoSQL 包括 Redis，MongoDB,Memcached，回答这样的问题一般从为什么要用？它有什么优势？如果能结合自己项目中的实际使用就更好了。

参考答案：NoSQL=Not only SQL 是运用非关系型的数据存储，对之前一直推崇的关系型数据库来说，是一个挑战也是一个新的思维注入。

NoSQL 数据库的特点是：

非关系型存储（包括四大类：Key-Value 健值存储数据库、列存储数据库、文档型数库、图形数据库），从关系型数据库中解脱出来，这让面向对象的开发人员用起来更得心应手。

服务器硬性要求不高，只要是一个 PC 机都可以（当然 NoSQL 产品都是先基于内存再序列号到硬盘的存储，所以内存越大越好），小公司使用起来没压力。

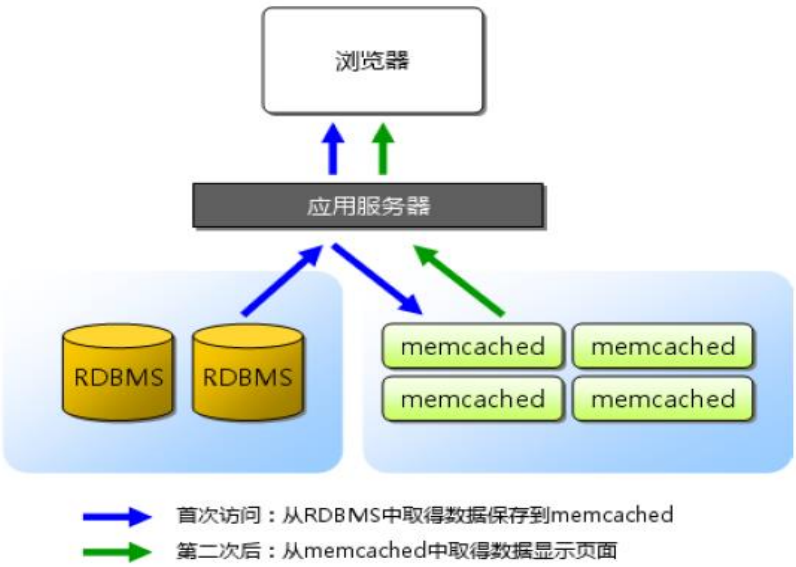
开源易扩展，NoSQL 的产品很多，彼此之间没什么关联关系；而且是基于内存操作，又是以对象存储，对于面向对象的开发人员来说，扩展应该是很容易的。

大数据量高性能，这貌似是是大家最关心的，它的出现也是互联网对性能极尽要求，不再去维护复杂的关联关系，而且是基于内存存储，性能很高。

NoSQL 数据库虽然优点很多，但并不能取代关系型数据库，笔者在实际工作中在对 Redis、MongoDB 使用时，就出现过数据丢失，由于不维护关联关系，查起来很费劲，所以现在最完美的设计是 NoSQL+关联型数据库配合使用，真正的达到读写分离、性能提升。

14.9　Memcached 的工作原理

解析：Memcached 一般被称为查询缓存，它现在也是 NoSQL 代表产品之一。下面是一个 Web 应用中架构图，可以关注一下 Memcached 使用的位置。



参考答案：

Memcached 是可以独立发布的应用缓存服务器，做多台服务集群时，各个缓存服务器之间是没有什么影响的，主导方在于调用端的配置。

Memcached 实现就是一个巨大的、存储了很多<key,value>对的哈希表。通过 key，可以存储或查询任意的数据，客户端可以把数据存储在多台 memcached 上。当查询数据时，首先参考节点列表计算出 key 的哈希值，选中一个节点，客户端将请求发送给选中的节点，然后 memcached 节点通过一个内部的哈希算法，查找真正的 Item。

Memcached 对于容错机制和数据冗余机制几乎不处理，数据冗余或者出错一般都会直接再查一次数据库，因为没有过多额外的处理机制，这使用 Memcaced 性能很高。

14.10　数据库连接池的机制

解析：数据库连接是一种关键的有限的昂贵的资源，这一点在多用户的网页应用程序中体现得尤为突出。对数据库连接的管理能显著影响到整个应用程序的伸缩性和健壮性，影响到程序的性能指标。

数据库连接池正是针对这个问题提出来的。数据库连接池负责分配、管理和释放数据库连接，它允许应用程序重复使用一个现有的数据库连接，而不是再重新建立一个；释放空闲时间超过最大空闲时间的数据库连接来避免因为没有释放数据库连接而引起的数据库连接遗漏。

参考答案：数据库连接池的能够有效的避免开发人员在应用中手动的重复创建和关闭带来的资源浪费。

数据库连接池的基本思想就是为数据库连接建立一个“缓冲池”，预先在缓冲池中放入一定数量的连接，当需要建立数据库连接时，只需要从缓冲池中取出一个了，使用完毕后再放回去。我们可以通过设定连接池最大数来防止系统无尽的与数据库连接。也可以设置最大空闲时间，释放空闲时间超过最大空闲时间的数据库连接来避免因为没有释放数据库连接而引起的数据库连接遗漏。更为重要的是我们可以通过连接池的管理机制监视数据库连接使用数量，使用情况，为系统开发，测试以及性能调整提供依据。

14.11　Maven 的工作原理

解析：这是一个低概率的面试题，但随着越来越多的公司用到 Maven 来管理项目，被问的频率也越来越高。笔者经历的过的公司都有用 Maven 在管理项目，少数用 Ant，求职者应该在两者之间至少了解一种。

参考答案：Maven 作为 Apache 的一个开源项目，它最早的意图只是为了给 apache 组织的几个项目提供统一的开发、测试、打包和部署，能让开发者在多个项目中方便的切换。现在已经成为了很多公司的主要的项目管理框架。

Maven 的基本原理，采用远程仓库和本地仓库以及一个类似 build.xml 的 pom.xml，将 pom.xml 中定义的 jar 文件从远程仓库下载到本地仓库，各个应用使用同一个本地仓库的 jar，同一个版本的 jar 只需下载一次，而且避免每个应用都去拷贝 jar。同时它采用了现在流行的插件体系架构，只保留最小的核心，其余功能都通过插件的形式提供，所以 maven 下载很小（1.1M），在执行 maven 任务时，才会自动下载需要的插件。对于局域网络也可以建立公司内部的私服系统来代替不可能随时上网访问的中央仓库。

14.12　谈一下对 web.xml 的理解

解析：这应该是一个低命中率的题，笔者工作以来参加过很多次面试，也就被问到过一次，把这个题目收录进来，主要是借以说一下 Web 项目的程序入口，一般的 Web 维护程序如何维护，顺便说一下 web.xml 中的常用配置项。

参考答案：对于 Java 程序员来说，程序的一般入口是 Main 方法（对于 App 应用），Applet，然后就是 Servlet。对于开发人员来说，拿到一个 web 项目，在没有任何参考文档的前提下，应该从 web.xml 入手来读程序代码。
对于 web.xml 来说，一般包含 context-param：上下文参数，常用来配置包含的其它 xml 文件，比如 Spring 的配置等。listener：监听器。filter：过滤器，拦截器。servlet:web 应用入口中，还有一些常用欢迎页，异常跳转等等。系统启动后，会首先加载 web.xml 配置，然后初始化 listener>filter>servlet（一般顺序）。如果 context-pararm 中引入了 job.xml 相关的配置，也要注意 job.xml 也是程序入口，通常用来声明一个定时器。

14.13　Encache,Memcached 的区别

解析：这个问题在 NoSQL 没有今天这么火热的时候，经常会放在一起问，笔者认为应该是这两个缓存在之前的应用开发中使用较多的缘故。现在依然属于中高命中率面试题。

参考答案：
Encache 缓存是纯 java 开发的一种插件式缓存，一般与我们的应用集成发布，对于分布式的实现是通过 RMI 调用来实现的。支持容灾机制、支持硬盘序列化（前提对象必须继承了序列化接口）。优点：效率高，功能强大；缺点：客户端单一。
Memcached 缓存是由 C 语言开发的，一般独立部署在服务器，支持 c,java,php 等多种客户端，比 Encache 有更多的用户端。memcached 服务器端是使用不相互通信的协议，分布式实现是由客户端配置来实现的。支持硬盘序列和容灾机制，因为要经过网络通信，性能不如 Encache。优点：支持客户端多，部署灵活；缺点：功能不够完善（序列化到硬盘需要第三方应用实现）。

14.14　能谈一下 linux 常用的命令吗

解析：这是一个常见面试题，也是比较务实的一个题；现在各个公司一般都是开发环境是 Win，而运行环境是 Linux，掌握一些常用的命令是很有必要的（有兴趣的朋友可以装一下 VM，研究一下 jdk,oracle,weblogic 安装部署，以及应用服务的发布，这可能会是你超越别人的一个捷径），下面列出一些与工作中常用到的，如果想全面学习还是要参考 liunx 相关的书。

参考答案：

文件&文件目录操作->

cd /file #进入 file 目录；cd.. #退出一级目录；cd/ #退到根目录。

pwd #查看当前的位置路径；

ls #查看文件目录中的文件；ls -ll #显示详细信息；ls -a#显示隐藏信息。

rm -f file#删除当前目录下的 file 文件；rm -fr dir #删除 dir 目录（包含子文件）。

mkdir file #新建一个文件夹；mkdir file1 file2 #创建两个文件目录。

mv source target #将文件 source 重命名 target。

cp sou tar #将文件 sou 复制 tar;cp /root/sou . #将 root 下的 sou 复制到当前目录。

cat file #显示文件内容;tac file 反向查看一个文件内容；head -i file #查看一个文件的前 i 行；tail -i file #查看一个文件的后 i 行。

tail -f file #查看文件的动态日志；tail -f nohup.out #weblogic 下 out 的动态日志。

vi file #编辑文件。

find /-name file #从根开始搜索文件和目录。

find / -user username #从根开始搜索属于 username 的文件和目录

进程查看操作->

ps #显示当前系统进程；ps -ef #显示系统所有进程。

ps -ef|grep java#显示系统 java 进程；ps -ef|grep app #显示 app 应用进程名。

kill -9 i #杀死编号为 i 的进程;killall -9 i #对包有名字为 i 的进程杀死。

top #显示系统的活跃情况，按 CPU 资源百分比来分。

free#显示系统内存及 swap 使用情况。

注意：top,free 命令在系统性能监控时，很有作用。

14.15 如何设计高并发下的秒杀功能

分析：这是一个没有标准答案的面试题，主要是用来考察解决问题的能力，不一定有结果，但最好能回答点思路。目前互网联创业公司太多，张嘴就来高并发、分布式，而如果你的面试官是阿里、京东出来的，类似的问题会经常被问到，这里给一些参考答案，以供抛砖引玉。

参考答案：不过多么强大的服务器，在秒级内达到数十万数百万的并发估计都可能挂掉。秒杀对于淘宝的用户来说，同一时间数万并发应该是常见的，但这个压力并不一定会到应用服务器上；比如前面的负载均衡服务器肯定先进行大的分流，分流到应用服务器时的在线人员可能依旧很大，这时可以通过登录队列来控制排队数，比如只有 30 个物品参与秒杀，那我就从队列中取出三十人进行秒杀的业务逻辑，大部分用户阻塞在队列外就 OK 了。

14.16　谈一下高并发框架的设计

分析：以前认为这样的问题只有在参加高级开发工程师、系统加构师时才会有被问到的可能；笔者今年去参加面试，几乎随便一个互联网公司都会或多或少的提到此类问题。有很多面试官仅仅只是问一句有没有参与过，无论你回答有或者无，都没有下文了，可见很多面试官心理也没有底。而我一个朋友被问的很深，入职后却是要负责一个N多年的老项目维护项目，他很是无语（求职者一定要问清职位）。对于高并发的框架设计，以现在目前的主流技术来说，还是很容易设计的，笔者刚刚参与过一个查询服务相关的框架设计，当时在线人数也有 200 万左右，这里与大家分享一下。

参考答案：对于高并发框架的设计通常要分三部分来考虑：应用程序、数据库环境、部署环境。

应用程序：首先在设计应用设计上就要考虑到应用的分布式部署（部署多少台应用服务器），开发选择的框架一定要支持分布式的部署才行；对于访问量大的页面可以采用静态页（结合一些前端脚本技术使用）；尽可能使用查询缓存，做到业务部分的读写分离；对于长业务逻辑处理的部分要尽可能分离出来等等。

数据库环境：对于大并发量的框架设计，数据库一定要采用集群的方式（我们公司之前的环境就是使用单独数据库服务，虽然使用了大量的缓存服务，还是会因为连接数问题导致服务挂掉）。

部署环境：最好将应用部署多台服务器上（每台服务器也可以装多个应用服务器），必须部署负责均衡的服务器，必须有数据库集群的服务器，最好有独立的缓存服务器。

第四部分　数据库技术

说明，本节谈到的数据库面试，是在应聘 Java 面试中穿插的数据库面试；根据笔者的面试经验，Java 开发人员面数据库技术问题多是一些适用性问题，所以对于面试 DBA 或者其它数据职位的求职者来说，这里的东西就不够看了。而且从目前的行情来看，工作经验越久，面试越单一（应聘高级 Java 职位可能都很少提到数据库问题，最多是一些框架相关的问题）；而对于初级开发人员来说，反而苦逼，从前端 UI 到后端数据库，有可能连 phostshop, Dreamweaver 都有人问，所以工作经验越少的朋友面试越应该做好准备。

第 15 章　JDBC 操作部分

15.1　Statement 与 PreparedStatement 的区别

解析：这个是属于高命中率面试题，无论是初中高级职位都有可能被问到的，此题可以很好的考察求职者的基础能力。

参考答案：

1、PreparedStatement 支持预编译，对于批量处理可以大大提高效率；Statement 不支持预编译。

2、PreparedStatement 可避免如类似单引号的编码麻烦，Statement 不可以。

3、Statement 每次执行 sql 语句都要数据库执行 SQL 编译，间接增加了数据服务器的压力；preparedStatement 可以减少与服务器的交互，减轻数据服务器的压力。

4、PreparedStatement 可防止 Sql 助于，更加安全，而 Statement 不行。

5、如果是单条语句的执行 PreparedStatement 开销要大些，反而 Statement 更方便性能要好。

6、在 sql 语句出错时 PreparedStatement 不易检查，而 Statement 则更便于查错。

15.2　JDBC 连接数据库的步骤

解析：这个主要是针对刚毕业的求职者，或者工作经验比较短的求职者，小小一道题，卡死Ｎ多人（也有很多工作五六年的朋友），不管是出于哪种原因你没有答好，要么是让你丢掉 Offer，要么是失去谈薪资的优势，所以小问题也要好好关注，而且小题的过滤性很强的。

参考答案：

1、加载 JDBC 驱动程序（Class.forName(.....)）。

2、获得 Connection con； DriverManager.getConnection(url , uname , password)。

3、创建 statement stmt；con.prepareStatement(sql)。

4、执行 Sql；stmt.executeQuery(sql)。

5、处理 ResultSet 结果集。

6、关闭 JDBC 对象。

第 16 章　MYSQL 部分

16.1　Mysql 的常用连接方式

解析：对于用惯了 Oracle 的开发程序员来说，虽然转型 Mysql 问题不大，但刚开始总会对一些关键性名词和书写方式感觉别扭，笔者有一次被问到“全连接”愣了好半天没反应过来，下面介绍一些 MySql 的连接方式。

参考答案:

内连接(自然连接)：显示关联的两个表中相匹配的数据。

左外连接(左边的表不加限制)：Left Join 左表是驱动表。

右外连接(右边的表不加限制)：Right Join 右表是驱动表。

全外连接(左右两表都不加限制)：Left Join union Right Join。

说明：这里的全外连接又称全连接，由于 mysql 不支持 full join 之类的写法，实际上它查询的是左连接和右连接的并集。

16.2　Mysql 乱码如何解决

解析：这个面试问题是比较务实的一个问题，也是在开发过程中经常遇到的问题，下面给一些我之前遇到乱码的解决方式，求职者可以了解一下，在实际工作中也会有用。

参考答案：先分清是否是数据库乱码，如果是其它则用其它方式处理。

1.在安装数据库的过程中将默认的拉丁文-->GBK、UTF-8、GB2312。

2.在创建数据库时设置选择 GBK 或者 gb2312 或者 UTF-8。

3.Mysql 安装目录下的 my.ini 文件，将 "default-character-set=xxxxx" 中的 xxxxx 改成 GBK 或者 gb2312 或者 utf-8。

4.Mysql 安装目录下的\data\databasename(数据库名)\db.opt 文件打开 default-character-set=gbk default-collation=gbk_chinese_ci; 如果上面不是 gbk 和 gbk_chinese_ci 则改成支持中文的 GBK 或者 gb2312 或者 utf-8。

5.进入 Mysql 的 dos 命令下:进入某数据库后 show full columns from tablename ; 查看数据类型，如果不是支持中文的类型则执行 alter table tablename convert to character set gbk 。

6.在创建数据库时(用命令创建时)create database databasename CHARACTER SET gbk;

说明，上面是我几年前总结的，那个时候还比较倾向于用 GBK，其实现实的开发工作中，一般都会用 UTF-8，也建议要习惯 UTF-8，毕竟 UTF-8 支持语言文字边界要多一些，而且更国际化一些。

16.3　Mysql 如何获取系统时间

解析：mysql 里面的时间函数特别的多，如果你不确认面试官想问你是获取日期，还是获取时间，还是获取日期+时间，可以反问一下。不过一般没有必要，因为你只要回答一下开发中常用到的几个函数就行了，面试官如果想知道详细，会追问你的。

参考答案：mysql 中常用取系统时间的函数有 now();sysdate();current_time。

第 17 章　ORACLE 部分

17.1　oracle 中 rownum 与 rowid 的理解

分析：属于面试中的高命中率题目，这两个关键字在 oracle 开发中还是会经常用得到。

参考答案：rownum 和 rowid 都是伪列，但两者的根本是不同的。rownum 是随查询结果动态改变的，而 rowid 是一个物理地址，硬盘物理数据不变就不会变。

rownum 是根据 sql 查询出的结果给每行分配一个逻辑编号，所以你的 sql 不同也就会导致最终 rownum 不同，一般常用在数据库分页中。

rowid 是物理结构上的，在每条记录 insert 到数据库中时，都会有一个唯一的物理记录（不会变），开发中常到的是过滤重复数据。

17.2　union 和 union all 有什么不同

参考答案：union 和 union all 都是对两个表的查询结果就归并到一起，所以首先要求两个查询语句的结构要完全一致（查询字段名称、顺序要一样）。

不同点在于 union 是对两个查询的结果进行按默认规则排序并删除重复记录，而 union all 只是简单的对两个查询结果进行合并返回。所以从效率上来说 union all 要快的多（如果说两个查询语句中确认没有重复的数据，建议优先使用 union all）。

17.3　truncate 和 delete 的区别

参考答案：truncate 和 delete 都可以对表中的数据进行删除处理。

1、delete 属于 DML，一般是在应用程序中操作，删除数据时一般会加上 where 条件进行过滤；truncate 一般是在 Oracle 客户端上执行，不需要过滤条件。

2、delete 删除数据是一行一行的删除，同时会记录删除日志，如果出现误删除的数据，可以从删除日志中恢复；truncate 是按照页进行删除，不会记录操作

日志，删除会数据将不能再被找回。truncate 的效率将高于 delete。
3、delete 如果绑定了触发器，删除时将会触发触发器；而 truncate 不会。
4、delete 删除数据时将会先锁定各行，truncate 将是暴力删除。
5、如果有 identity 产生的自增 id 列，delete from 后仍然从上次的数开始增加，即种子不变，而 truncate 后，种子会恢复初始。
6、用 truncate 删除运行中的相关表数据时，一定要备份。

17.4　谈谈你对 SQL 优化的看法

解析：这是一个高命中率的题，一般应用框架搭建好后，开发人员都是按照固定的流程开发，只要不是特别差的开发人员，Java 应用程序这块的优化空间一般是不大的。如果提高系统的性能，多是从 SQL 方面进行优化，好的 SQL 与差的 SQL 那可是天地差别，所以要了解一些 SQL 的基本常识对于面试和工作来说都很有帮助。

参考答案：SQL 优化主要分为两方面，一方面为 DB 服务端优化，一方面为应用中的 SQL 优化。

DB 服务端优化：

1、合理的创建索引，尤其是查询系统，数据量大的时候必须有索引。
2、如果查询的 SQL 特别复杂的时候，而且应用端又不能优的时候，建议封装成存储过程调用。
3、进可能使用数据库提供的原始函数。

应用端的优化：

1、尽量避免复杂的 SQL 语句，如果嵌套层次很深的语句，最好分成几个子语句进行操作。
2、合理排列过滤条件，根据数据库的解析的不同，尽量把过滤性大的条件放到先解析的位置。mysql 是从左到右解析（从上到下），Oracle 是从右向左解析（从下到上）；合理的排列过滤条件。
3、尽量少用 select * 的查询，精确到字段；这样不仅能优化，在 ORM 映射上将会给你带来很大的好处，尤其是字段增添的维护上。
4、避免滥用 in,not in ，根据业务逻辑合理的选择 exist,not exist 来替代。
5、尽量减少与数据服务器的交互，减少连接时间，不要在循环中频繁的和数据库服务器交互。
6、尽量避免在建立了索引的数据列上进行计算，not，<>，!=，IS NULL，IS NOT NULL 和使用函数的操作，索引字段不要有 null 值。
7、尽可能的用 in 查询来代替 or 查询。
8、尽量少用 like“%name%”这样的查询将不走索引。
9、尽可能的不要到系统出现问题了再想到按照上面的方式去优化。

17.5　存储过程与函数的区别

解析：此问题在以前的面试过程中命中率很高，最近发现问的较少了，这两

者的区别还是很明显的，求职者只要答出几个关键点可以了。

参考答案：

1、相同点，两者都在数据服务端执行，可以减少应用服务器压力；都可以传入参数，都可以有返回值。

2、函数只能有一个返回值，而存储过程可以返回多个，或者不返回。

3、函数一般用来完成某个特定的计算，过程一般用来封装某锻炼比较复杂的 SQL 执行或者业务处理。

4、函数一般在 SQL 语句中调用，调用灵活，不可以单独执行；过程一般通过特定的方式特用，可以在 PLSQL 中单独调用。

5、函数的声明用 function，过程的声明用 procedure。

第五部分　笔试与上机

在 Java 面试指南第一个版本中，本来不打算插入笔试题，由于笔试题的范围广，知识点分列的散，命中率太低，而且十分占篇章，将考虑在以后的版本中把笔试题单独拿出来，对于工作经验<=三年的求职者来说，参与笔试的可能性较大，只有笔试成绩达到公司的期望值才会安排面试（某金融互联网公司的笔试成绩只有大于 30 分才有面试机会，而好多有五年工作经验的人都挂掉了）。在本版本中先列出一些当前碰到比较多的笔试题，对于详细的还是放到以后的版本中。

第 18 章　程序阅读

18.1　读代码写执行结果

第 01 题：

```
publicstaticvoid main(String[] args) {
        System.out.println("the value:" + FinallyTest.getValue());
    }
    publicstaticint getValue() {
        int i = 100;
        try {
            return i;
        } finally {
            ++i;
            System.out.println("the finally i:" + i);
        }
    }
```

解析：考察 reutrn finally 执行顺序（结合上面的面试分析）。

```
参考答案：the finally i:101
the value:100
```

拓展：如果将 return i 放到最后执行，结果如何？

第 02 题：

```
class Sup {
    publicinti = 10;
    publicvoid A() {
        System.out.println("in Sup.A()...");
    }
    publicvoid B(){
        System.out.println("Sup中B()方法被执行...");
        this.A() ;
    }
}
publicclass Sub extends Sup {
    publicinti = 30;
    publicvoid A() {
        System.out.println("in Sub.A()...");
    }
    publicstaticvoid main(String[] args) {
        Sup F = new Sub();
        Sub S = new Sub();
        F.B() ;
        S.B() ;
        System.out.print("F.i + S.i = " + (F.i + S.i)+"\n");
        F.A() ;
    }
}
```

解析：考察面向对象三大特征之一继承。这里难点在于 S.B()方法的调用，调用的子类继承父类方法，而 B 方法中的 this 的指向，它是属于子对象的。

```
参考答案：Sup中B()方法被执行...
in Sub.A()...
Sup中B()方法被执行...
in Sub.A()...
F.i + S.i = 40
in Sub.A()...
```

第 03 题：

```
classSup {
    static {
        System.out.print("1");
    }
    public Sup() {
        System.out.print("2");
    }
```

```
}
publicclass Sub extendsSup {
    static {
        System.out.print("3");
    }
    public Sub() {
        System.out.print("4");
    }
    publicstaticvoid main(String[] args) {
        SupF = newSup();
        SupS = new Sub();
    }
}
```

解析：考察静态块的加载，静态块在累装载时（实例化）加载，而且静态块只加载一次，弄清这点，结果就很容易写出来了。

参考答案：13224

拓展：如果有 static{}块下面再加几个{}语句块，执行顺序又当如何呢？

第 04 题：

```
        Integer i1 = 100 ;
        Integer i2 = 100 ;
        System.out.println(i1==i2);
        Integer i3 = 200 ;
        Integer i4 = 200 ;
        System.out.println(i3==i4);
```

解析：考察 int 型如何自动转成 Integer 对象的，下面是 Integer 转换的源码。

```
publicstatic Integer valueOf(int i) {
    finalint offset = 128;
    if (i >= -128 && i <= 127) { // must cache
    return IntegerCache.cache[i + offset];
    }
returnnew Integer(i);
 }
```

－128 至 127 是通过缓存的方式转换的，大于 127 是直接 new 生成的，明白这点，这题的答案也就很容易了。

参考答案：true false

第 05 题：

```
    String s = "Hello";
    char[] ch = { 'w', 'o', 'r' , 'l', 'd' };
    publicstaticvoid main(String args[]) {
        Example ex = new Example();
        ex.change(ex.s, ex.ch);
        System.out.print(ex.s + "-");
```

```
        System.out.print(ex.ch);
    }

    publicvoid change(String s, char ch[]) {
        s = "world";
        ch[0] = 'W';
    }
```

解析：这是在网上最近经常看到的一个题，主要用来考虑值传递和引用传递的区别，这里要区别 change 中 s 只是一个局部变量，而 ch 是一个数组的引用，所以局部变量不会影响方法外的结构，但数组、集合等引用地址的改变，会造成对象中内容的变化。

参考答案：Hello-World

第 06 题

```
publicstaticint get(int i) {
        int result = 0;
        switch (i) {
        case 1:
            result = result + i;
        case 2:
            result = result + i * 2;
        case 3:
            result = result + i * 3;
        default:
            result = result + i * 10;
        }
        return result;
    }

    publicstaticvoid main(String[] s) {
        System.out.println(SwitchTest.get(2));
    }
```

解析：这个题型属于较老的一种面试题，主要考察的是 switch-case 中 break 的应用，如果在 switch-case 中不合理的使用 break 语句，上面的代码逻辑将使程序变得毫无意义。

参考答案：30

18.2 String 经典系列

第 01 题：写出程序执行结果？（某金融互联网公司 2015 年笔试题）

```
publicstaticvoid main(String[] args) {
        String s="" ;
        StringBuffer sb1 = new StringBuffer("s");
```

```
        StringBuffer sb2 = new StringBuffer("s");
        StringBuffer sb3 = new StringBuffer(sb2);
        StringBuffer sb4 = sb3;
        if(sb1.equals(sb2)){s+="1" ;}
        if(sb2.equals(sb3)){s+="2" ;}
        if(sb3.equals(sb4)){s+="3" ;}
        String s1 = new String("s");
        String s2 = new String("s");
        String s3 = new String(s2);
        String s4 = s3;
        if(s1.equals(s2)){s+="4" ;}
        if(s2.equals(s3)){s+="5" ;}
        if(s3.equals(s4)){s+="6" ;}
        System.out.println(s);
    }
```

解析：考察 equals 的重写问题？这里 StringBuffer 是没有重写 equals 方法的。

```
参考答案：3546
```

第 02 题

```
String s1 = "ab" ;
        String s2 = "a" ;
        String s3 = "b" ;
        System.out.println(s1=="a"+"b");
        System.out.println(s1=="a"+s3);
        System.out.println(s1==s2+s3);
```

解析：这里是考察 String 常量池的概念，上面的难点主要在 s1=="a"+s3 的比较上，由于 s3 在编译时是当作变量处理，所以会重新生成对象，而"a"+"b"是在编译之前就完成了组合并且存放在变量池中的。

```
参考答案：true
false
false
```

拓展：如果在 s2,s3 前加上 final 结果会如何呢？true.

第 03 题

```
publicstaticvoid main(String[] args) {
        String a = "ab";
        final String bb = getChangeB();
        String b = "a" + bb;
        System.out.println(a == b);
    }

    privatestatic String getChangeB() {
        return"b";
    }
```

解析：在第02题中的拓展提问中，String变量被声明成final时会，变量会到常量

池中，上面的结果也是全部变成了true；而此题却说明被final修饰的变量并不一定是放到常量池中的。上面的getChangeB()方法中实际上为返回的变量在堆中开辟了一块空间，而上面的`final String bb = getChangeB();`调用其实就相当于`final String bb = new String("b");`重新生成了对象。

参考答案：false

18.3 分析程序是否异常

第 01 题：阅读两段小程序，指出程序运行结果？

```
a、)
publicstaticvoid main(String[] args) {
    final String str = "word";
    System.out.println("str:" + str);
}
b、)
publicstaticvoid main(String[] args) {
    static String str = "word";
    System.out.println("str:" + str);
}
```

解析：考察局部变量如何声明初始化的问题，这里的 b 中，由于方法中的变量不能被 static 修饰，将编译异常。

参考答案：a、) str:word
b 、)编译异常

第 02 题

```
abstractclass AbstractUser {
    public String name = "word";
    publicabstractvoidget(String name){};
}
publicclass User extends AbstractUser{
    publicvoid get(String name) {
        name = super.name ;
        System.out.println("hello:"+name);
    }
}
```

解析：此题看着像一道程序分析题，其实如果眼光锐力点，上来就会发现这是一个有问题的程序片段，抽象类中的抽象方法是不能定义实例的，即没有{}。

参考答案：编译不通过，抽象方法不能有{}。

第 03 题

```
abstractclass AbstractUser {
    public String name = "word";
```

```
    privateabstractvoidget(String name);
}
publicclass User extends AbstractUser{
    publicvoid get(String name) {
        name = super.name ;
        System.out.println("hello:"+name);
    }
}
```

解析：此题与上面相比，好像没什么问题了，但犯了一个更基本的常识错误，抽象方法一定不能定义 private，抽象方法必须重写才有意义。

参考答案：编译不通过，抽象方法不能定义成私有的。

第 19 章上机编程

19.1　写出一个单例模式

解析：对于工作经验不多的求职者，上机或比试时，此题的命中率都是相当的高，主要考察动手编码能力和是否对单例模式了解（写法很多，这里给出一种参考）。

参考答案：

```
class SingletonTest {
    privatestatic SingletonTest instance;
    private SingletonTest() {
    }
    publicstatic SingletonTest getInstance() {
        if (instance == null) {
            instance = new SingletonTest();
        }
        returninstance;
    }
}
```

说明：这只是最常用的一种写法，在实际开发中单例虽然只加载一次，但在复杂的系统中，也有可能并发出现多实例的问题，所以最好考虑同步问题，比如加锁。

19.2　按字节截取的字符串

题目介绍：编程：编写一个截取字符串的函数，输入为一个字符串和字节数，

输出为按字节截取的字符串。但是要保证汉字不被截半个，如"我ABC"4，应该截为"我AB"，输入"我ABC汉DEF"，6，应该输出为"我ABC"而不是"我ABC+汉的半个"。

解析：这是一个很老的机试题了，但最近碰到两个公司都在考此题，其中一家还是外企，看来笔试也有周期性回归的趋势。

参考答案：

```
publicstaticvoid split(String source, int num){
       int k = 0;
       StringBuffer sbf = new StringBuffer("") ;
       for (int i = 0; i < source.length(); i++) {
          byte[] b = (source.charAt(i) + "").getBytes();
          k = k + b.length;
          if (k > num) {
             break;
          }
          sbf.append(source.charAt(i)) ;
       }
       System.out.println(sbf.toString());
   }
publicstaticvoid main(String[] args) {
       String SOURCE = "我ABC汉DEF" ;
       int NUM = 5  ;
       CutString.split(SOURCE, NUM) ;
}
```

19.3　多线程编程题

题目介绍：设计 4 个线程，其中两个线程每次对 j 增加 1，另外两个线程对 j 每次减少 1，写出程序。

解析：这也是一个很老的机试题了，最近在一家外企笔试题中偶遇到，以下程序使用内部类实现线程，对 j 增减的时候没有考虑顺序问题。

参考答案：

```
publicclass ThreadTest {
   privateintj;
   publicstaticvoid main(String args[]) {
       ThreadTest tt = new ThreadTest();
       Inc inc = tt.new Inc();
       Dec dec = tt.new Dec();
       for (int i = 0; i < 2; i++) {
          Thread t = new Thread(inc);
          t.start();
          t = new Thread(dec);
          t.start();
```

```
        }
    }
    privatesynchronizedvoid inc() {
        j++;
        System.out.println(Thread.currentThread().getName() +
"-inc:" + j);
    }
    privatesynchronizedvoid dec() {
        j--;
        System.out.println(Thread.currentThread().getName() +
"-dec:" + j);
    }
    class Inc implements Runnable {
        publicvoid run() {
            for (int i = 0; i < 100; i++) {inc();}
        }
    }
    class Dec implements Runnable {
        publicvoid run() {
            for (int i = 0; i < 100; i++) {dec();}
        }
    }
}
```

19.4 写一个冒泡算法

解析：如果涉及到算法排序笔试的，冒泡算法考察的是是多的，但求职者可以多准备几种排序算法，以备笔试之需（这里只是一个样例，在实际笔试或工作中，算法之类的东西一定要封装起来，可以重用）。

参考答案：

```
publicstaticvoid main(String[] args) {
        int input[] = { 9,41,2,15,42,16,85,64,12,48,9 };
        for (int i = 0; i < input.length - 1; i++) {
            for (int j = 0; j < input.length - i - 1; j++) {
                if (input[j] < input[j + 1]) {
                    int temp = input[j];
                    input[j] = input[j + 1];
                    input[j + 1] = temp;
                }
            }
        }
        for (int a = 0; a < input.length; a++) {
```

```
            System.out.print(input[a] + " ");
        }
    }
```

19.5　向 int 数组随机插入 100 个数

题目介绍：产生一个 int 数组，长度为 100，并向其中随机插入 1-100，并且不能重复。

解析：最近看到网上问到此题的比较多，而且回答的人不多，此题应该是某互联网公司的笔试题，这里给一个参考。代码写的不算严谨，但思路还是清晰的，代码很容易读，不多作解释了。

```
参考答案：
publicstaticvoid main(String[] args) {
        int[] obj = newint[100]; //声明一个100长度的数组
        List<Integer> list = new ArrayList<Integer>();
        for (int i = 1; i <= 100; i++) {
            list.add(i); //我把100个数存到集合中
        }
        for (int i = 0; i < 100; i++) {
            int index = new Random().nextInt(list.size()); //随机取
一个小于当前集合长度的下标
            obj[i] = list.get(index);//取出集合中的数据放到数组里
            list.remove(index);//集合中删除此下标
        }
        for (int i = 0; i < 100; i++) { // 打印输出
            System.out.print(obj[i] + " ");
            if ((i + 1) % 10 == 0) {
                System.out.println();
            }
        }
s}
```

第六部分　离职流程

20.1　离职信

解析：离开一家公司，除了直接和主管，人力资源口头说明以外，一定要提供一份正式的离职信。无论是以何种原因离开，开心的不开心的，笔者都建议在

离职信中都要以感谢、感激的心态；毕竟这是我们曾经服务过的公司，我们从这里或多或少的积累了经验、收获了技能、认识了同事、交到了朋友。也不排除我们的离职是由于不能忍受公司的勾心斗角、任务繁重、受到排挤、管理不善等原因，但既然都准备离开了，就把这一切放在老公司吧，要不然带着负担到新的公司，只是恶性循环而已，怀着一颗感恩的心开始职涯的新的里程；下面是一个通用的模板（特殊情况可以邮件我，我可是专业的，呵呵）。

离职申请

Xxxx 公司领导您好：

入职 xxx 公司已经三年了，感谢公司一直以来的关心和照顾，在这个快乐的大家庭里让我收获良多、成长良多。今因个人原因，特向公司提出请辞，望其批准，在最后这段时间内，我会认真完成交接工作，烦请安排。谢谢！

祝：工作顺利。

xxxxxx

2015.02.17

20.2　离职流程

解析：离和流程提出以后，接下来的流程应该是安抚谈话、离职审批、交接流程、工资结算、离职证明。这块属于个人交流，仅供参考。

一、安抚谈话

如果不是公司主动赶走的员工，当你提交离职信后，你的主管或者其它领导会一个安抚谈话的过程；一般手段是：1、问你离职的原因？感觉待遇低给你加薪；工作压力太大，给你调职（个人认为这算是比较有诚意的谈话，但为什么处理问题非要等到员工提离职呢？这是领导应该反思的）2、谈人生谈理想，画大饼，承诺加薪，承诺年终（个人认为比较扯淡，或许因为你走了对项目影响较大，慢慢的把工作重心给你转移，把你架空然后你可以滚蛋了）。笔者并非喜欢从消极的方面考虑问题，所以也建议，考虑清楚了再离职，不要以离职作为理由和公司谈条件，这样的员工公司和同事都会鄙视你。

二、交接流程

离开公司只是为了更好的追求，你到新的公司报到和别人来接替你的工作角色是一样的，所以交接一定要认真负责的完成；你也希望到新的职位上，工作有一个完善的交接过程。这就是所谓的与人方便，与己方便，不要到处挖坑（最后往往坑的是自己）。

除了按照工作的交接流程走之外，可以自己列一下交接清单，或者写一下交接文档。这些东西除了有助于新员工的入职，还可以是对自己有一个很好的总结，工作了这么多年，不要只在简历上留下一个简单的项目名称。

三、工资结算

大家有始有终，最后的薪资容易出问题，所以薪资这块一定要清清楚楚，做到不是我的一分不取（第一家公司多发我 3000 多就退回了，后来取公积金时需要公司开证明，大家都很愉快），是自己的一定要争取，不要为了拿到离职证明，

放弃很多应该有的福利（比如报销费用，年终）。

四、离职证明

不管大小公司，无论新的公司是否要求提供离职证明，最好在老的公司一定要拿到离职证明，这也是你一段工作经历的一个凭证。

第七部分　附录：简历模板

个人简历

个人资料		
姓名：xxx	性别：x	
出生日期：xxxx/xx/xx	籍贯：xx	
毕业院校：xxxxxxxx	专业：计算机科学技术	
工作地点：xxx	邮箱	
联系方式：xxx-xxxx-xxxx		
申请职位：　JAVA 开发高级工程师/项目经理		

学校教育

20xx/x-20xx/x　计算机科学技术　xxxxxxxxxx　本科

工作经验

2009/7-2012/3　　东 xxxx　　任职　　软件开发工程师

开发项目：XXXxxxxx 服务系统

xxxxxxx 服务端

XXXxxxxxxxx 系统

2012/4-今 xxxx　　任职　　软件开发设计工程师

设计开发:　GISxxxxxxxx

国网 xxxxx 应用系统

xxxxxxx 公共查询服务

xxxxx 管理应用后台

专业技能

★语言及框架

. 熟练 J2SE 的 CoreJava 部分，熟练使用 API 常用库，了解 JVM 原理和垃圾回收机制。

. 熟练 J2EE 的基础及核心知识 JDBC、JSP、Servlet 和 JavaBean 以及框架的快速搭建。

．熟练 Struts(1/2)、MyBatis、Hibernate、Spring 技术,特别清楚各个框架的流程以及框架的选择和搭建。

．对基于 MVC 开发模式的主流开发框架如 SS(2)H（I）及之间的整合具有深该的理解和应用。

．熟练运用 Web 页面高级标签，如 JSP 标签，Struts 标签等，熟悉基本的 JavaScript、CSS。

．熟练多种常用开源框架工具 SVN、MyBatis、Cache、Ant、SoapUI、Log4J、Lunce、OracleText、FindDugs 等。

xxxxxxxxxxxxxxxxxxx 扩展 xxxxxxxxxxxxxxxxxxxx

★数据库

．熟练使用 Mysql、ORACLE 等关系数据库，能够在熟练在第三客户端工具上进行过程、视图、索引、触发器等方面开发，也能够进行性能优化。

．能够根据业务需求通过 PowerDesinger 进行数据库的设计，以及应用级别的管理维护等。

xxxxxxxxxxxxxxxxxxx 扩展 xxxxxxxxxxxxxxxxxxxx

★其它方面

．熟练 SOA 常用的框架开发，尤其对 WebService 相关的 CXF、XFIRE、Axis2、JDK 各种实现引擎能够很好选择合理的框架并且搭建，对 EJB 等其它形式的远程接口调用也较为熟悉。

．熟练 PowerDesinger、Enterprise Architect 系统设计工具的使用及管理流程。

．了解常用的 Linux 命令，能够 linux 下熟练运用 Tomcat Web 服务器，Weblogic 服务器等其它管理工作。

xxxxxxxxxxxxxxxxxxx 扩展 xxxxxxxxxxxxxxxxxxxx

项目经验

项目名称：xxx 企业 OA 系统

所用技术：Struts2+Mybatis+Mysql 数据库

开发环境：Myeclipse+Tomcat7+WinXP(Win2003)+Mysql5.5

项目描述：OA 追求高效智能化的办公方式，满足公司智能办公的需要，方便公司人员、资源的管理与协调、操作方便、安全有效的办公辅助管理系统，主要包括角色管理、权限管理、业务流程、车间管理、财务管理、人事管理、工作日志、休闲区域、月度报表、库存管理等十大模块，30 多个小模块构成。

责任描述:

项目负责人，主要负责项目前期需求调研、编写需求文档、系统设计、数据库设计、工作分配与管理、版本控制与管理、项目管理等。

项目名称：浙江电网 xxx 应用平台

所用技术：Java 语言+CXF 框架+Oracle 数据库

开发环境：Eclipse+WebLogic+WinXP(Linux)+Oracle10.3

项目描述：xxxxxxxxx。

责任描述:

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

扩展使用.....

扩展使用.....

自我评价