Web 后台开发工程师岗位面试题

# Java

## 基础

### 语言特性

#### 易

1. Java中，一个char类型的变量能否存储一个中文汉字（JVM采用哪种字符编码）

答：可以。Java采用Unicode编码，2个字节表示一个字符。

1. Override与Overload有什么区别

答：

1. overload是重载，重载是一种参数多态机制，即代码通过参数的类型或个数不同而实现的多态机制。 是一种静态的绑定机制（在编译时已经知道具体执行的是哪个代码段）；
2. override是覆盖。覆盖是一种动态绑定的多态机制。即在父类和子类中同名元素（如成员函数）有不同 的实现代码。执行的是哪个代码是根据运行时实际情况而定的。
3. ==与equals()有什么区别

答：

1. 如果运算符两边的变量都是基本数据类型，那么只要这两个变量的值相等，判断结果返回true；
2. 如果运算符两边的变量都是引用数据类型，那么只有当这两个变量都指向同一对象（两个变量在内存中的地址相同）判断结果返回true。
3. String类是否可以继承

答：不可以。String是final类型。

1. 请简述深复制与浅复制的区别

答：

1. 浅复制时，类内属性为基本类型则直接复制相应值，如果是对象则复制其实例引用（复制对象和新对象中的引用指向同一对象）
2. 深复制则会完全新建完全新的对象属性。
3. 如果希望java类能进行复制操作，则该类必须实现cloneable接口。
4. Object类有哪些方法

答：

1. Object()：默认构造方法；
2. clone()：创建并返回此对象的一个副本；
3. equals(Object obj)：指示某个其他对象是否与此对象“相等”；
4. finalize()：当垃圾回收器确定不存在对该对象的更多引用时，由对象的垃圾回收器调用此方法；
5. getClass()：返回一个对象的运行时类；
6. hashCode()：返回该对象的哈希码值；
7. notify()：唤醒在此对象监视器上等待的单个线程；
8. notifyAll()：唤醒在此对象监视器上等待的所有线程；
9. toString()：返回该对象的字符串表示；
10. wait()：导致当前的线程等待，直到其他线程调用此对象的 notify() 方法或 notifyAll() 方法；
11. wait(long timeout)：导致当前的线程等待，直到其他线程调用此对象的 notify() 方法或 notifyAll() 方法，或者超过指定的时间量；
12. wait(long timeout， int nanos)：导致当前的线程等待，直到其他线程调用此对象的 notify() 方法或 notifyAll() 方法，或者其他某个线程中断当前线程，或者已超过某个实际时间量；

#### 中

1. final， finally， finalize有什么区别

答：

1. final是修饰符，如果一个类被声明为final，意味着它不能再派生出新的子类，不能作为父类被继承。因此一个类不能既被声明为 abstract的，又被声明为final的。将变量或方法声明为final，可以保证它们在使用中不被改变。被声明为final的变量必须在声明时给定初值，而在以后的引用中只能读取，不可修改。被声明为final的方法也同样只能使用，不能重载。
2. finally是在异常处理时提供 finally 块来执行任何清除操作。如果抛出一个异常，那么相匹配的 catch 子句就会执行，然后控制就会进入 finally 块（如果有的话）。
3. finalize是方法名。Java允许使用 finalize() 方法在垃圾收集器将对象从内存中清除出去之前做必要的清理工作。这个方法是由垃圾收集器在确定这个对象没有被引用时对这个对象调用的。它是在 Object 类中定义的，因此所有的类都继承了它。子类覆盖 finalize() 方法以整理系统资源或者执行其他清理工作。finalize() 方法是在垃圾收集器删除对象之前对这个对象调用的。
4. String s = new String("xyz");创建了几个对象

答：1个或2个。

当常量池中有”xyz”这个常量时，则仅创建1个对象，否则创建2个对象。

1. Java 中，switch表达式支持哪些类型的数据，支持String吗

答：byte、short、int、char、枚举。JDK 1.7之后支持String。

#### 难

1. 如果用equals()判断两个对象相等/不相等，那么它们的hashCode()值是否一定相等/不相等，反过来呢

答：

1. 若两个对象equals()相等，则两个对象的hashCode()一定相等;
2. 若两个对象equals()不相等，则两个对象的hashCode()不一定不相等；
3. 若两个对象hashCode()相等，则两个对象的equals()不一定相等；
4. 若两个对象hashCode()不相等，则两个对象的equals()一定不相等；

### 常见集合

#### 易

1. HashMap**与Hashtable有什么区别**

答：

1. HashTable的方法是同步的，HashMap未经同步，所以在多线程场合要手动同步HashMap这个区别就像Vector和ArrayList一样；
2. HashTable不允许null值（key和value都不可以），HashMap允许null值（key和value都可以）；
3. HashTable有一个contains(Object value)，功能和containsValue(Object value)功能一样；
4. HashTable使用Enumeration，HashMap使用Iterator；
5. HashTable中hash数组默认大小是11，增加的方式是old\*2+1。HashMap中hash数组的默认大小是16，而且一定是2的指数；
6. 哈希值的使用不同，HashTable直接使用对象的hashCode，而HashMap重新计算hash值；
7. ArrayList**和LinkedList有什么区**别

答：

a）ArrayList

ArrayList其实是包装了一个数组 Object[]，当实例化一个ArrayList时，一个数组也被实例化，当向ArrayList中添加对象是，数组的大小也相应的改变。这样就带来以下有缺点：

快速随机访问：你可以随即访问每个元素而不用考虑性能问题，通过调用get(i)方法来访问下标为i的数组元素。

向其中添加对象速度慢：当你创建数组是并不能确定其容量，所以当改变这个数组时就必须在内存中做很多事情。

操作其中对象的速度慢：当你要想数组中任意两个元素中间添加对象时，数组需要移动所有后面的对象。

b）LinkedList

LinkedList是通过节点直接彼此连接来实现的。每一个节点都包含前一个节点的引用，后一个节点的引用和节点存储的值。当一个新节点插入时，只需要修改其中保持先后关系的节点的引用即可，当删除记录时也一样。这样就带来以下有缺点：

操作其中对象的速度快：只需要改变连接，新的节点可以在内存中的任何地方

不能随机访问：虽然存在get()方法，但是这个方法是通过遍历接点来定位的所以速度慢。

1. List， Set， Map 分别继承自什么接口

答：

1. List和Set继承自Collection接口；
2. Map没有从接口继承；

#### 中

1. HashMap的Key值可以是一个可变对象吗，为什么

答：不可以。可变对象会使key的哈希值产生变化，最终导致数据丢失。

1. HashMap的初始大小是多少，它的扩容机制是怎样的

答：16。当HashMap的容量达到当前大小\*负载因子（默认为0.75）时，其容量增大1倍。

#### 难

1. HashMap中产生哈希碰撞会带来哪些影响

答：如果多个hashCode()的值落到同一个桶内的 时候，这些值是存储到一个链表中的。最坏的情况下，所有的key都映射到同一个桶中，这样hashmap就退化成了一个链表——查找时间从O(1)到 O(n)。

### 异常处理

#### 易

1. Error与Exception有什么区别和联系

答：

联系

a）两者有共同的父类Throwable

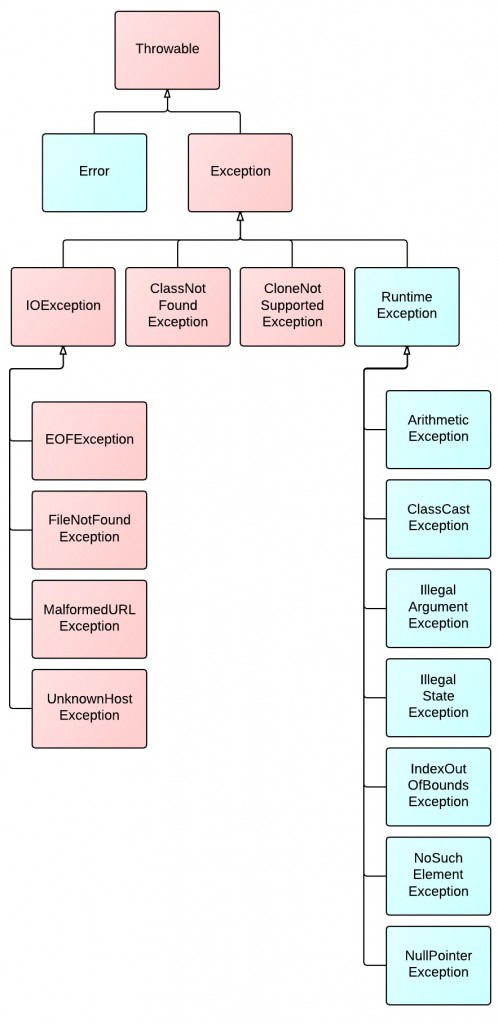
区别

1. Error 表示恢复不是不可能但很困难的情况下的一种严重问题。比如说内存溢出。不可能指望程序能处理这样的情况；
2. Exception 表示一种设计或实现问题。也就是说，它表示如果程序运行正常，从不会发生的情况。
3. 请说出5个以上常见的Exception

答：NullPointerException、ClassNotFoundException、ArrayIndexOutOfBoundsException、FileNotFoundException、ArithmeticException、StringIndexOutOfBoundsException

#### 中

1. CheckedException 和 UncheckedException有什么区别



1. CheckedException: 这类异常都是Exception的子类 。异常的向上抛出机制进行处理，如果子类可能产生A异常，那么在父类中也必须throws A异常。可能导致的问题：代码效率低，耦合度过高。C#中就没有使用这种异常机制。
2. UncheckedException: 这类异常都是RuntimeException的子类，虽然RuntimeException同样也是Exception的子类，但是它们是特殊的，它们不能通过client code来试图解决，所以称为Unchecked exception 。

#### 难

1. 在try-catch-finally中，如果try-finally或者catch-finally中都有return，那么两个return的执行顺序是怎样的？

答：在try-catch-finally中，如果try-finally或者catch-finally中都有return，则两个return语句都执行并且最终返回到调用者那里的是finally中return的值；而如果finally中没有return，则理所当然的返回的是try或者catch中return的值，但是finally中的代码是必须要执行的，而且是在return前执行，除非碰到exit()。

### GC

#### 易

1. 什么是GC？为什么要有GC？GC常见的两种方式？
2. Garbage Collection简称为GC，是垃圾回收的意思；
3. 内存处理器是编程人员容易出现问题的地方，忘记或者错误的内存回收会导致程序或系统的不稳定甚至崩溃。Java语言提供的GC功能可以自动的检测对象是否超过作用域，从而达到自动回收内存的目的。
4. GC包括Minor GC和Full GC

#### 中

1. minor GC与Full GC有什么区别

a）新生代 GC（Minor GC）：指发生在新生代的垃圾收集动作，因为 Java 对象大多都具

备朝生夕灭的特性，所以 Minor GC 非常频繁，一般回收速度也比较快。

b）老年代 GC（Major GC / Full GC）：指发生在老年代的 GC，出现了 Major GC，经常

会伴随至少一次的 Minor GC（但非绝对的，在 ParallelScavenge 收集器的收集策略里

就有直接进行 Major GC 的策略选择过程） 。MajorGC 的速度一般会比 Minor GC 慢 10

倍以上。

1. Java中是否会发生内存泄漏，有哪些原因可能导致内存泄漏

答：

会。在Java中，内存泄漏就是存在一些被分配的对象，这些对象有下面两个特点：

a）首先，这些对象是可达的，即在有向图中，存在通路可以与其相连；

b）其次，这些对象是无用的，即程序以后不会再使用这些对象；

如果对象满足这两个条件，这些对象就可以判定为Java中的内存泄漏，这些对象不会被GC所回收，然而它却占用内存。

#### 难

1. JVM有哪些垃圾标记算法
2. 引用计数（Reference Counting）算法。某对象有一个引用，即增加一个计数，删除一个引用则减少一个计数。垃圾回收时，只用收集计数为0的对象。此算法最致命的是无法处理循环引用的问题。
3. 根搜索（GC Roots Tracing）算法。基本原理是：GCRoot对象作为起始点（根）。如果从根到某个对象是可达的，则该对象称为“可达对象”（存活对象，不可回收对象）。否则就是不可达对象，可以被回收。

GC Roots对象包括：

* 虚拟机栈（栈帧中的本地变量表）中的引用的对象。
* 方法区域中的类静态属性引用的对象。
* 方法区域中常量引用的对象。
* 本地方法栈中JNI（Native方法）的引用的对象。

1. JVM有哪些垃圾收集算法
2. 标记-清除算法。标记—清除算法包括两个阶段：“标记”和“清除”。在标记阶段，确定所有要回收的对象，并做标记。清除阶段紧随标记阶段，将标记阶段确定不可用的对象清除。

标记—清除算法是基础的收集算法，标记和清除阶段的效率不高，而且清除后回产生大量的不连续空间，这样当程序需要分配大内存对象时，可能无法找到足够的连续空间。

1. 复制算法。复制算法是把内存分成大小相等的两块，每次使用其中一块，当垃圾回收的时候，把存活的对象复制到另一块上，然后把这块内存整个清理掉。复制算法实现简单，运行效率高，但是由于每次只能使用其中的一半，造成内存的利用率不高。现在的JVM用复制方法收集新生代，由于新生代中大部分对象（98%）都是朝生夕死的，所以两块内存的比例不是1:1(大概是8:1)。
2. 标记-整理算法。标记—整理算法和标记—清除算法一样，但是标记—整理算法不是把存活对象复制到另一块内存，而是把存活对象往内存的一端移动，然后直接回收边界以外的内存。标记—整理算法提高了内存的利用率，并且它适合在收集对象存活时间较长的老年代。
3. 分代收集算法。分代收集是根据对象的存活时间把内存分为新生代和老年代，根据个代对象的存活特点，每个代采用不同的垃圾回收算法。新生代采用标记—复制算法，老年代采用标记—整理算法。
4. 引用计数法的判定效率很高，为为什么在主流的JVM里都没有选用它来管理内存

答：难以解决对象循环引用的问题。

### 序列化

#### 易

1. 什么是序列化和反序列化，Java中如何实现对象序列化

答：

定义：

1. 把对象转换为字节序列的过程称为对象的序列化；
2. 把字节序列恢复为对象的过程称为对象的反序列化。

用途：

1. 把对象的字节序列永久地保存到硬盘上，通常存放在一个文件中；
2. 在网络上传送对象的字节序列。

实现：

1. 将需要被序列化的类实现Serializable接口；
2. 然后使用一个输出流(如：FileOutputStream)来构造一个ObjectOutputStream（对象流）对象；
3. 接着，使用ObjectOutputStream对象的writeObject(Object obj)方法就可以将参数为obj的对象写出（即保存其状态），要恢复的话则用输入流。

#### 中

1. 如果一个类能够被序列化，那么它的父类和子类是否也能够被序列化

答：

1. 子类：可以被序列化
2. 父类：若父类已实现Serializable接口，则父类相应字段及属性的处理和该类相同；若父类未实现序列化，则该类的父类所有字段属性将不会序列化。

#### 难

1. 一个类的方法和静态的类属性是否可以被序列化。

答：不可以。

1. Java序列化时，serialVersionUID有什么作用

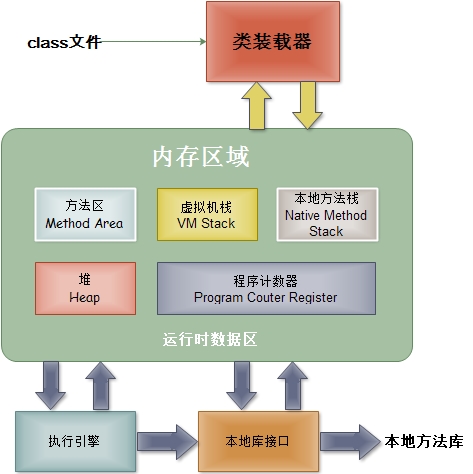
答：Java的序列化机制是通过在运行时判断类的serialVersionUID来验证版本一致性的。在进行反序列化时，JVM会把传来的字节流中的serialVersionUID与本地相应实体（类）的serialVersionUID进行比较，如果相同就认为是一致的，可以进行反序列化，否则就会出现序列化版本不一致的异常。

## 加分项

### JVM

#### 易

1. JVM内存的几个组成部分中，哪些是线程共享的，哪些是线程独占的



1. 程序计数器（Program Counter Register）：线程私有。一块较小的内存空间，是当前线程所执行的字节码的行号指示器；

异常：是JVM中，唯一一个没有规定任何OutOfMemoryError情况的区域；

1. Java虚拟机栈（JVM Stack）：线程私有。每个方法被执行的时候 都会创建一个“栈帧”用于存储局部变量表(包括参数)、操作栈、方法出口等信息。每个方法被调用到执行完的过程，就对应着一个栈帧在虚拟机栈中从入栈到出栈的过程。

异常：StackOverflowError（线程请求的栈深度大于虚拟机所允许的深度）、OutOfMemoryError（虚拟机栈动态扩展时无法申请到足够的内存）

1. 本地方法栈（Native Method Stack）：与虚拟机栈基本类似，区别在于虚拟机栈为虚拟机执行的java方法服务，而本地方法栈则是为Native方法服务。

异常：StackOverflowError、OutOfMemoryError

1. Java 堆（Java Stack）：线程共享，也称为“永久代”、非堆（Non-Heap）。是Java虚拟机所管理的内存中最大的一块。
2. 方法区（Method Area）：线程共享。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。

异常：OutOfMemoryError（方法区无法满足内存分配需求）

1. 运行时常量池（Runtime Constant Pool）：方法区的一部分，用于存放编译时生成的各种字面量和符号引用。

异常：OutOfMemoryError（运行时常量池是方法区的一部分，当常量池无法再申请到内存）

#### 中

1. 什么是新生代和老年代

答：

a）新生代

程序新创建的对象都是从新生代分配内存，新生代由Eden Space和两块相同大小的Survivor Space(通常又称S0和S1或From和To)构成，可通过-Xmn参数来指定新生代的大小，也可以通过-XX:SurvivorRation来调整Eden Space及Survivor Space的大小。

b）老年代

用于存放经过多次新生代GC任然存活的对象，例如缓存对象，新建的对象也有 可能直接进入老年代，主要有两种情况：①.大对象，可通过启动参数设置-XX:PretenureSizeThreshold=1024(单位为字节，默 认为0)来代表超过多大时就不在新生代分配，而是直接在老年代分配。②.大的数组对象，切数组中无引用外部对象。

老年代所占的内存大小为-Xmx对应的值减去-Xmn对应的值。

#### 难

1. 如果一个程序发生OutOfMemoryError，可能会是哪些原因导致的，该如何解决
2. OutOfMemoryError： PermGen space。发生这种问题的原意是程序中使用了大量的jar或class，使java虚拟机装载类的空间不够，与Permanent Generation space有关。解决方法：

* 增加java虚拟机中的XX:PermSize和XX:MaxPermSize参数的大小；
* 清理应用程序中因重复等问题导致的不必要的jar；

1. OutOfMemoryError： Java heap space。发生这种问题的原因是java虚拟机创建的对象太多，在进行垃圾回收之间，虚拟机分配的到堆内存空间已经用满了，与Heap space有关。解决方法：

* 检查程序，看是否有死循环或不必要地重复创建大量对象；
* 增加Java虚拟机中Xms（初始堆大小）和Xmx（最大堆大小）参数的大小；

1. OutOfMemoryError：unable to create new native thread。这种错误在Java线程个数很多的情况下容易发生。解决方法：

* 减少虚拟机内存的分配。

1. 说说java.lang.StackOverflowError出现的场景

答：当前线程栈满，可能原因：函数调用层级过多、递归次数过多等。

解决方案：

1. 检查程序中是否有死递归之类的情况，进行调整。
2. JVM参数设置： -XX:ThreadStackSize，如 -XX:ThreadStackSize=512

### 泛型

#### 易

1. 请简单介绍Java泛型，使用泛型有何优缺点

答：

1. 泛型的定义：泛型是JDK 1.5的一项新特性，它的本质是参数化类型（Parameterized Type）的应用，也就是说所操作的数据类型被指定为一个参数，在用到的时候在指定具体的类型。这种参数类型可以用在类、接口和方法的创建中，分别称为泛型类、泛型接口和泛型方法。
2. 优点：安全简单。在编译的时候检查类型安全，并且所有的强制转换都是自动和隐式的，提高代码的重用率；
3. 缺点：要做显式的强制类型转换，而这种转换是要求开发者对实际参数类型可以预知的情况下进行的。对于强制类型转换错误的情况，编译器可能不提示错误，在运行的时候才出现异常，这是一个安全隐患。

#### 中

1. List<String>和原始类型List之间的区别

答：带参数类型是类型安全的，而且其类型安全是由编译器保证的，但原始类型List却不是类型安全的。你不能把String之外的任何其它类型的Object存入String类型的List中，而你可以把任何类型的对象存入原始List中。使用泛型的带参数类型你不需要进行类型转换，但是对于原始类型，你则需要进行显式的类型转换。

#### 难

1. List<? extends T>和List <? super T>之间有什么区别

答：

a）<? extends T> 表示类型的上界，表示参数化类型的可能是T 或是 T的子类；

b）<? super T> 表示类型下界（Java Core中叫超类型限定），表示参数化类型是此类型的超类型（父类型），直至Object

### 多线程

#### 易

1. Sleep ()、suspend ()和 wait ()之间有什么区别
2. sleep ()使当前线程在指定的时间处于“非运行”（Not Runnable）状态。线程一直持有对象的监视器。比如一个线程当前在一个同步块或同步方法中，其它线程不能进入该块或方法中。如果另一线程调用了 interrupt ()方法，它将唤醒那个“睡眠的”线程。
3. suspend ()是过时的方法，使用 suspend ()导致线程进入停滞状态，该线程会一直持有对象的监视器，suspend ()容易引起死锁问题。
4. wait ()使当前线程出于“不可运行”状态，和 sleep ()不同的是wait 是 object 的方法而不是 thread。调用 object.wait ()时，线程先要获取这个对象的对象锁，当前线程必须在锁对象保持同步，把当前线程添加到等待队列中，随后另一线程可以同步同一个对象锁来调用 object.notify ()，这样将唤醒原来等待中的线程，然后释放该锁。
5. Java中有哪几种实现多线程的方式

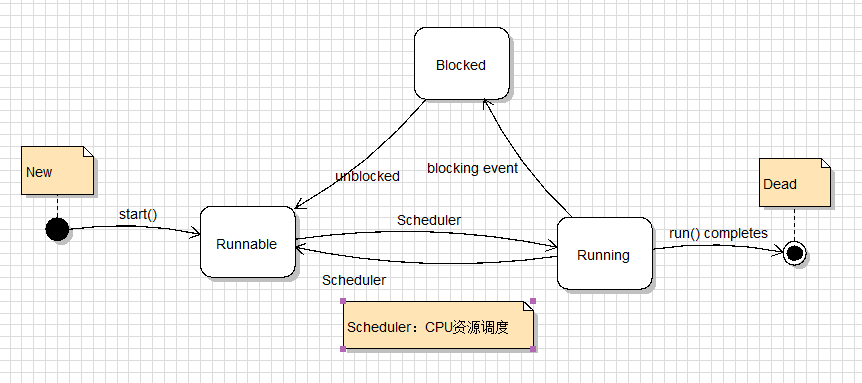
答：

1. 继承Thread类，重写run函数；
2. 实现Runnable接口，重写run函数；
3. Thread.start ()与 Thread.run ()有什么区别？

答：Thread.start ()方法(native)启动线程，使之进入就绪状态，当 cpu 分配时间该线程时，由 JVM 调度执行 run ()方法。

#### 中

1. 线程有哪几种状态，他们之间是如何调度的



1. 在一个对象上两个线程可以调用两个不同的同步实例方法吗？

答：不能，因为一个对象已经同步了实例方法，线程获取了对象的对象锁。所以只有执行完该方法释放对象锁后才能执行其它同步方法。

#### 难

1. 什么是 ThreadLocal 类，怎么使用它

答：ThreadLocal 是一个线程级别的局部变量，并非“本地线程”。ThreadLocal 为每个使用该变量的线程提供了一个独立的变量副本，每个线程修改副本时不影响其它线程对象的副本。其关键点如下：

1. 一个线程局部变量(ThreadLocal variables)为每个线程方便地提供了一个单独的变量。
2. ThreadLocal 实例通常作为静态的私有的(private static)字段出现在一个类中，这个类用来关联一个线程。
3. 当多个线程访问 ThreadLocal 实例时，每个线程维护 ThreadLocal 提供的独立的变量副本。

### 反射

#### 易

1. 什么是反射

答：

1. Java反射机制是在运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法；
2. 对于任意一个对象，都能够调用它的任意一个方法和属性；
3. 这种动态获取的信息以及动态调用对象的方法的功能称为Java语言的反射机制；
4. 反射的概念是由Smith在1982年首次提出的，主要是指程序可以访问、检测和修改它本身状态或行为的一种能力。

#### 中

1. 使用Java的反射机制，需要哪几个步骤

答：

1. 获得你想操作类的Class对象
2. 通过第一步获得的Class对象去取得操作类的方法或是属性名
3. 操作第二步取得的方法或是属性

#### 难

1. 有哪几种获取操作类的Class对象的方式

答：

1. 调用Class的静态方法forName，如上例；
2. 使用类的.class语法，如：Class<?> cls = String.class；
3. 调用对象的getClass方法，如：String str = "abc"；Class<?> cls = str .getClass()；
4. Class.forName是用来做什么的

答：初始化参数指定的类，并且返回此类对应的Class 对象。例如，以下代码片段返回 java.lang.Thread 类对应的Class对象。

Class t =Class.forName("java.lang.Thread")

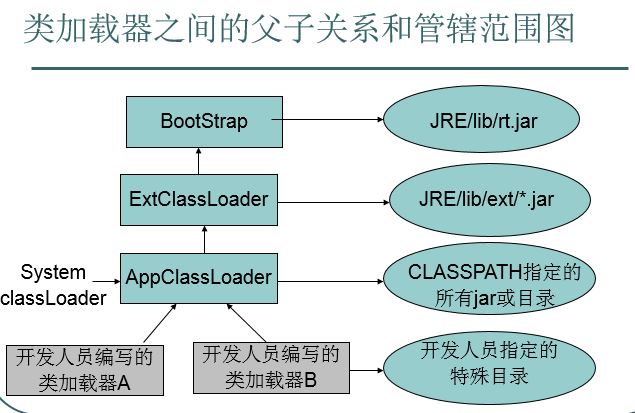
调用forName("X") 将导致名为 X 的类被初始化。

### 类加载机制

#### 易

1. Java有哪几种类加载器

答：



1. 启动类加载器（Bootstrap ClassLoader），C++实现，是虚拟机自身的一部分。负责将%JAVA\_HOME%\lib目录中，或被-Xbootclasspath参数所指定的路径，并且是虚拟机识别的类加载到虚拟机内存中。
2. 其它类加载器，Java实现，独立于虚拟机外部并都继承自抽象类java.lang.ClassLoader：
   * 扩展类加载器（Extension ClassLoader），负责加载%JAVA\_HOME%\lib\ext目录中的，或被java.ext.dirs系统变量所制定的路径中的所有类库，开发者可直接使用该类加载器；
   * 应用程序类加载器（Application ClassLoader），也叫系统类加载器：负责加载用户类路径（ClassPath）上所指定的类库，开发者可直接使用该类加载器。

#### 中

1. 请简述Java的类加载过程

答：Java程序运行的场所是内存，当在命令行下执行：

java HelloWorld

命令的时候，JVM会将HelloWorld.class加载到内存中，并形成一个Class的对象HelloWorld.class。

其中的过程就是类加载过程：

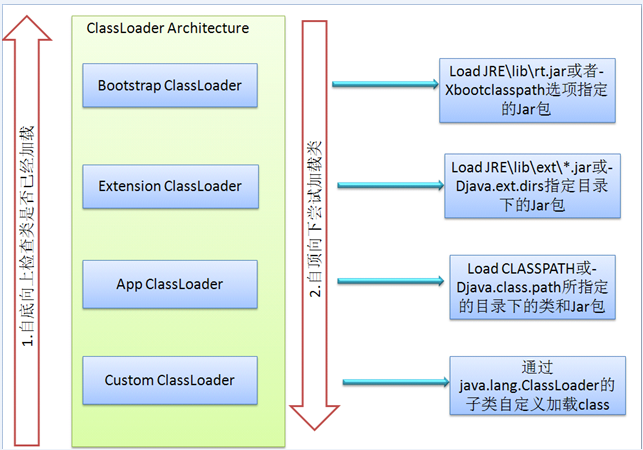
1. 寻找jre目录，寻找jvm.dll，并初始化JVM；
2. 产生一个Bootstrap Loader（启动类加载器）；
3. Bootstrap Loader自动加载Extended Loader（标准扩展类加载器），并将其父Loader设为Bootstrap Loader。
4. Bootstrap Loader自动加载AppClass Loader（系统类加载器），并将其父Loader设为Extended Loader。
5. 最后由AppClass Loader加载HelloWorld类。

#### 难

1. 能不能自己写一个java.lang.String类

答：可以，但是得需要自己写类加载器去加载。

1. 简述类加载器的双亲委派模型（Parents Delegation Model）



当JVM加载一个类的时候，下层的加载器会将将任务委托给上一层类加载器，上一层加载检查它的命名空间中是否已经加载这个类，如果已经加载，直接使用这个类。如果没有加载，继续往上委托直到顶部。检查完了之后，按照相反的顺序进行加载，如果Bootstrap加载器找不到这个类，则往下委托，直到找到类文件。对于某个特定的类加载器来说，一个Java类只能被载入一次，也就是说在Java虚拟机中，类的完整标识是（classLoader，package，className）。一个类可以被不同的类加载器加载。

# 数据库

## 基础

### SQL

#### 易

1. 数据库中，内联接和外联接有什么区别

答：内链接是保证两个表中所有的行都要满足联接条件，而外联接则不然。

在外联接中，某些不满足条件的列也会显示出现，也就是说，只限制其中一个表的行，而不限制另一个表的行。分左联接，右联接，全联接。

1. SQL中，where与having有什么区别

答：where 子句的作用是在对查询结果进行分组前，将不符合where条件的行去掉，即在分组之前过滤数据，条件中不能包含聚组函数，使用where条件显示特定的行。

having 子句的作用是筛选满足条件的组，即在分组之后过滤数据，条件中经常包含聚组函数，使用having 条件显示特定的组，也可以使用多个分组标准进行分组。

1. delete和turncate table有何区别

答：

1. truncate和不带where子句的delete、以及drop都会删除表内的数据
2. drop、truncate都是DDL语句(数据定义语言),执行后会自动提交
3. truncate 和 delete 只删除数据不删除表的结构(定义)
4. drop 语句将删除表的结构被依赖的约束(constrain)、触发器(trigger)、索引(index)；依赖于该表的存储过程/函数将保留,但是变为 invalid 状态
5. delete 语句是数据库操作语言(dml)，这个操作会放到 rollback segement 中，事务提交之后才生效；如果有相应的 trigger，执行的时候将被触发。
6. truncate、drop 是数据库定义语言(ddl)，操作立即生效，原数据不放到 rollback segment 中，不能回滚，操作不触发 trigger

#### 中

1. VARCHAR 和 CHAR 两种类型有什么区别

CHAR存储定长数据很方便，CHAR字段上的索引效率级高，比如定义char(10)，那么不论你存储的数据是否达到了10个字节，都要占去10个字节的空间。

VARCHAR。存储变长数据，但存储效率没有CHAR高。如果一个字段可能的值是不固定长度的，我们只知道它不可能超过10个字符，把它定义为 VARCHAR(10)是最合算的。VARCHAR类型的实际长度是它的值的实际长度+1。为什么“+1”呢？这一个字节用于保存实际使用了多大的长度。

#### 难

1. MySQL/Oracle中如何查询一个表的索引

show index from XXX

### 设计

#### 易

1. 什么是数据库三范式

答：

第一范式：1NF是对属性的原子性约束，要求属性具有原子性，不可再分解；

第二范式：2NF是对记录的惟一性约束，要求记录有惟一标识，即实体的惟一性；

第三范式：3NF是对字段冗余性的约束，即任何字段不能由其他字段派生出来，它要求字段没有冗余。

#### 中

1. 数据库主键有哪几种设计方式

答：

GUID、自增、int、自定义

1. 什么时候会采用反范式的数据库设计方法
   * 1. 冗余提升效率
     2. 数据仓库

#### 难

1. 各种主键设计方式有何优缺点

答：

1. 采用GUID作为主键，这种的优势就是全球唯一，无论是用数据库还是用代码，产生都很方便，维护很方便。但性能是最低的，使用的时候要注意；
2. 采用数据库自动增长的方式。这种方式性能好，产生也很方便。但是维护很麻烦。一般的话，这种主键用于一些特殊的表，比如说日志表，其不需要维护；
3. 采用int型作为主键，这种方式是性能好，产生也很方便，维护也方便，但是主键的不具有什么实在意义；
4. 采用自定义编号，性能一般，产生方便，维护一般。 为什么说维护一般呢？比如一个erp系统里面的商品编号，如果我在系统单据里面使用过，客户要求修改其商品编号，修改点就非常多。但是它又有一个很好的优点，可以让编号非常有意义，比如单据中的单据编号，可以按单据标识+日期来制定。

### 事务

#### 易

1. 数据库事务有哪些特性

答：原子性（atomicity）。一个事务是一个不可分割的工作单位，事务中包括的诸操作要么都做，要么都不做。

一致性（consistency）。事务必须是使数据库从一个一致性状态变到另一个一致性状态。一致性与原子性是密切相关的。

隔离性（isolation）。一个事务的执行不能被其他事务干扰。即一个事务内部的操作及使用的数据对并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰。

持久性（durability）。持续性也称永久性（permanence），指一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。接下来的其他操作或故障不应该对其有任何影响。

1. for update 和 for update nowait 的区别

for update nowait和 for update 都会对所查询到得结果集进行加锁，所不同的是，如果另外一个线程正在修改结果集中的数据，for update nowait 不会进行资源等待，只要发现结果集中有些数据被加锁，立刻返回（ORACLE）。

MYSQL只有innodb支持for update，但不支持for update nowait。

#### 中

1. 开启事务、提交与回滚事务的sql语句

BEGIN TRANSACTION

COMMIT

ROLLBACK

1. 什么是事务？什么是锁？

答：事务就是被绑定在一起作为一个逻辑工作单元的SQL语句分组，如果任何一个语句操作失败那么整个操作就被失败，以后操作就会回滚到操作前状态，或者是上有个节点。为了确保要么执行，要么不执行，就可以使用事务。要将有组语句作为事务考虑，就需要通过ACID测试，即原子性，一致性，隔离性和持久性。

锁：在所以的DBMS中，锁是实现事务的关键，锁可以保证事务的完整性和并发性。与现实生活中锁一样，它可以使某些数据的拥有者，在某段时间内不能使用某些数据或数据结构。当然锁还分级别的。

#### 难

1. 维护数据库的完整性和一致性，你喜欢用触发器还是自写业务逻辑？为什么？

答：尽可能使用约束，如check，主键，外键，非空字段等来约束，这样做效率最高，也最方便。其次是使用触发器，这种方法可以保证，无论什么业务系统访问数据库都可以保证数据的完整新和一致性。最后考虑的是自写业务逻辑，但这样做麻烦，编程复杂，效率低下。

## 加分项

### 存储过程/触发器/游标

#### 易

1. 什么是存储过程

答：存储过程是一个预编译的SQL语句，优点是允许模块化的设计，就是说只需要创建一次，以后在该程序中就可以调用多次。如果某次操作需要执行多次SQL，使用存储过程比单纯SQL执行要快。可以降低网络的通信量。封装和隐藏了数据逻辑

1. 存储过程有哪些优缺点

优点：

1. 允许模块化设计，可重复调用；
2. 作为预编译命令，有更快的执行速度；
3. 减少网络流量；
4. 可作为安全机制，屏蔽数据库细节；

缺点：

a）无法支持数据库平台无关性

#### 中

1. 游标的使用有哪几个关键步骤

答：

1. 声明游标；
2. 打开游标；
3. 提取游标；
4. 关闭游标；

#### 难

1. 如果两个不同的存储过程在一个方中被同是调用，当其中任一个发生异常时，要求同时回滚两个存储过程的操作，怎么样处理

可以创建另个一个存储过程，在这个存储中，创建一个自组事务，在这个自组事务中分别去调用那两个存储过程。语法如下：

create procedure CallTwoProc

as

begin transaction

execute 第一个存储过程

execute 第二个存储过程

commit transaction

go

### 性能调优

#### 易

1. 索引有什么作用，有什么优缺点

答：作用：索引就一种特殊的查询表，数据库的搜索引擎可以利用它加速对数据的检索。它很类似与现实生活中书的目录，不需要查询整本书内容就可以找到想要的数据。索引可以是唯一的，创建索引允许指定单个列或者是多个列。缺点是它减慢了数据录入的速度，同时也增加了数据库的尺寸大小。

#### 中

1. 你知道/用过哪些数据库性能优化方法
2. 慢日志记录
3. EXPLAIN PLAN——SQL语言优化命令
4. 调整数据结构的设计
5. 调整服务器配置
6. 调整数据库配置

#### 难

1. MySQL有哪几种存储引擎，有什么区别

MyISAM是MySQL的默认存储引擎。MyISAM不支持事务、也不支持外键，但其访问速度快，对事务完整性没有要求。

InnoDB存储引擎提供了具有提交、回滚和崩溃恢复能力的事务安全。但是比起MyISAM存储引擎，InnoDB写的处理效率差一些并且会占用更多的磁盘空间以保留数据和索引。

MEMORY存储引擎使用存在内存中的内容来创建表。每个MEMORY表只实际对应一个磁盘文件。MEMORY类型的表访问非常得快，因为它的数据是放在内存中的，并且默认使用HASH索引。但是一旦服务关闭，表中的数据就会丢失掉。

FEDERATED引擎是专门针对远程数据库的实现。一般情况下在本地数据库中建表会在数据库目录中生成相应的表定义文件，并同时生成相应的数据文件。通过FEDERATED引擎创建的表只是在本地有表定义文件，数据文件则存在于远程数据库中

1. 建立联合索引时，你会考虑哪些方面的问题

索引字段的顺序，最常用的放左边，值最多的放在最左边。

# Web

## Web基础

### 易

1. Session与cookies有什么区别

答：

1. cookie数据存放在客户的浏览器上，session数据放在服务器上。
2. cookie不是很安全，别人可以分析存放在本地的COOKIE并进行COOKIE欺骗（CSRF），考虑到安全应当使用session。
3. session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能，考虑到减轻服务器性能方面，应当使用COOKIE。
4. 单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie。
5. HTTP Get与Post有什么区别

答：

1. get是从服务器获取数据，Post是向服务器传送数据
2. get是把参数数据队列加到提交表单的Action属性所指的URL中，值和表单内容各个字段一一对应，在URL可以看到，Post是把表单字段内容等放于html的header内一起传送到Action属性所指的URL地址.不可见
3. get方式，服务端可以用Requset.Query获取变量值，Post方式，服务器用requset.form获取提交数据
4. get传输大小不可大于2KB，Post无限制
5. get不安全但效率高.Post安全
6. 数据查询用get，其他用Post

### 中

1. TCP与HTTP有什么区别

答：TPC/IP协议是传输层协议，主要解决数据如何在网络中传输，而HTTP是应用层协议，主要解决如何包装数据。关于TCP/IP和HTTP协议的关系，网络有一段比较容易理解的介绍：“我们在传输数据时，可以只使用（传输层）TCP/IP协议，但是那样的话，如果没有应用层，便无法识别数据内容，如果想要使传输的数据有意义，则必须使用到应用层协议，应用层协议有很多，比如HTTP、FTP、TELNET等，也可以自己定义应用层协议。WEB使用HTTP协议作应用层协议，以封装HTTP 文本信息，然后使用TCP/IP做传输层协议将它发到网络上。

### 难

1. Web开发常常会出现乱码问题，请对可能产生乱码的原因进行分析

答：http://blog.chinaunix.net/uid-200142-id-1762758.html

## Servlet

1. 请简单描述Servlet的生命周期

答：Servlet 生命周期：Servlet 加载--->实例化--->服务--->销毁。

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

答：前者仅是容器中控制权的转向，在客户端浏览器地址栏中不会显示出转向后的地址；

后者则是完全的跳转，浏览器将会得到跳转的地址，并重新发送请求链接。这样，从浏览器的地址栏中可以看到跳转后的链接地址。

所以，前者更加高效，在前者可以满足需要时，尽量使用forward()方法，并这样也有助于隐藏实际链接。

在有些情况下，比如，需要跳转到一个其它服务器上的资源，则必须使用sendRedirect()方法。

forward是服务器请求资源，服务器直接访问目标地址的URL，把那个URL的响应内容读取过来，然后把这些内容再发给浏览器，浏览器根本不知道服务器发送的内容是从哪儿来的，所以它的地址栏中还是原来的地址。

redirect就是服务端根据逻辑，发送一个状态码，告诉浏览器重新去请求那个地址，一般来说浏览器会用刚才请求的所有参数重新请求，所以session，request参数都可以获取。

## TCP/IP

### 易

1. 请简单描述TCP协议中的三次握手流程
2. 第一次握手：Client将标志位SYN置为1，随机产生一个值seq=J，并将该数据包发送给Server，Client进入SYN\_SENT状态，等待Server确认。
3. 第二次握手：Server收到数据包后由标志位SYN=1知道Client请求建立连接，Server将标志位SYN和ACK都置为1，ack=J+1，随机产生一个值seq=K，并将该数据包发送给Client以确认连接请求，Server进入SYN\_RCVD状态。
4. 第三次握手：Client收到确认后，检查ack是否为J+1，ACK是否为1，如果正确则将标志位ACK置为1，ack=K+1，并将该数据包发送给Server，Server检查ack是否为K+1，ACK是否为1，如果正确则连接建立成功，Client和Server进入ESTABLISHED状态，完成三次握手，随后Client与Server之间可以开始传输数据了。

### 中

1. 请简单描述TCP协议中的四次挥手流程

答：

由于TCP连接时全双工的，因此，每个方向都必须要单独进行关闭，这一原则是当一方完成数据发送任务后，发送一个FIN来终止这一方向的连接，收到一个FIN只是意味着这一方向上没有数据流动了，即不会再收到数据了，但是在这个TCP连接上仍然能够发送数据，直到这一方向也发送了FIN。首先进行关闭的一方将执行主动关闭，而另一方则执行被动关闭。

1. 第一次挥手：Client发送一个FIN，用来关闭Client到Server的数据传送，Client进入FIN\_WAIT\_1状态；
2. 第二次挥手：Server收到FIN后，发送一个ACK给Client，确认序号为收到序号+1（与SYN相同，一个FIN占用一个序号），Server进入CLOSE\_WAIT状态；
3. 第三次挥手：Server发送一个FIN，用来关闭Server到Client的数据传送，Server进入LAST\_ACK状态；
4. 第四次挥手：Client收到FIN后，Client进入TIME\_WAIT状态，接着发送一个ACK给Server，确认序号为收到序号+1，Server进入CLOSED状态，完成四次挥手。

### 难

1. 在TCP协议中，为什么建立连接时是三次握手，而断开连接时需要四次

答：因为服务端在LISTEN状态下，收到建立连接请求的SYN报文后，把ACK和SYN放在一个报文里发送给客户端。而关闭连接时，当收到对方的FIN报文时，仅仅表示对方不再发送数据了但是还能接收数据，己方也未必全部数据都发送给对方了，所以己方可以立即close，也可以发送一些数据给对方后，再发送FIN报文给对方来表示同意现在关闭连接，因此，己方ACK和FIN一般都会分开发送。

## Web服务器

### 易

1. 你用过哪些Web服务器

答：Tomcat、Apache、Nginx、 JBoss、 WebLogic

### 中

1. 这些Web服务器（Tomcat、JBoss、WebLogic）有什么区别

tomcat：

tomcat服务器占用资源少，稳定且免费。是一个轻量级的服务器，主要是应用于中小型项目 ，当并发访问的用户比较少时，可以选用tomcat服务器。tomcat服务器是运行jsp和servlet的很好的容器，但是它不支持EJB等。项目在tomcat中的部署很方便。

jboss：

jboss是一个基于j2ee的开放源码应用服务器，它也是免费的。它是一个管理ejb的容器，jboss核心服务仅支持ejb服务器，所以是不包括jsp和servlet的web容器。当然了，它可以和tomcat等进行绑定使用来同时支持jsp，servlet以及ejb的规范。jboss有一个典型的特点：当有servlet的系统调用到jboss里面的ejb时不经过网络，因为jboss和web服务器在同一个java虚拟机中运行，这可以大大提高运行效率和安全性。

weblogic：

而相比之下，weblogic服务器则功能更强大了一些，它属于应用级服务器，它不尽支持jsp和servlet，而且还支持更多的java的规范。 他用于开发，集成，部署和管理大型的分布式web应用，网络应用和数据库应用。这种大型的服务器有着自己独特的优势，即标准领先（它的标准包括ejb，jsb，jms，jdbc，xml和wml），扩展性无限（它的体系架构具有高扩展性，主要包括哭户籍连接的共享，资源pooling以及动态网页和ejb组件集群），快速开发（凭借对ejb和jsp的支持，以及其对servlet组件的架构体系，可加速部署应用），部署灵活，可靠等。但是一般的系统部署基本不会用到它，因为一般非基于ejb等的分布式开发项目，仅仅用tomcat即可满足我们的需求，所以无需动用重量级的weblogic。

站在技术支持的角度一句话来概括这三个服务器的话：即weblogic相当于tomcat和jboss结合在一起使用（因为weblogic支持servlet和jsp以及ejb，而tomcat仅支持servlet和jsp，jboss仅支持ejb）

### 难

1. 正向代理和反向代理有什么区别

答：

正向代理，也就是传说中的代理，他的工作原理就像一个跳板， 简单的说， 我是一个用户，我访问不了某网站，但是我能访问一个代理服务器 这个代理服务器呢，他能访问那个我不能访问的网站 于是我先连上代理服务器，告诉他我需要那个无法访问网站的内容 代理服务器去取回来，然后返回给我

从网站的角度，只在代理服务器来取内容的时候有一次记录 有时候并不知道是用户的请求，也隐藏了用户的资料，这取决于代理告不告诉网站

结论：正向代理 是一个位于客户端和原始服务器(origin server)之间的服务器，为了从原始服务器取得内容，客户端向代理发送一个请求并指定目标(原始服务器)，然后代理向原始服务器转交请求并将获得的内容返回给客户端。客户端必须要进行一些特别的设置才能使用正向代理。

反向代理

举例: 例用户访问 http://ooxx.me/readme 但ooxx.me上并不存在readme页面 他是偷偷从另外一台服务器上取回来，然后作为自己的内容吐给用户。

这里所提到的 ooxx.me 这个域名对应的服务器就设置了反向代理功能

结论： 反向代理正好相反，对于客户端而言它就像是原始服务器，并且客户端不需要进行任何特别的设置。客户端向反向代理 的命名空间(name-space)中的内容发送普通请求，接着反向代理将判断向何处(原始服务器)转交请求，并将获得的内容返回给客户端，就像这些内容 原本就是它自己的一样。

两者区别

从用途上来讲：

正向代理的典型用途是为在防火墙内的局域网客户端提供访问Internet的途径。正向代理还可以使用缓冲特性减少网络使用率。反向代理的典型用途是将 防火墙后面的服务器提供给Internet用户访问。反向代理还可以为后端的多台服务器提供负载平衡，或为后端较慢的服务器提供缓冲服务。

另外，反向代理还可以启用高级URL策略和管理技术，从而使处于不同web服务器系统的web页面同时存在于同一个URL空间下。

从安全性来讲：

正向代理允许客户端通过它访问任意网站并且隐藏客户端自身，因此你必须采取安全措施以确保仅为经过授权的客户端提供服务。

反向代理对外都是透明的，访问者并不知道自己访问的是一个代理。

# 设计模式

### 易

1. 你了解哪些设计模式，请对其中的两种做详细的介绍。

答：

单例模式是只包含一个被称为单例类的特殊类。通过单例模式可以保证系统中一个类只有一个实例而且该实例易于外界访问，从而方便对实例个数的控制并节约系统资源。

单例的实现要点：

一是单例模式的类只提供私有的构造函数，

二是类定义中含有一个该类的静态私有对象，

三是该类提供了一个静态的共有的函数用于创建或获取它本身的静态私有对象。

### 中

1. 简单工厂、工厂方法和抽象工厂模式有什么区别

简单工厂模式是属于创建型模式，又叫做静态工厂方法（Static Factory Method）模式，但不属于23种GOF设计模式之一。简单工厂模式是由一个工厂对象决定创建出哪一种产品类的实例。简单工厂模式是工厂模式家族中最简单实用的模式，可以理解为是不同工厂模式的一个特殊实现。

工厂方法(Factory Method)模式的意义是定义一个创建产品对象的工厂接口，将实际创建工作推迟到子类当中。核心工厂类不再负责产品的创建，这样核心类成为一个抽象工厂角色，仅负责具体工厂子类必须实现的接口，这样进一步抽象化的好处是使得工厂方法模式可以使系统在不修改具体工厂角色的情况下引进新的产品。

抽象工厂模式：当每个抽象产品都有多于一个的具体子类的时候，工厂角色怎么知道实例化哪一个子类呢？比如每个抽象产品角色都有两个具体产品。抽象工厂模式提供两个具体工厂角色，分别对应于这两个具体产品角色，每一个具体工厂角色只负责某一个产品角色的实例化。每一个具体工厂类只负责创建抽象产品的某一个具体子类的实例。

每一个模式都是针对一定问题的解决方案，工厂方法模式针对的是一个产品等级结构；而抽象工厂模式针对的是多个产品等级结构。

1. 请描述观察者模式的主要思想

观察者模式（Observer）完美的将观察者和被观察的对象分离开。举个例子，用户界面可以作为一个观察者，业务数据是被观察者，用户界面观察业务数据的变化，发现数据变化后，就显示在界面上。面向对象设计的一个原则是：系统中的每个类将重点放在某一个功能上，而不是其他方面。一个对象只做一件事情，并且将他做好。观察者模式在模块之间划定了清晰的界限，提高了应用程序的可维护性和重用性。

观察者设计模式定义了对象间的一种一对多的依赖关系，以便一个对象的状态发生变化时，所有依赖于它的独享都得到通知并自动刷新。

### 难

1. 策略模式和模板方法有什么异同点

策略模式定义了一系列的算法，并将每一个算法封装起来，而且使它们还可以相互替换。略模式让算法独立于使用它的客户而独立变化。

策略模式属于对象的行为模式。其用意是针对一组算法，将每一个算法封装到具有

共同接口的独立的类中，从而使得它们可以相互替换。策略模式使得算法可以在不影响到客户端的情况下发生变化。

模板方法是定义一个算法的骨架，而将一些实现步骤延迟到子类中。把不变的行为搬到超类，去除子类中重复的代码来体现他的优势。

# 加分项

## Nodejs

### 易

1. nodejs的适合应用场景有哪些？不适合的应用场景有哪些

答：适合：IO密集型。 不适合：CPU密集型。

### 中

1. 请简述nodejs使用的是哪种模块化设计方式？这种模式有什么特点？

答：CMD模式。特点：依赖就近，按需加载，在需要用到某个模块的时候去导入模块。

### 难

1. Nodejs如何利用多核cpu？

答：利用cluster API，根据cpu核数开启子进程。

## 缓存

### 易

1. 常用的缓存技术有哪些

答：Memcache Redis Ehcache

### 中

1. Memcached， Redis， MongoDB有哪些区别

答：

mongodb和memcached不是一个范畴内的东西。mongodb是文档型的非关系型数据库，其优势在于查询功能比较强大，能存储海量数据。mongodb和memcached不存在谁替换谁的问题。

和memcached更为接近的是redis。它们都是内存型数据库，数据保存在内存中，通过tcp直接存取，优势是速度快，并发高，缺点是数据类型有限，查询功能不强，一般用作缓存。在我们团队的项目中，一开始用的是memcached，后来用redis替代。

相比memcached：

* redis具有持久化机制，可以定期将内存中的数据持久化到硬盘上。
* redis具备binlog功能，可以将所有操作写入日志，当redis出现故障，可依照binlog进行数据恢复。
* redis支持virtual memory，可以限定内存使用大小，当数据超过阈值，则通过类似LRU的算法把内存中的最不常用数据保存到硬盘的页面文件中。
* redis原生支持的数据类型更多，使用的想象空间更大。
* 一致性哈希，用在redis的sharding中，一般是在负载非常高需要水平扩展时使用。我们还没有用到这方面的功能，一般的项目，单机足够支撑并发了。redis 3.0将推出cluster，功能更加强大。
* redis更多优点，请移步官方网站查询。
* 性能

Redis作者的说法是平均到单个核上的性能，在单条数据不大的情况下Redis更好。为什么这么说呢，理由就是Redis是单线程运行的。

因为是单线程运行，所以和Memcached的多线程相比，整体性能肯定会偏低。

因为是单线程运行，所以IO是串行化的，网络IO和内存IO，因此当单条数据太大时，由于需要等待一个命令的所有IO完成才能进行后续的命令，所以性能会受影响。

1. Redis有哪几种持久化方式

答：

a）RDB持久化：

该机制是指在指定的时间间隔内将内存中的数据集快照写入磁盘。

b）AOF持久化:

该机制将以日志的形式记录服务器所处理的每一个写操作，在Redis服务器启动之初会读取该文件来重新构建数据库，以保证启动后数据库中的数据是完整的。

c）无持久化：

我们可以通过配置的方式禁用Redis服务器的持久化功能，这样我们就可以将Redis视为一个功能加强版的memcached了。

d）同时应用AOF和RDB。

参考：http://www.cnblogs.com/stephen-liu74/archive/2012/02/23/2365017.html

### 难

1. 请简单描述Redis的一致性哈希算法

答：参考http://blog.csdn.net/x15594/article/details/6270242

## 框架技术

## Spring

### IOC

1. IOC主要思想是什么，它有哪些好处

答：通过引入IOC容器，利用依赖关系注入的方式，实现对象之间的解耦。

参考：http://blog.csdn.net/m13666368773/article/details/7802126

1. 如果需要自己实现IOC，你会怎么做

参考：http://blog.csdn.net/it\_man/article/details/4402245

### AOP

#### 易

1. AOP主要思想是什么，有哪些常见的场景会用到AOP

答：权限、日志、

#### 中

1. 什么是动态代理

答：<http://langyu.iteye.com/blog/410071>

1. Spring 提供了哪几种生成代理对象的方式  
   答：Spring提供了两种方式来生成代理对象: JDKProxy和Cglib。

JDKProxy：通过在运行期间创建一个接口的实现类来完成对目标对象的代理；

Cglib：它在运行期间生成的代理对象是针对目标类扩展的子类；

#### 难

1. Spring是如何决定何时采用JDKProxy，何时采用Cglib的

答：具体使用哪种方式生成由AopProxyFactory根据AdvisedSupport对象的配置来决定，默认的策略是如果目标类是接口，则使用JDK动态代理技术，否则使用Cglib来生成代理。

## Struts

1. 请简述Struts响应客户请求的工作流程
2. 客户端发出一个指向servlet容器的请求
3. 这个请求会经过几个过滤器，最后会到达FilterDispatcher过滤器。
4. 过滤器FilterDispatcher是struts2框架的心脏，在处理用户请求时，它和请求一起相互配合访问struts2的底层框架结构。在web容器启动时，struts2框架会自动加载配置文件里相关参数，并转换成相应的类。如：ConfigurationManager、ActionMapper和ObjectFactory。ConfigurationManager 存有配置文件的一些基本信息，ActionMapper存有action的配置信息。在请求过程中所有的对象（Action，Results，Interceptors，等）都是通过ObjectFactory来创建的。过滤器会通过询问ActionMapper类来查找请求中需要用到的Action。
5. 如果找到需要调用的Action，过滤器会把请求的处理交给ActionProxy。ActionProxy为Action的代理对象。ActionProxy通过ConfigurationManager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。ActionInvocation在ActionProxy层之下，它表示了Action的执行状态,或者说它控制的Action的执行步骤。它持有Action实例和所有的Interceptor。
7. ActionInvocation实例使用命名模式来调用，1. ActionInvocation初始化时，根据配置，加载Action相关的所有Interceptor。2. 通过ActionInvocation.invoke方法调用Action实现时，执行Interceptor。在调用Action的过程前后，涉及到相关拦截器(intercepetor)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

## Hibernate/Ibatis

### 易

1. 请简单介绍Hibernate的工作流程及其优点

答：

工作流程

1. 读取并解析配置文件
2. 读取并解析映射信息，创建SessionFactory
3. 打开Sesssion
4. 创建事务Transaction
5. 持久化操作
6. 提交事务
7. 关闭Session
8. 关闭SessionFactory

优点：

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。

### 中

1. 请简述Hibernate中lazy-load的基本概念

Hibernate 的延迟加载（lazy load）是一个被广泛使用的技术。这种延迟加载保证了应用只有在需要时才去数据库中抓取相应的记录。通过延迟加载技术可以避免过多、过早地加载数据表里的数据，从而降低应用的内存开销。Hibernate 的延迟加载本质上就是代理模式的应用，当程序通过 Hibernate 装载一个实体时，默认情况下，Hibernate 并不会立即抓取它的集合属性、关联实体所以对应的记录，而是通过生成一个代理来表示这些集合属性、关联实体，这就是代理模式应用带来的优势。

1. Hibernate有哪些主键生成机制
2. Assigned:主键由外部程序负责生成，无需Hibernate参与。
3. hilo:通过hi/lo 算法实现的主键生成机制，需要额外的数据库表保存主键生成历史状态。
4. seqhilo:与hilo 类似，通过hi/lo 算法实现的主键生成机制，只是主键历史状态保存在Sequence中，适用于支持Sequence的数据库，如Oracle。
5. increment:主键按数值顺序递增。此方式的实现机制为在当前应用实例中维持一个变量，以保存着当前的最大值，之后每次需要生成主键的时候将此值加1作为主键。这种方式可能产生的问题是：如果当前有多个实例访问同一个数据库，那么由于各个实例各自维护主键状态，不同实例可能生成同样的主键，从而造成主键重复异常。因此，如果同一数据库有多个实例访问，此方式必须避免使用。
6. identity:采用数据库提供的主键生成机制。如DB2、SQL Server、MySQL中的主键生成机制。
7. sequence:采用数据库提供的sequence 机制生成主键。如Oralce 中的Sequence。
8. native:由Hibernate根据底层数据库自行判断采用identity、hilo、sequence其中一种作为主键生成方式。
9. uuid.hex:由Hibernate基于128 位唯一值产生算法生成16 进制数值（编码后以长度32 的字符串表示）作为主键。
10. uuid.string:与uuid.hex 类似，只是生成的主键未进行编码（长度16）。在某些数据库中可能出现问题（如PostgreSQL）。
11. foreign:使用外部表的字段作为主键。
12. 在Hibernate中，对象有哪几种状态

答：

a）瞬时状态

在数据库中没有与之匹配的数据,没有纳入session的管理，执行完直接销毁

b）持久状态

在数据库中有与之匹配的数据,纳入session的管理, 在清理缓存的时候（脏数据检测),会时刻与数据库中数据保存统一

c）托管状态

在数据库中有与之匹配的数据,没有纳入session的管理

总结： hibernate做save或update操作，一般都会根据对象状态来操作，企业开发中一般都是用saveOrUpdate来替代save或update方法

### 难

1. 请简述Hibernate的缓存策略

答：参考：http://www.blogjava.net/frostwood/archive/2010/01/06/308465.html

## 其它

### Linux

#### 易

1. 你用过哪些Linux系统，它们各自有什么特点，你比较熟练的是哪一种

答：

a）Fedora

Fedora Core和其他三个一样，也很出色！可以这么说每一个Linux版本都很优秀，中国的Red Flag也不错，对中文的支持很好，少了安装中文字体的麻烦！ 比较符合国人的使用习惯！Ylmf OS那就更简单了，如果你比较懒，可以用它，集成了好多常用软件，足够你使用的了！

b）Redhat

目前许多高手都使用它，长相一般但却很强悍，一般是Linux高手的首选系统，相比之下就是桌面系统有点不大一样，拥有强大的rpm软件包管理系统，界面更加简洁，如果你不喜欢太多花哨的桌面系统可以考虑用它！

Redhat，应该称为Redhat系列，包括RHEL(Redhat Enterprise Linux，也就是所谓的Redhat Advance Server，收费版本)、Fedora Core(由原来的Redhat桌面版本发展而来，免费版本)、CentOS(RHEL的社区克隆版本，免费)。Redhat应该说是在国内使用人群最多的Linux版本，甚至有人将Redhat等同于Linux，而有些老鸟更是只用这一个版本的Linux。所以这个版本的特点就是使用人群数量大，资料非常多，言下之意就是如果你有什么不明白的地方，很容易找到人来问，而且网上的一般Linux教程都是以Redhat为例来讲解的。Redhat系列的包管理方式采用的是基于RPM包的YUM包管理方式，包分发方式是编译好的二进制文件。稳定性方面RHEL和CentOS的稳定性非常好，适合于服务器使用，但是Fedora Core的稳定性较差，最好只用于桌面应用。

c）Debian

Debian,或者称Debian系列，包括Debian和Ubuntu等。Debian是社区类Linux的典范，是迄今为止最遵循GNU规范的Linux系统。Debian最早由Ian Murdock于1993年创建，分为三个版本分支（branch）： stable, testing 和 unstable。其中，unstable为最新的测试版本，其中包括最新的软件包，但是也有相对较多的bug，适合桌面用户。testing的版本都经过unstable中的测试，相对较为稳定，也支持了不少新技术（比如SMP等）。而stable一般只用于服务器，上面的软件包大部分都比较过时，但是稳定和安全性都非常的高。Debian最具特色的是apt-get / dpkg包管理方式，其实Redhat的YUM也是在模仿Debian的APT方式，但在二进制文件发行方式中，APT应该是最好的了。Debian的资料也很丰富，有很多支持的社区，有问题求教也有地方可去:)

d）Ubuntu

目前使用最多的Linux，简单方便，有KDE和GNOME等视窗界面可供选择，拥有强大的apt-get软件管理程序，安装个管理软件很方便！作为新手，推荐使用。

Ubuntu严格来说不能算一个独立的发行版本，Ubuntu是基于Debian的unstable版本加强而来，可以这么说，Ubuntu就是一个拥有Debian所有的优点，以及自己所加强的优点的近乎完美的 Linux桌面系统。根据选择的桌面系统不同，有三个版本可供选择，基于Gnome的Ubuntu，基于KDE的Kubuntu以及基于Xfc的Xubuntu。特点是界面非常友好，容易上手，对硬件的支持非常全面，是最适合做桌面系统的Linux发行版本。

e）Gentoo

Gentoo,伟大的Gentoo是Linux世界最年轻的发行版本，正因为年轻，所以能吸取在她之前的所有发行版本的优点，这也是Gentoo被称为最完美的Linux发行版本的原因之一。Gentoo最初由Daniel Robbins（FreeBSD的开发者之一）创建，首个稳定版本发布于2002年。由于开发者对FreeBSD的熟识，所以Gentoo拥有媲美FreeBSD的广受美誉的ports系统 ——Portage包管理系统。不同于APT和YUM等二进制文件分发的包管理系统，Portage是基于源代码分发的，必须编译后才能运行，对于大型软件而言比较慢，不过正因为所有软件都是在本地机器编译的，在经过各种定制的编译参数优化后，能将机器的硬件性能发挥到极致。Gentoo是所有Linux发行版本里安装最复杂的，但是又是安装完成后最便于管理的版本，也是在相同硬件环境下运行最快的版本。

f）FreeBSD

FreeBSD并不是一个Linux系统！但FreeBSD与Linux的用户群有相当一部分是重合的，二者支持的硬件环境也比较一致，所采用的软件也比较类似，所以可以将FreeBSD视为一个Linux版本来比较。FreeBSD拥有两个分支：stable和current。顾名思义，stable是稳定版，而 current则是添加了新技术的测试版。FreeBSD采用Ports包管理系统，与Gentoo类似，基于源代码分发，必须在本地机器编后后才能运行，但是Ports系统没有Portage系统使用简便，使用起来稍微复杂一些。FreeBSD的最大特点就是稳定和高效，是作为服务器操作系统的最佳选择，但对硬件的支持没有Linux完备，所以并不适合作为桌面系统。

g）OpenSUSE

OpenSUSE：在欧洲非常流行的一个Linux，由Novell公司发放，号称是世界上最华丽的操作系统，独家开发的软件管理程序zypper || yast得到了许多用户的赞美，和ubuntu一样，支持kde和gnome，xface等桌面，桌面特效比较丰富，缺点是KDE虽然华丽多彩，但比较不 稳定。新手用这个也很容易上手。

#### 中

1. 在Linux里，你最常用的有哪些命令

答：文件管理类、系统管理类、vim类等

#### 难

1. 对SELinux是否有了解，请简单介绍

答：

SELinux是一种基于 域-类型 模型（domain-type）的强制访问控制（MAC）安全系统，它由NSA编写并设计成内核模块包含到内核中，相应的某些安全相关的应用也被打了SELinux的补丁，最后还有一个相应的安全策略。

标准的UNIX安全模型是"任意的访问控制"DAC。就是说，任何程序对其资源享有完全的控制权。假设某个程序打算把含有潜在重要信息的文件仍到/tmp目录下，那么在DAC情况下没人能阻止他！

而MAC情况下的安全策略完全控制着对所有资源的访问。这是MAC和DAC本质的区别。

### 并发

1. 如何合理的配置java线程池？如CPU密集型的任务，基本线程池应该配置多大？IO密集型的任务，基本线程池应该配置多大？用有界队列好还是无界队列好？任务非常多的时候，使用什么阻塞队列能获取最好的吞吐量？

答：

1. 配置线程池时CPU密集型任务可以少配置线程数，大概和机器的cpu核数相当，可以使得每个线程都在执行任务；
2. IO密集型时，大部分线程都阻塞，故需要多配置线程数，2\*cpu核数；
3. 有界队列和无界队列的配置需区分业务场景，一般情况下配置有界队列，在一些可能会有爆发性增长的情况下使用无界队列；
4. 任务非常多时，使用非阻塞队列使用CAS操作替代锁可以获得好的吞吐量。
5. 多读少写的场景应该使用哪个并发容器，为什么使用它？比如你做了一个搜索引擎，搜索引擎每次搜索前需要判断搜索关键词是否在黑名单里，黑名单每天更新一次。

答：

1. CopyOnWriteArrayList这个容器适用于多读少写；
2. 读写并不是在同一个对象上。在写时会大面积复制数组，所以写的性能差，在写完成后将读的引用改为执行写的对象；

### Web安全

#### 易

1. 有哪些SQL 注入的防范措施

答：关键字过滤、sql预编译等

#### 中

1. CSRF和XSS有什么区别

XSS（Cross-site scripting），为了和CSS层叠样式表区分所以取名XSS。是一种网站应用程序的安全漏洞攻击，是代码注入的一种。它允许恶意用户将代码注入到网页上，其他用户在观看网页时就会受到影响。这类攻击通常包含了HTML以及用户端脚本语言。

CSRF（Cross-site request forgery跨站请求伪造，也被称为“one click attack”或者session riding，通常缩写为CSRF或者XSRF，是一种对网站的恶意利用。尽管听起来像跨站脚本（XSS），但它与XSS非常不同，并且攻击方式几乎相左。XSS利用站点内的信任用户，而CSRF则通过伪装来自受信任用户的请求来利用受信任的网站。

#### 难

1. 如何防范服务器被攻击
2. Sql注入防范
3. CSRF和XSS防范
4. 设置Web目录的访问权限
5. 最小服务原则。Web服务器应以最小权限进行配置，只提供Web服务，这样可以有效地阻止系统的危险命令，如ftp、cmd、vbscript等。
6. 鉴别信息加密存储。将保存在数据库users表中的用户名、口令信息以密文形式保存，也可以对users表进行加密处理，这样可以大大增加对鉴别信息访问的安全级别。
7. 用户权限分离。应尽可能的禁止或删除数据库中sa权限用户的访问，对不同的数据库划分不同的用户权限，这样不同的用户只能对授权给自己的数据库执行查询、插入、更新、删除操作，就可以防止不同用户对非授权的数据库进行访问。