**Java面试题库**

这套面试题主要目的是帮助那些还没有java软件开发实际工作经验，而正在努力寻找java软件开发工作的朋友在笔试时更好地赢得笔试和面试。由于这套面试题涉及的范围很泛，很广，很杂，大家不可能一天两天就看完和学完这套面试宝典，即使你已经学过了有关的技术，那么至少也需要一个月的时间才能消化和掌握这套面试宝典，所以，大家应该早作准备，从拿到这套面试题库之日起，就要坚持在每天闲暇之余学习其中几道题目，日积月累，等到出去面试时，一切都水到渠成，面试时就自然会游刃有余了。

答题时，先答是什么，再答有什么作用和要注意什么（这部分最重要，展现自己的心得）

答案的段落分别，层次分明，条理清晰都非常重要，从这些表面的东西也可以看出一个人的习惯、办事风格、条理等。

要讲你做出答案的思路过程，或者说你记住答案的思想都写下来。把答题想着是辩论赛。答题就是给别人讲道理、摆事实。答题不局限于什么格式和形式，就是要将自己的学识展现出来！

别因为人家题目本来就模棱两可，你就心里胆怯和没底气了，不敢回答了。你要大胆地指出对方题目很模糊和你的观点，不要把面试官想得有多高，其实他和你就是差不多的，你想想，如果他把你招进去了，你们以后就是同事了，可不是差不多的吗？

## Java基础部分

基础部分的顺序：基本语法，类相关的语法，内部类的语法，继承相关的语法，异常的语法，线程的语法，集合的语法，io 的语法，虚拟机方面的语法。

### 1、一个".java"源文件中是否可以包括多个类（不是内部类）？有什么限制？

可以有多个类，但只能有一个public的类，并且public的类名必须与文件名相一致。

### 2、Java有没有goto?

java中的保留字，现在没有在java中使用。

### 3、说说&和&&的区别。

&和&&都可以用作逻辑与的运算符，表示逻辑与（and），当运算符两边的表达式的结果都为true时，整个运算结果才为true，否则，只要有一方为false，则结果为false。

&&还具有短路的功能，即如果第一个表达式为false，则不再计算第二个表达式。

&还可以用作位运算符，当&操作符两边的表达式不是boolean类型时，&表示按位与操作。

### 4、short s1 = 1; s1 = s1 + 1;有什么错? short s1 = 1; s1 += 1;有什么错?

对于short s1 = 1; s1 = s1 + 1; 由于s1+1运算时会自动提升表达式的类型，所以结果是int型，再赋值给short类型s1时，编译器将报告需要强制转换类型的错误。

对于short s1 = 1; s1 += 1;由于 += 是java语言规定的运算符，java编Z译器会对它进行特殊处理，因此可以正确编译。

### 5、char型变量中能不能存储一个中文汉字?为什么?

char型变量是用来存储Unicode编码的字符的，unicode编码字符集中包含了汉字，所以，char型变量中当然可以存储汉字啦。不过，如果某个特殊的汉字没有被包含在unicode编码字符集中，那么，这个char型变量中就不能存储这个特殊汉字。补充说明：unicode编码占用两个字节，所以，char类型的变量也是占用两个字节。

### 6、用最有效率的方法算出2乘以8等於几?

2 << 3，

因为将一个数左移n位，就相当于乘以了2的n次方，那么，一个数乘以8只要将其左移3位即可，而位运算cpu直接支持的，效率最高，所以，2乘以8等於几的最效率的方法是2 << 3。

### 7、什么是方法重载和方法重写，有什么异同？

方法重载发生在同一个类中,同名不同参的方法;

方法重写是子类重写父类的方法,同名同参;

两者区别：语法区别,重载要求同名不同参,与访问修饰符 、返回类型、异常无关； 重写要求同名同参、访问修饰符必须大于等于父类,返回类型要一致（基本数据类型必须相同）,子类抛出的异常必须是父类异常的子类或子集！

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

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

### 9、"=="和equals方法究竟有什么区别？

1.“==”常用来比较基本数据类型，8种基本数据类型有byte、short、long、double、char、int、float、boolean，

因为变量直接存储的就是他们的值，所以用"=="去比较，比较的就是他们的值。

但是复合数据类型用“==”比较的是他的堆内存地址。

2.“equals”对于复合数据类型比较的也是它的堆内存地址（不能作用于基本数据类型的变量）。

但在实际项目中常用来比较String字符串的内容相等是为什么呢？

那是因为对于String（还有Date，Integer）类型重写了equals方法，使其比较的是存储对象的内容是否相等，而不是堆内存地址。

### 10、静态变量和实例变量的区别？

区别：静态变量前要加static关键字，而实例变量前则不加。

运行时的区别：实例变量属于某个对象的属性，必须创建了实例对象，其中的实例变量才会被分配空间，才能使用这个实例变量。静态变量不属于某个实例对象，而是属于类，所以也称为类变量，只要程序加载了类的字节码，不用创建任何实例对象，静态变量就会被分配空间，静态变量就可以被使用了。总之，实例变量必须创建对象后才可以通过这个对象来使用，静态变量则可以直接使用类名来引用。

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

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

### 12、Integer与int的区别

int是java提供的8种原始数据类型之一。Java为每个原始类型提供了封装类，Integer是java为int提供的封装类。int的默认值为0，而Integer的默认值为null，即Integer可以区分出未赋值和值为0的区别，int则无法表达出未赋值的情况，Integer提供了多个与整数相关的操作方法，例如，将一个字符串转换成整数，Integer中还定义了表示整数的最大值和最小值的常量。

### 13、构造器Constructor是否可被override?

构造器Constructor不能被继承，因此不能重写Override，但可以被重载Overload。

### 14、接口是否可继承接口? 抽象类是否可实现接口? 抽象类是否可继承具体类? 抽象类中是否可以有main方法？

接口可以继承接口。抽象类可以实现(implements)接口，抽象类可以继承具体类。抽象类中可以有静态的main方法。

### 15、写clone()方法时，通常都有一行代码，是什么？

clone 有缺省行为，super.clone();因为首先要把父类中的成员复制到位，然后才是复制自己的成员。

### 16、面向对象的特征有哪些方面

1封装：

封装是类内部细节隐藏起来，对外提供公共访问的接口，封装的目标就是要实现软件部 件的“高内聚、低耦合”。1.将类中属性私有化，2.对外提供公有的访问方法set/get方 法 3.在set/get中加入逻辑判断

2.继承：

在定义和实现一个类的时候，可以在一个已经存在的类的基础之上，把这个已经存在的类所定义的内容作为自己的内容，并可以加入若干新的内容，这就是继承。继承是子类自动共享父类数据和方法的机制，这是类之间的一种关系，提高了软件的可重用性和可扩展性。

3.多态：

多态是指程序中定义的引用变量所指向的具体类型和通过该引用变量发出的方法调用在编程时并不确定，而是在程序运行期间才确定，即一个引用变量倒底会指向哪个类的实例对象，该引用变量发出的方法调用到底是哪个类中实现的方法，必须在由程序运行期间才能决定。这就是多态性。多态性增强了软件的灵活性和扩展性。

### 17、抽象类和接口有什么异同？

语法上：

抽象类有构造方法、属性、方法、可以包含零个或多个抽象方法;

接口则无构造方法,所有属性都是静态常量、所有方法都是抽象方法 (jdk8可以有默认和静态方法);

设计理念上:

抽象类代表的是子类是什么，是is-a的关系,接口代表的是子类能做什么, 是一种约束和能力体现，是can-do的关系。

### 18、abstract的method是否可同时是static,是否可同时是native，是否可同时是synchronized?

abstract的method 不可以是static的，因为抽象的方法是要被子类实现的。

native方法表示该方法要用另外一种依赖平台的编程语言实现的，不存在着被子类实现的问题，所以，它也不能是抽象的，不能与abstract混用。

synchronized与abstract合用的问题，不能，synchronized应该是作用在一个具体的方法上才有意义。而且，方法上的synchronized同步所使用的同步锁对象是this，而抽象方法上无法确定this是什么。

### 19、什么是内部类？Static Nested Class 和 Inner Class的不同。

将一个类定义在另一个类里面或者一个方法里面，这样的类称为内部类。

内部类分类：成员内部类、局部内部类、匿名内部类和静态内部类。  
静态的Static nested class是不可以直接调用它的外部类enclosing class的，但是可以通过外部类的引用来调用，就像你在一个类中写了main方法一样。   
非静态类inner class 可以自由的引用外部类的属性和方法，但是它与一个实例绑定在了一起，不可以定义静态的属性、方法 。   
Inner Class（内部类）定义在类中的类。 Nested Class（嵌套类）是静态（static）内部类。

### 20、内部类可以引用它的包含类的成员吗？有没有什么限制？

完全可以。如果不是静态内部类，那没有什么限制！

如果你把静态嵌套类当作内部类的一种特例，那在这种情况下不可以访问外部类的普通成员变量，而只能访问外部类中的静态成员。

### 21、Anonymous Inner Class (匿名内部类) 是否可以extends(继承)其它类，是否可以implements(实现)interface(接口)?

可以继承其他类或实现其他接口。不仅是可以，而是必须!

**22、类和类之间有哪几种关系？面向对象设计原则有哪些？**

类与类之间有 ：继承、实现、依赖、关联、聚合 、组合关系；

面向对象设计原则：

1.依赖倒置原则 2.迪米特法则3.接口隔离原则4.聚合/组合原则

5.里氏替换原则 6.开闭原则7.单一职能原则

### 23、String是最基本的数据类型吗?

基本数据类型包括byte、int、char、long、float、double、boolean和short。

java.lang.String类是final类型的，因此不可以继承这个类、不能修改这个类。

为了提高效率节省空间，我们应该用StringBuffer类

### 24、String s = "Hello";s = s + " world!";这两行代码执行后，原始的String对象中的内容到底变了没有？

没有。因为String被设计成不可变(immutable)类，所以它的所有对象都是不可变对象。在这段代码中，s原先指向一个String对象，内容是 "Hello"，然后我们对s进行了+操作，那么s所指向的那个对象是否发生了改变呢？答案是没有。这时，s不指向原来那个对象了，而指向了另一个 String对象，内容为"Hello world!"，原来那个对象还存在于内存之中，只是s这个引用变量不再指向它了。

### 25、String s = new String("xyz");创建了几个String Object? 二者之间有什么区别？

两个或一个，”xyz”对应一个对象，这个对象放在字符串常量缓冲区，常量”xyz”不管出现多少遍，都是缓冲区中的那一个。New String每写一遍，就创建一个新的对象，它一句那个常量”xyz”对象的内容来创建出一个新String对象。如果以前就用过’xyz’，这句代表就不会创建”xyz”自己了，直接从缓冲区拿。

### 26、String 和StringBuffer的区别

String和StringBuffer，它们可以储存和操作字符串，即包含多个字符的字符数据。这个String类提供了数值不可改变的字符串。而这个StringBuffer类提供的字符串进行修改。当你知道字符数据要改变的时候你就可以使用StringBuffer。

### 27、如何区分运行时异常与检查异常，他们使用时有何区别？

RuntimeException及其所有子类都属于运行时异常，在程序中可以不用处理，直接由jvm处理，而检查异常必须在程序中处理。

### 28、什么是java序列化，如何实现java序列化？

序列化就是将内存中的对象保存到硬盘或传输到网络中

先将需要序列化的类实现Serializable接口，

然后通过ObjectOutputStream或XmlEncoder类的writeObject()方法实现

### 29、final, finally, finalize的区别。

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

内部类要访问局部变量，局部变量必须定义成final类型，

finally是异常处理语句结构的一部分，表示总是执行。

finalize是Object类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，可以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等。JVM不保证此方法总被调用

### 31、error和exception有什么区别?

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。 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。

### 32、Java中的异常处理机制的简单原理和应用。

异常是指java程序运行时发生的非正常情况或错误，Java使用面向对象的方式来处理异常，它把程序中发生的每个异常也都分别封装到一个对象来表示的，该对象中包含有异常的信息。

异常的根类为java.lang.Throwable，Throwable它的两个子类：Error和Exception，

Error 表示应用程序本身无法克服和恢复的一种严重问题。

Exception表示程序还能够克服和恢复的问题，又分为运行时异常和编译异常。

可以使用几个关键字try ，catch ，finally， throw ，throws处理异常

### 33、请写出你最常见到的5个runtime exception。

RuntimeException异常有：

[NullPointerException](mk:@MSITStore:D:\\Resources\\课件资料\\帮助文档\\JDK_API_1_6_zh_CN.CHM::/java/lang/../../java/lang/NullPointerException.html" \o "java.lang 中的类) 空指针异常

[ArrayIndexOutOfBoundsException](mk:@MSITStore:D:\\Resources\\课件资料\\帮助文档\\JDK_API_1_6_zh_CN.CHM::/java/lang/../../java/lang/ArrayIndexOutOfBoundsException.html" \o "java.lang 中的类) 数组角标越界异常

[ClassCastException](mk:@MSITStore:D:\\Resources\\课件资料\\帮助文档\\JDK_API_1_6_zh_CN.CHM::/java/lang/../../java/lang/ClassCastException.html" \o "java.lang 中的类) 类型转换异常

[IllegalArgumentException](mk:@MSITStore:D:\\Resources\\课件资料\\帮助文档\\JDK_API_1_6_zh_CN.CHM::/java/lang/../../java/lang/IllegalArgumentException.html" \o "java.lang 中的类) 非法参数异常

[ArithmeticException](mk:@MSITStore:D:\\Resources\\课件资料\\帮助文档\\JDK_API_1_6_zh_CN.CHM::/java/lang/../../java/lang/ArithmeticException.html" \o "java.lang 中的类) 算术异常

**34、sleep() 、yield() 和 wait() 有什么区别?**

1、wait()方法是Object的方法,sleep()是Thread的方法

2、wait()方法必须写在synchronized里面

3、(重要)wait()方法在等待的时候会释放对象锁，sleep()睡过去也抱着对象锁不放！

4、yield()让当前正在执行的线程暂停该方法不会阻塞线程，而是将线程转入可运行状态

**35、java中如何编写一个线程类，并启动他？线程同步如何避免死锁？**

答：继承Java.lang.Thread类，并覆盖run() 方法，调用该对象的start()方法启动线程

实现Java.lang.Runnable接口，并实现run() 方法，将该对象实例作为Thread构造的参数再调用start()方法，使用了代理模式。

使锁数量尽可能少，保证它们的顺序在程序中一致，避免嵌套锁。

尝试定时的锁，使用每个显式Lock类中定时tryLock特性，来替代使用内部锁机制

### 36、同步和异步有何异同，在什么情况下分别使用他们？举例说明。

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率。

### 37、多线程有几种实现方法?同步有几种实现方法?

多线程有两种实现方法，分别是继承Thread类与实现Runnable接口

同步的实现方面有两种，分别是synchronized,wait与notify

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，注意并不是给所有唤醒线程一个对象的锁，而是让它们竞争。

### 38、当一个线程进入一个对象的一个synchronized方法后，其它线程是否可进入此对象的其它方法?

分几种情况：

1.其他方法前是否加了synchronized关键字，如果没加，则能。

2.如果这个方法内部调用了wait，则可以进入其他synchronized方法。

3.如果其他个方法都加了synchronized关键字，并且内部没有调用wait，则不能。

4.如果其他方法是static，它用的同步锁是当前类的字节码，与非静态的方法不能同步，因为非静态的方法用的是this。

**39、如何实现线程之间的通信？**

Java提供了3个方法解决线程之间的通信问题

wait() ：表示线程一直等待，并释放对象锁，直到其它线程通知再找回对象锁

notify() ：唤醒一个同一个对象上处于等待状态的线程

notifyAll()：唤醒同一个对象上所有调用wait()方法的线程，优先级别高的线程优先运行

### 40、简述synchronized和java.util.concurrent.locks.Lock的异同 ？

主要相同点：Lock能完成synchronized所实现的所有功能

主要不同点：Lock有比synchronized更精确的线程语义和更好的性能。synchronized会自动释放锁，而Lock一定要求程序员手工释放，并且必须在finally从句中释放。Lock还有更强大的功能，例如，它的tryLock方法可以非阻塞方式去拿锁。

### 41、说出ArrayList,Vector, LinkedList的存储性能和特性？

ArrayList和Vector都是使用数组方式存储数据，它们都允许按索引查询，但是插入元素要涉及数组元素移动等内存操作，所以查询快而插入删除慢，Vector由于使用了synchronized方法，性能上较ArrayList差，而LinkedList使用双向链表实现，按索引查询需要向前或向后遍历，插入数据只要记录前后项，所以插入删除较快。

### 42、HashMap和Hashtable的区别

1.（同步性）HashTable的方法是同步的，HashMap不能同步。

2.（继承的父类不同)HashTable是继承自Dictionary类，而HashMap是继承自AbstractMap类。不过它们都实现了同时实现了map、Cloneable（可复制）、Serializable（可序列化）这三个接口。

3.（对null key和null value的支持不同）.HashTable不允许null值（key和value都不可以），HashMap允许使用null值（key和value）都可以。这样的键只有一个，可以有一个或多个键所对应的值为null。

4.（遍历方法不同）HashTable使用Enumeration遍历，HashMap使用Iterator进行遍历。

5.（初始化和扩容方式不同）HashTable中hash数组初始化大小及扩容方式不同。

Hashtable默认的初始大小为11，之后每次扩充，容量变为原来的2n+1。HashMap默认的初始化大小为16。之后每次扩充，容量变为原来的2倍。

创建时，如果给定了容量初始值，那么Hashtable会直接使用你给定的大小，而HashMap会将其扩充为2的幂次方大小。也就是说Hashtable会尽量使用素数、奇数。而HashMap则总是使用2的幂作为哈希表的大小。

### 43、List 和 Map 区别?

一个是存储单列数据的集合，另一个是存储键和值这样的双列数据的集合，List中存储的数据是有顺序，并且允许重复；Map中存储的数据是没有顺序的，其键是不能重复的，它的值是可以有重复的。

### 44、Collection 和 Collections的区别。

　　Collection是集合类的上级接口，继承与他的接口主要有Set 和List.

Collections是针对集合类的一个帮助类，他提供一系列静态方法实现对各种集合的搜索、排序、线程安全化等操作。

### 45、两个对象值相同(x.equals(y) == true)，但却可有不同的hash code，这句话对不对?

正确：

如果对象要保存在HashSet或HashMap中，它们的equals相等，那么，它们的hashcode值就必须相等。

如果不是要保存在HashSet或HashMap，则与hashcode没有什么关系了，这时候hashcode不等是可以的，例如arrayList存储的对象就不用实现hashcode，当然，我们没有理由不实现，通常都会去实现的。

**46、Set里的元素是不能重复的，那么用什么方法来区分重复与否呢?Java中是否有内存泄漏**

答：Set里的元素是不能重复的，用equals和hashCode方法来区分重复与否。添加元素时先判断hashCode然后再判断equals。

垃圾回收机制只能回收没有引用的对象，在set集合添加了元素后又将计算hash值的属性修改了将导致内存泄漏。单例对象在JVM的整个生命周期中存在，如果单例对象持有外部对象的引用，那么外部对象将不能被jvm回收，所以不正确使用单例也能引起内存泄露。

### 47、java中有几种类型的流？JDK为每种类型的流提供了一些抽象类以供继承，请说出他们分别是哪些类？

字节流，字符流。字节流继承于InputStream， OutputStream，字符流继承于InputStreamReader， OutputStreamWriter。在java.io包中还有许多其他的流，主要是为了提高性能和使用方便。

### 48、字符流和字节流有什么区别？

字符流和字节流的使用非常相似，但是实际上字节流的操作不会经过缓冲区（内存）而是直接操作文本本身的，而字符流的操作会先经过缓冲区（内存）然后通过缓冲区再操作文件

### 49、字节流和字符流哪个好？怎么选择？

缓大多数情况下使用字节流会更好，因为字节流是字符流的包装，而大多数时候 IO 操作都是直接操作磁盘文件，所以这些流在传输时都是以字节的方式进行的（图片等都是按字节存储的）

如果对于操作需要通过 IO 在内存中频繁处理字符串的情况使用字符流会好些，因为字符流具备缓冲区，提高了性能

### 50、描述一下JVM加载class文件的原理机制?

JVM中类的装载是由ClassLoader和它的子类来实现的,Java ClassLoader 是一个重要的Java运行时系统组件。它负责在运行时查找和装入类文件的类。

### 51、heap和stack有什么区别。

java的内存分为两类，一类是栈内存，一类是堆内存。栈内存是指程序进入一个方法时，会为这个方法单独分配一块私属存储空间，用于存储这个方法内部的局部变量，当这个方法结束时，分配给这个方法的栈会释放，这个栈中的变量也将随之释放。

堆是与栈作用不同的内存，一般用于存放不放在当前方法栈中的那些数据，例如，使用new创建的对象都放在堆里，所以，它不会随方法的结束而消失。方法中的局部变量使用final修饰后，放在堆中，而不是栈中。

### 52、GC是什么? 为什么要有GC?

GC是垃圾收集的意思（Gabage Collection）,内存处理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提供的GC功能可以自动监测对象是否超过作用域从而达到自动回收内存的目的，Java语言没有提供释放已分配内存的显示操作方法。

### 53、垃圾回收器的基本原理是什么？垃圾回收器可以马上回收内存吗？有什么办法主动通知虚拟机进行垃圾回收？

对于GC来说，当程序员创建对象时，GC就开始监控这个对象的地址、大小以及使用情况。通常，GC采用有向图的方式记录和管理堆(heap)中的所有对象。通过这种方式确定哪些对象是"可达的"，哪些对象是"不可达的"。当GC确定一些对象为"不可达"时，GC就有责任回收这些内存空间。可以。程序员可以手动执行System.gc()，通知GC运行，但是Java语言规范并不保证GC一定会执行。

### 54、能不能自己写个类，也叫java.lang.String？

可以，但在应用的时候，需要用自己的类加载器去加载，否则，系统的类加载器永远只是去加载jre.jar包中的那个java.lang.String。

**55、 DOM解析和SAX解析各有什么特点？**

DOM解析构建整个文档驻留内存的树。如果文档很大，就会要求有极大的内存。

SAX是一个解析速度快并且占用内存少的xml解析器，是基于事件驱动来解析，可以随时停止解析不能载入整个文档到内存。

### 二、WEB部分

### 1、HTTP请求的GET与POST方式的区别

get方式以显式提交表单，可以在URL（地址栏）看见我们传的参数，post方式是隐式传值，不可见。现在编程基本上都是在get()方法里面加post()或者在post()方法里面加get()。

### 2、说一说Servlet的生命周期?

servlet有良好的生存期的定义，包括加载和实例化、初始化、处理请求以及服务结束。这个生存期由javax.servlet.Servlet接口的init,service和destroy方法表达。

Servlet被服务器实例化后，容器运行其init方法，请求到达时运行其service方法，service方法自动派遣运行与请求对应的doXXX方法（doGet，doPost）等，当服务器决定将实例销毁的时候调用其destroy方法。

web容器加载servlet，生命周期开始。通过调用servlet的init()方法进行servlet的初始化。通过调用service()方法实现，根据请求的不同调用不同的do\*\*\*()方法。结束服务，web容器调用servlet的destroy()方法。

### 3、SERVLET API中forward() 与redirect()的区别？

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，并且，这样也有助于隐藏实际的链接。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。

### 4、Request对象的主要方法：

setAttribute(String name,Object)：设置名字为name的request的参数值

getAttribute(String name)：返回由name指定的属性值

getCookies()：返回客户端的所有Cookie对象，结果是一个Cookie数组

getCharacterEncoding()：返回请求中的字符编码方式

getParameter(String name)：获得客户端传送给服务器端的有name指定的参数值

### 5、request.getAttribute() 和 request.getParameter() 有何区别?

一个是获取作用域中的值，一个是获取请求参数的值

### 6、 jsp有哪些内置对象?作用分别是什么? 分别有什么方法？

JSP共有以下9个内置的对象：

out 用来传送回应的输出

exception 针对错误网页，未捕捉的例外

request表示HttpServletRequest对象。它包含了有关浏览器请求的信息，并且提供了几个用于获取cookie, header, 和session数据的有用的方法。

response表示HttpServletResponse对象，并提供了几个用于设置送回 浏览器的响应的方法

pageContext它是用于方便存取各种范围的名字空间、servlet相关的对象的API

session表示一个请求的HttpSession对象。Session可以存储用户的状态信息

applicaton 表示一个ServletContext对象。这有助于查找有关servlet引擎和servlet环境的信息

config表示一个javax.servlet.ServletConfig对象。该对象用于存取servlet实例的初始化参数。

page表示从该页面产生的一个servlet实例

### 7、JSP和Servlet有哪些相同点和不同点，他们之间的联系是什么？

JSP是Servlet技术的扩展，本质上是Servlet的简易方式，更强调应用的外表表达。JSP编译后是"类servlet"。Servlet和JSP最主要的不同点在于，Servlet的应用逻辑是在Java文件中，并且完全从表示层中的HTML里分离开来。而JSP的情况是Java和HTML可以组合成一个扩展名为.jsp的文件。JSP侧重于视图，Servlet主要用于控制逻辑。

### 8、MVC的各个部分都有那些技术来实现?如何实现?

MVC是Model－View－Controller的简写。Model 代表的是应用的业务逻辑（通过JavaBean，EJB组件实现）， View 是应用的表示面（由JSP页面产生），Controller 是提供应用的处理过程控制（一般是一个Servlet），通过这种设计模型把应用逻辑，处理过程和显示逻辑分成不同的组件实现。这些组件可以进行交互和重用。

### 9、Web服务优化

1) .尽量减少HTTP请求

两种：{

A. 合并脚本跟样式文件，如果可以把多个css文件组成一个，把多个js文件组成一个。

B. CSS Sprites利用CSS background相关元素进行背景绝对定位，把多个图片合成一个图片。

}

2).使用浏览器缓存

在用户浏览网站的不同页面时，有很多内容是重复的，比如JS,CSS，图片等，如果能建议或强制浏览器在本地缓存这些文件，将大大的降低了页面产生的流量，从而降低页面的载入时间。

1. 服务器端告诉浏览器不要缓存此文件，每次到服务器更新文件。

2. 服务器端没有给浏览器任何指示

3. 服务器端要求缓存此文件，并设置了过期时间，在缓存未到期前，浏览器直接使用本地缓存文件，不会与服务端产生任何通信，我们要做的就是尽量强制浏览器到第四种状态，特别JS,CSS，图片等变动较少的文件。

3) .使用压缩组件

IE和Firefox浏览器都支持客户端GZIP，传输之前，先使用GZIP压缩再传输给客户端，客户端接收之后由浏览器解压，这样虽然稍微占用了一些服务器和客户端的CPU，但是换来的是更高的带宽利用率。对于纯文本来讲，压缩率是相当可观的。如果每个用户节约50%的带宽，那么租用来的那点带宽就可以服务多一倍的客户，并且缩短了数据的传输时间。

4）.图片，JS的预载入，可以在登录页面预载入JS和图片

5）.将脚本放在底部

A. 使用脚本时，对于位于脚本以下的内容，逐步呈现将被阻塞。

B. 在下载脚本时会阻塞并行下载

放在底部可能出现JS的问题，当脚本没加载进来，用户就触发脚本事件。

6）.将样式文件放在页面顶部

7）.使用外部的css和js

8）.精简js去除不必要的，精简css

### 三、jQuery部分

### 1、你在公司是怎么用jquery的？

在项目中是怎么用的是看看你有没有项目经验(根据自己的实际情况来回答) 你用过的选择器啊，复选框啊，表单啊，ajax啊，事件等配置Jquery环境

下载jquery类库在页面引用jquery类库即可

<script type="text/javascript" src="jquery/jquery-1.7.2.min.js"/>

接下来通过在

<script>

$(function(){代码});

</script>

### 2、你为什么要使用jquery？

1.因为jQuery是轻量级的框架，大小不到30kb,它有强大的选择器，

2.出色的DOM操作的封装，有可靠的事件处理机制(jQuery在处理事件绑定的时候相当 的可靠)

3.完善的ajax(它的ajax封装的非常的好，不需要考虑复杂浏览器的兼容性和 XMLHttpRequest对象的创建和使用的问题。) 出色的浏览器的兼容性。

4.而且支持链式操作，隐式迭代。

5.行为层和结构层的分离，还支持丰富的插件，jquery的文档也非常的丰富。

### 3、 $(this) 和 this 关键字在 jQuery 中有何不同？

$(this) 返回一个 jQuery 对象，你可以对它调用多个 jQuery 方法，比如用 text() 获取文本，用val() 获取值等等。而 this 代表当前元素，它是 JavaScript 关键词中的一个，表示上下文中的当前 DOM 元素。你不能对它调用 jQuery 方法，直到它被 $() 函数包裹，例如 $(this)

### 4、你知道jquery中的选择器吗，请讲一下有哪些选择器？

jQuery中的选择器大致分为:基本选择器，层次选择器，过滤选择器，表单选择器

### 5、 jquery对象和dom对象是怎样转换的？

答 ：1.Jquery转DOM对象:jQuery 对象是一个数组对象，可以通过[index]的丰富得到相应的DOM对象还可以通过get[index]去得到相应的DOM对象。

2.DOM对象转jQuery对象:$(DOM对象)

### 6、你是如何使用jquery中的ajax的？

如果是一些常规的ajax程序的话，使用load(),$.get(),$.post(),就可以搞定了，一般我会使用的是$.post() 方法。

如果需要设定beforeSend(提交前回调函数),error(失败后处理),success(成功后处理)及 complete(请求完成后处理)回调函数等，这个时候我会使用$.ajax()

### 7、jquery中$.get()提交和$.post()提交有区别吗？

1 $.get() 方法使用GET方法来进行异步请求的。

$.post() 方法使用POST方法来进行异步请求的。

2 get请求会将参数跟在URL后进行传递，而POST请求则是作为HTTP消息的实体

内容发送给Web服务器的，这种传递是对用户不可见的。

3 get方式传输的数据大小不能超过2KB 而POST要大的多

4 GET 方式请求的数据会被浏览器缓存起来，因此有安全问题。

### 8、jquery中的load方法一般怎么用的？

load方法一般在 载入远程HTML 代码并插入到DOM中的时候用

通常用来从Web服务器上获取静态的数据文件。

如果要传递参数的话，可以使用$.get() 或 $.post()

### 9、在jquery中你是如何去操作样式的？

addClass() 来追加样式

removeClass() 来删除样式

toggle() 来切换样式

### 10、简单的讲叙一下jquery是怎么处理事件的，你用过哪些事件？

首先去装载文档，在页面家在完毕后，浏览器会通过javascript 为DOM元素添加事件。

### 11、你使用过jquery中的动画吗，是怎样用的？

使用过。

hide() 和 show() 同时修改多个样式属性。像高度，宽度，不透明度。

fadeIn() 和fadeOut() fadeTo() 只改变不透明度

slideUp() 和 slideDown() slideToggle() 只改变高度

animate() 属于自定义动画的方法.

### 12、你使用过jquery中的插件吗？

比如分页插件(jQuery Pagination)，验证插件(jQuery formValidator表单验证插件)

### 13、你在jquery中使用过哪些插入节点的方法，它们的区别是什么？

append(),appendTo(),prepend(),prependTo(),after(),insertAfter()

before(),insertBefore()

大致可以分为 内部追加和外部追加

append() 表式向每个元素内部追加内容。

appendTo() 表示将所有的元素追加到指定的元素中。

例$(A)appendTo(B) 是将A追加到B中

下面的方法解释类似

### 14、jquery中如何来获取或和设置属性？

jQuery中可以用attr()方法来获取和设置元素属性

removeAttr() 方法来删除元素属性

### 15、如何来设置和获取HTML 和文本的值？

html()方法 类似于innerHTML属性 可以用来读取或者设置某个元素中的HTML内容

注意：html() 可以用于xhtml文档 不能用于xml文档

text() 类似于innerText属性 可以用来读取或设置某个元素中文本内容。

val() 可以用来设置和获取元素的值

### 16、你jquery中有哪些方法可以遍历节点？

children() 取得匹配元素的子元素集合,只考虑子元素不考虑后代元素

next() 取得匹配元素后面紧邻的同辈元素

prev() 取得匹配元素前面紧邻的同辈元素

siblings() 取得匹配元素前后的所有同辈元素

closest() 取得最近的匹配元素

find() 取得匹配元素中的元素集合 包括子代和后代

## 四、数据库部分

### 1、用两种方法查询部门信息，并将部门名称翻译成中文（scott用户）。

a、SELECT deptno,dname,CASE deptno

WHEN 10 THEN '财务部'

WHEN 20 THEN '研发部'

WHEN 30 THEN '销售部'

ELSE '行政部' END 部门 FROM dept;

b、SELECT deptno,dname,DECODE(deptno,10,'财务部',20,'研发部',30,'销售部','行政部') 部门 FROM dept;

### 2、用两种方法查询各个部门中工资高于本部门的平均工资的员工（scott用户）

a、SELECT \* FROM emp,

(SELECT deptno,avg(sal) avgsal FROM emp GROUP BY deptno) T

WHERE emp.deptno=T.deptno AND sal>avgsal;

b、SELECT \* FROM emp e WHERE sal>

(SELECT AVG(sal) FROM emp WHERE deptno=e.deptno)

### 3、存储过程与触发器区别?

a、存储过程：存储在数据库中，编译后永久有效，用户通过指定存储过程的名称并指定参数（如果存储过程具有参数）来执行。

b、触发器：是与表事件相关的特殊存储过程，程序的执行不被程序调用，也不是由程序手动启动，而是由事件触发，以便在操作表时（插入，删除，更新）执行将被激活。

比如，每插入一个帖子，都希望将版面表中的最后发帖时间，帖子总数字段进行同步更新，用触发器做效率就很高。

### 4、数据库三范式是什么?

第一范式（1NF）：字段具有原子性,不可再分。所有关系型数据库系统都满足第一范式。

数据库表中的字段都是单一属性的，不可再分。例如，姓名字段，其中的姓和名必须作为一个整体，无法区分哪部分是姓，哪部分是名，如果要区分出姓和名，必须设计成两个独立的字段。

第二范式（2NF）：第二范式（2NF）是在第一范式（1NF）的基础上建立起来的。

要求数据库表中的每个实例或行必须可以被惟一地区分。通常需要为表加上一个列，以存储各个实例的惟一标识。这个惟一属性列被称为主关键字或主键。

第三范式（3NF）：满足第三范式（3NF）必须先满足第二范式（2NF）。

简而言之，第三范式（3NF）要求一个数据库表中不包含已在其它表中已包含的非主关键字信息。

所以第三范式具有如下特征：  
         1，每一列只有一个值   
         2，每一行都能区分。   
         3，每一个表都不包含其他表已经包含的非主关键字信息。

例如，帖子表中只能出现发帖人的id，而不能出现发帖人的id，还同时出现发帖人姓名，否则，只要出现同一发帖人id的所有记录，它们中的姓名部分都必须严格保持一致，这就是数据冗余。

### 5、说出一些数据库优化方面的经验?

有外键约束会影响插入和删除性能，如果程序能够保证数据的完整性，那在设计数据库时就去掉外键。

表中允许适当冗余，譬如，主题帖的回复数量和最后回复时间等

还有索引和SQL语句写法对查询性能的改进也是值得关注。

SQL优化：

1.对查询进行优化，应尽量避免全表扫描，首先应考虑在 where 及 order by 涉及的列上建立索引。

2.应尽量避免在 where 子句中使用!=或<>操作符，否则将引擎放弃使用索引而进行全表扫描。

3.应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致引擎放弃使用索引而进行全表扫描，如：select id from t where num is null可以在num上设置默认值0，确保表中num列没有null值，然后这样查询：select id from t where num=0

4.应尽量避免在 where 子句中使用 or 来连接条件，否则将导致引擎放弃使用索引而进行全表扫描，如：select id from t where num=10 or num=20 可以这样查询：select id from t where num=10 union all select id from t where num=20

5.下面的查询也将导致全表扫描：select id from t where name like '%abc%'若要提高效率，可以考虑全文检索。

6.in 和 not in 也要慎用，否则会导致全表扫描，如：select id from t where num in(1,2,3)对于连续的数值，能用 between 就不要用 in 了：select id from t where num between 1 and 3

7.如果在 where 子句中使用参数，也会导致全表扫描。因为SQL只有在运行时才会解析局部变量，但优化程序不能将访问计划的选择推迟到运行时；它必须在编译时进行选择。然而，如果在编译时建立访问计划，变量的值还是未知的，因而无法作为索引选择的输入项。如下面语句将进行全表扫描：select id from t where num=@num可以改为强制查询使用索引：select id from t with(index(索引名)) where num=@num

8.应尽量避免在 where 子句中对字段进行表达式操作，这将导致引擎放弃使用索引而进行全表扫描。如：select id from t where num/2=100应改为:select id from t where num=100\*2

9.应尽量避免在where子句中对字段进行函数操作，这将导致引擎放弃使用索引而进行全表扫描。如：select id from t where substring(name,1,3)='abc'--name以abc开头的id

select id from t where datediff(day,createdate,'2005-11-30')=0--'2005-11-30'生成的id应改为:select id from t where name like 'abc%' select id from t where createdate>='2005-11-30' and createdate<'2005-12-1'

10.不要在 where 子句中的“=”左边进行函数、算术运算或其他表达式运算，否则系统将可能无法正确使用索引。

11.在使用索引字段作为条件时，如果该索引是复合索引，那么必须使用到该索引中的第一个字段作为条件时才能保证系统使用该索引，否则该索引将不会被使用，并且应尽可能的让字段顺序与索引顺序相一致。

12.不要写一些没有意义的查询，如需要生成一个空表结构：select col1,col2 into #t from t where 1=0 这类代码不会返回任何结果集，但是会消耗系统资源的，应改成这样：create table #t(...)

13.很多时候用 exists 代替 in 是一个好的选择：select num from a where num in(select num from b)用下面的语句替换：select num from a where exists(select 1 from b where num=a.num)

14.并不是所有索引对查询都有效，SQL是根据表中数据来进行查询优化的，当索引列有大量数据重复时，SQL查询可能不会去利用索引，如一表中有字段sex，male、female几乎各一半，那么即使在sex上建了索引也对查询效率起不了作用。

15.索引并不是越多越好，索引固然可以提高相应的 select 的效率，但同时也降低了 insert 及 update 的效率，因为 insert 或 update 时有可能会重建索引，所以怎样建索引需要慎重考虑，视具体情况而定。一个表的索引数最好不要超过6个，若太多则应考虑一些不常使用到的列上建的索引是否有必要。

16.应尽可能的避免更新 clustered 索引数据列，因为 clustered 索引数据列的顺序就是表记录的物理存储顺序，一旦该列值改变将导致整个表记录的顺序的调整，会耗费相当大的资源。若应用系统需要频繁更新 clustered 索引数据列，那么需要考虑是否应将该索引建为 clustered 索引。

17.尽量使用数字型字段，若只含数值信息的字段尽量不要设计为字符型，这会降低查询和连接的性能，并会增加存储开销。这是因为引擎在处理查询和连接时会逐个比较字符串中每一个字符，而对于数字型而言只需要比较一次就够了。

18.尽可能的均衡使用 varchar/nvarchar 和 char/nchar ，变长字段存储空间小，可以节省存储空间，对于查询来说，固定长度的字段内搜索效率显然要高些。

19.任何地方都不要使用 select \* from t ，用具体的字段列表代替“\*”，不要返回用不到的任何字段。

20.尽量使用表变量来代替临时表。如果表变量包含大量数据，请注意索引非常有限（只有主键索引）。

21.避免频繁创建和删除临时表，以减少系统表资源的消耗。

22.临时表并不是不可使用，适当地使用它们可以使某些例程更有效，例如，当需要重复引用大型表或常用表中的某个数据集时。但是，对于一次性事件，最好使用导出表。

23.在新建临时表时，如果一次性插入数据量很大，那么可以使用 select into 代替 create table，避免造成大量 log ，以提高速度；如果数据量不大，为了缓和系统表的资源，应先create table，然后insert。

24.如果使用到了临时表，在存储过程的最后务必将所有的临时表显式删除，先 truncate table ，然后 drop table ，这样可以避免系统表的较长时间锁定。

25.尽量避免使用游标，因为游标的效率较差，如果游标操作的数据超过1万行，那么就应该考虑改写。

26.使用基于游标的方法或临时表方法之前，应先寻找基于集的解决方案来解决问题，基于集的方法通常更有效。

27.与临时表一样，游标并不是不可使用。对小型数据集使用 FAST\_FORWARD 游标通常要优于其他逐行处理方法，尤其是在必须引用几个表才能获得所需的数据时。在结果集中包括“合计”的例程通常要比使用游标执行的速度快。如果开发时间允许，基于游标的方法和基于集的方法都可以尝试一下，看哪一种方法的效果更好。

28.在所有的存储过程和触发器的开始处设置 SET NOCOUNT ON ，在结束时设置 SET NOCOUNT OFF 。无需在执行存储过程和触发器的每个语句后向客户端发送 DONE\_IN\_PROC 消息。

29.尽量避免向客户端返回大数据量，若数据量过大，应该考虑相应需求是否合理。

30.尽量避免大事务操作，提高系统并发能力。

### 6、union和union all有什么不同?

UNION联合之后会去掉重复值，而UNION ALL则将所有的结果全部显示出来，不管是不是重复。所以UNION ALL 要比UNION快很多，如果可以确认合并的两个结果集中不包含重复的数据的话，那么就使用UNION ALL。

### 7.分页语句

**取出表中id第31到40的记录（以自动增长ID为主键）**

sql server方案：select top 10 \* from t where id not in (select top 30 id from t order by id );

mysql方案：select \* from t order by id limit 30,10;

oracle方案1：select \* from (select rownum r,tb.\* from (select \* from t order by id) tb) where r>30 and r<=40;

oracle方案2：select \* from (select t.\*,row\_number() over(order by id) r FROM t) where r<=40 and r>30;

### 8.用一条SQL语句 查询出每门课都大于80分的学生姓名

name   kecheng   fenshu   
张三     语文       81  
张三     数学       75  
李四     语文       76  
李四     数学       90  
王五     语文       81  
王五     数学       100  
王五     英语       90

答案：  
A、SELECT distinct name FROM score WHERE name NOT IN

(SELECT distinct name FROM score WHERE fenshu<=80);

1. SELECT name FROM score GROUP BY name HAVING MIN(fenshu)>80;
2. SELECT name FROM score PIVOT (sum(fenshu) FOR kecheng IN('语文' m1,'数学' m2,'英语' m3)) WHERE m1>80 AND m2>80 AND m3>80;

### 9.统计每年每月的信息（行列转换）

year  month amount  
1991   1     1.1  
1991   2     1.2  
1991   3     1.3  
1991   4     1.4  
1992   1     2.1  
1992   2     2.2  
1992   3     2.3  
1992   4     2.4  
查成这样一个结果  
year m1  m2  m3  m4  
1991 1.1 1.2 1.3 1.4  
1992 2.1 2.2 2.3 2.4

A、SELECT year,sum(CASE MONTHS WHEN 1 THEN amount END) m1,

sum(CASE MONTHS WHEN 2 THEN amount END) m2,

sum(CASE MONTHS WHEN 3 THEN amount END) m3

FROM amount GROUP BY YEAR;

B、SELECT year,SUM(DECODE(MONTHS,1,amount,0)) m1,

SUM(DECODE(MONTHS,2,amount,0)) m2,

SUM(DECODE(MONTHS,3,amount,0)) m3

FROM amount GROUP BY YEAR;

C、SELECT year,(SELECT amount FROM amount WHERE MONTHS=1 AND YEAR=a.year) m1,

(SELECT amount FROM amount WHERE MONTHS=2 AND YEAR=a.year) m2,

(SELECT amount FROM amount WHERE MONTHS=3 AND YEAR=a.year) m3

FROM amount a GROUP BY YEAR;

D、SELECT \* FROM amount PIVOT(SUM(amount) FOR months IN(1 m1,2 m2,3 m3));

### 10、删除除了id号不同,其他都相同的学生冗余信息

学生表 如下:  
id号   学号   姓名 课程编号 课程名称 分数  
1        2005001  张三  0001      数学    69  
2        2005002  李四  0001      数学    89  
3        2005001  张三  0001      数学    69  
A: delete from tablename where id not in(select min(id) from tablename group by 学号,姓名,课程编号,课程名称,分数)

### 11、一个用户表中有一个积分字段，假如数据库中有100多万个用户，若要在每年第一天凌晨将积分清零，你将考虑什么，你将想什么办法解决?

alter table drop column score;

alter table add colunm score int;

这样的操作是无法回滚的，只有inert update delete等DML语句才能回滚，

对于create table,drop table ,alter table等DDL语句是不能回滚。

解决方案一，update user set score=0;

解决方案二，假设上面的代码要执行好长时间，超出我们的容忍范围，那我就alter table user drop column score;alter table user add column score int。

### 12. sql面试题

Table **EMPLOYEES** Structure:

EMPLOYEE\_ID NUMBER Primary Key,

FIRST\_NAME VARCHAR2(25),

LAST\_NAME VARCHAR2(25),

Salary NUMBER(8,2),

HiredDate DATE,

Departmentid NUMBER(2)

Table **Departments** Structure:

Departmentid NUMBER(2) Primary Key,

DepartmentName VARCHAR2(25).

1. 基于上述EMPLOYEES表写出查询：写出雇用日期在今年的，或者工资在[1000,2000]之间的，或者员工姓名（last\_name）以’Obama’打头的所有员工，列出这些员工的全部个人信息。

select \* from employees where Year(hiredDate) = Year(date()) or (salary between 1000 and 200)

or left(last\_name,5)='Obama';

(2) 基于上述EMPLOYEES表写出查询：查出部门平均工资大于1800元的部门的所有员工，列出这些员工的全部个人信息。

select id,name,salary,departmentid did from employees where (select avg(salary)

from employees where departmentid = did) > 1800;

(3) 基于上述EMPLOYEES表写出查询：查出个人工资高于其所在部门平均工资的员工，列出这些员工的全部个人信息及该员工工资高出部门平均工资百分比。

select employees.\*,(employees.salary-t.avgSalary)\*100/employees.salary

from employees,(select departmentid,avg(salary) avgSalary from employees group by departmentid) as t

where employees.departmentid = t.departmentid and employees.salary>t.avgSalary;

### 13、行列转换

create table demo

(

country varchar2(20),*--国家*

sex number,*--性别,男=1,女=2*

pop number*--人口数量*

);

*--表数据如下：*

insert into demo values('中国',1,340);

insert into demo values('中国',2,240);

insert into demo values('美国',1,45);

insert into demo values('美国',2,55);

insert into demo values('加拿大',1,40);

insert into demo values('加拿大',2,65);

insert into demo values('英国',1,34);

insert into demo values('英国',2,60);

commit;

*--编写查询实现如下效果：*

*--效果1：*

（行列转换）

CASE-WHEN查询、DECODE()、相关子查询、Pivot四种方式实现

Pivot语法：select \* from 表 pivot（聚合函数 for 列名 in（类型））

**

--case when

select country 国家,

sum(case when sex=1 then pop end) 男,sum(case when sex=2 then pop end) 女 from demo

group by country

--decode

select country 国家,sum(decode(sex,1,pop)) 男, sum(decode(sex,2,pop)) 女 from demo

group by country

--子查询

select country 国家,

(select pop from demo where sex=1 and country=d.country) 男,

(select pop from demo where sex=2 and country=d.country) 女 from demo d

group by country

--11G|sql2005新增 行转列pivot 列转行unpivot

--pivot（聚合函数for列名 in（类型）），其中in('')中可以指定别名，in中还可以指定子查询

SELECT \* FROM demo Pivot(SUM (POP) FOR sex IN(1 男,2 女))

### **14、 Oracle函数和存储过程有什么区别？**

1.返回值的区别,函数有1个返回值,而存储过程是通过参数返回的,可以有多个或者没有

2.调用的区别,函数可以在查询语句中直接调用,而存储过程必须单独调用.

函数一般情况下是用来计算并返回一个计算结果而存储过程一般是用来完成特定的数据操作（比如修改、插入数据库表或执行某些DDL语句等等）

### 15、什么是索引以及索引的优缺点

1) 索引是对数据库表中一或多个列的值进行排序的结构，是帮助MySQL高效获取数据的数据结构。

2) 索引的几个基本类型：普通索引、唯一索引、主键索引、全文索引。

3) 索引使数据库程序无需扫描整个表，就可以在其中找到所需要的数据，索引包含了一个表中包含值的列表，其中包含了各个值的行所存储的位置，索引可以是单个或一组列，索引提供的表中数据的逻辑位置，合理划分索引能够大大提高数据库性能。

a) 通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性。

b) 可以大大加快数据的检索速度，这也是创建索引的最主要的原因。

c) 通过使用索引，可以在查询的过程中使用优化隐藏器，提高系统的性能

4) 缺点

a) 创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加。

b) 索引需要占物理空间，除了数据表占数据空间之外，每一个索引还要占一定的物理空间，如果要建立聚簇索引，那么需要的空间就会更大。

c) 索引降低了插入、删除、修改等维护任务的速度

### 16、什么是存储过程？它有什么优点？

1) 存储过程是一组预编译的SQL语句。

2) 优点：

a) 允许模块化程序设计，就是说只需要创建一次过程，以后在程序中就可以调用该过程任意次。

b) 允许更快执行，如果某操作需要执行大量SQL语句或重复执行，存储过程比SQL语句执行的要快。

c) 减少网络流量，例如一个需要数百行的SQL代码的操作有一条执行语句完成，不需要在网络中发送数百行代码。

d) 更好的安全机制，对于没有权限执行存储过程的用户，也可授权他们执行存储过程。

### 17、数据库事务概念

1) 事务是指一个工作单元，它包含了一组数据操作命令，并且所有的命令作为一个整体一起向系统提交或撤消请求操作，即这组命令要么都执行，要么都不执行。

原子性(Atomicity)、一致性(Consistency)、隔离性(Isolation)、持久性(Durability)。

a) 原子性:事务必须是原子工作单元；对于其数据修改，要么全都执行，要么全都不执行。

b) 一致性:事务在完成时，必须使所有的数据都保持一致状态。在相关数据库中，所有规则都必须应用于事务的修改，以保持所有数据的完整性。事务结束时，所有的内部数据结构（如 B 树索引或双向链表）都必须是正确的。

c) 隔离性:由并发事务所作的修改必须与任何其它并发事务所作的修改隔离。

d) 持久性:事务完成之后，它对于系统的影响是永久性的。该修改即使出现系统故障也将一直保持。

### 18、对乐观锁、悲观锁的理解

1) 数据库管理系统（DBMS）中的并发控制的任务是确保在多个事务同时存取数据库中同一数据时不破坏事务的隔离性和统一性以及数据库的统一性。

2) 乐观并发控制(乐观锁)和悲观并发控制（悲观锁）是并发控制主要采用的技术手段。

a) 悲观锁：假定会发生并发冲突，屏蔽一切可能违反数据完整性的操作（依赖于数据库的锁）select ... for update

b) 乐观锁：假设不会发生并发冲突，只在提交操作时检查是否违反数据完整性。

1.实体类中增加version属性(数据库也对应生成该字段)

2.Hibernate提供了其他实现乐观锁的办法。把<class>元素的optimistic-lock属性设为“all”：

<class name="Account" table="ACCOUNTS" optimistic-lock="all" dynamic-update="true">

### 19、Mysql，Oracle，SqlServer的区别

1) Mysql和Oracle同属于Oracle旗下公司。SQL Server 是Microsoft 公司旗下。都是属于关系型的数据库，都是目前的主流数据库，其基本操作语法大致相同。

2) Mysql安装配置简单，性能优越，开源免费。且适应于所有的平台，为中小型数据库。而Sqlserver是中大型数据库，且稳定。

3) Mysql有着运行速度快，使用成本低，可移植性强，适用用户广等优点。但是默认引擎是MyISAM不支持事务管理的。

4） Oracle是主流的大型数据库，大多数电信项目都是使用的Oracle，而Sqlserver与Mysql主要是个人以及中小型公司使用的的数据库，但是Oracle和Sqlserver需要收费，Mysql不用。

5) 如果按功能上来说，Oracle最为强大，Oracle支持递归查询，二后两者不支持；

### 20、SQL复制表语句（只复制表结构，源表名a，新表名b）

Create table b as select \* from a where 1=0;

### 21、删除数据表中重复记录，只保留一条，如将用户表users(userId,username)中相同username的记录保留一条。

delete from users where userId not in (select \* from (select max(userId) from users group by username having count(username)>1) as b);

## 五、SSM

### 1、请描述下Springmvc有哪些核心组件？

核心控制器：Dispatcher Servlet

控制器：Controller请求分发器：Handler Mapping视图解析器:ViewResolver

拦截器：Interceptors国际化:LocalResolver

数据验证:Validate

### 2、请描述下Springmvc的工作流程？

---请求处理流程:

DispatcherServlet --> Handler Mapping ---> Controller ---> ModelAndView

--->ViewResolver--->View

### 3、请描述下DispatcherServlet的作用？

DispatcherServlet是前端控制器设计模式的实现，提供Spring Web MVC的集中访问点，负责职责的分派，且与Spring IoC容器无缝集成，从而可以获得Spring的所有好处。

DispatcherServlet主要用作职责调度，用于控制流程，主要职责如下：

1、文件上传解析，如果请求类型是multipart将通过MultipartResolver进行文件上传解析；

2、通过HandlerMapping，将请求映射到处理器；

3、通过ViewResolver解析逻辑视图名到具体视图实现；

4、本地化国际化解析；

5、渲染具体的视图；

6、如果执行过程中遇到异常将交给HandlerExceptionResolver来解析。

### 4、你对Spring的理解。

1.Spring实现了工厂模式的工厂类，这个类名为BeanFactory（实际上是一个接口），在程序中通常BeanFactory的子类ApplicationContext。Spring相当于一个大的工厂类，在其配置文件中通过<bean>元素配置用于创建实例对象的类名和实例对象的属性。

2. Spring提供了对IOC良好支持，IOC是一种编程思想，利用这种思想可以很好地实现模块之间的解耦。IOC也称为DI（Depency Injection），控制对象产生反转到Spring或者说对象依赖Spring运行时注入系统。

3. Spring提供了对AOP技术的良好封装， AOP称为面向切面编程，就是系统中有很多各不相干的类的方法，在这些众多方法中要加入某种系统功能的代码，例如，加入日志，加入权限判断，加入异常处理，这种应用称为AOP。实现AOP功能采用的是代理技术，客户端程序不再调用目标，而调用代理类，代理类与目标类对外具有相同的方法声明，低层实现依赖JDK中Proxy类产生动态代理的方式为某个接口生成实现类，如果要为某个类生成子类，则会自动使用CGLI B。

### 5、mybatis中#和$符号的区别

#{}实现的是向prepareStatement中的预处理语句设置参数值，sql语句中#{}表示一个占位符，即‘?’。

使用占位符#{}可以有效防止sql注入，在使用时不需要关心参数值的类型，mybatis会自动进行java类型和jdbc类型的转换。#{}可以接收简单类型值或pojo属性值。

${}和#{}不同，通过${}可以将parameterType传入的内容拼接在sql中且不进行jdbc类型转换，使用${}不能防止sql注入，但是有时必须用${}，如order by排序，如果将列名和关键字通过参数传入sql，根据传的列名进行排序，应该写为：ORDER BY ${columnName} ${desc}，如果使用#{}将无法实现此功能。

### 6、为什么说 Mybatis 是半自动 ORM 映射工具？它与全自动的区别在哪里？

Hibernate 属于全自动 ORM 映射工具，使用 Hibernate 查询关联对象或者关联集合对象时，可以根据对象关系模型直接获取，所以它是全自动的。而 Mybatis 在查询关联对象或关联集合对象时，需要手动编写 sql 来完成，Mybatis 即使操作一个单表,也要自己写sql语句,所以，称之为半自动 ORM 映射工具。

### Mybatis 是否支持延迟加载？如果支持，它的实现原理是什么？

1. Mybatis 仅支持 association 关联对象和 collection 关联集合对象的延迟加载， association指的就是一对一， collection 指的就是一对多查询。在Mybatis配置文件中，可以配置是否启用延迟加载。
2. 他的原理是，使用CGLIB创建目标对象的代理对象，当调用目标方法时，进入拦截器方法，比如调用a.getB().getName()，拦截器invoke()方法发现a.getB()是null值，那么就会单独发送保存好的查询关联B对象的SQL，把B查询上来，然后调用a.setB(b)，于是a的对象b属性就有值了，接着完成a.getB().getName()方法的调用。这就是延迟加载的基本原理。

### MyBatis 的好处是什么？

1） MyBatis 把 sql 语句从 Java 源程序中独立出来，放在单独的 XML 文件中编写，给程序的维护带来了很大便利。  
2） MyBatis 封装了底层 JDBC API 的调用细节，并能自动将结果集转换成 Java Bean 对象，大大简化了 Java 数据库编程的重复工作。  
3）因为 MyBatis 需要程序员自己去编写 sql 语句，程序员可以结合数据库自身的特点灵活控制 sql 语句，因此能够实现比 Hibernate 等全自动 orm 框架更高的查询效率，能够完成复杂查询。

### 9、什么是 MyBatis 的接口绑定,有什么好处？

接口映射就是在Mybatis中任意定义接口，然后把接口里面的方法和sql语句绑定，我们直接调用接口方法就可以，这样比原来的SqlSession提高的方法我们可以有更加灵活的选择和设置。

### 10、接口绑定有几种实现方式,分别是怎么实现的?

接口绑定有两种实现方式，一种是通过注解绑定，就是子啊接口的方法上面加@select，@update等注解里面包含Sql语句来绑定；另外一种就是通过xml里面写sql来绑定，在这种情况下，要指定xml文件里面的namespace必须为接口的全路径名字。

当 Sql 语句比较简单时候,用注解绑定；当 SQL 语句比较复杂时候,用 xml 绑定,一般用xml 绑定的比较多。

### Spring AOP和AspectJ AOP有什么区别？

Spring AOP是属于运行时增强，而AspectJ是编译时增强。Spring AOP基于代理（Proxying），而AspectJ基于字节码操作（Bytecode Manipulation）。

Spring AOP已经集成了AspectJ，AspectJ应该算得上是Java生态系统中最完整的AOP框架了。AspectJ相比于Spring AOP功能更加强大，但是Spring AOP相对来说更简单。

如果我们的切面比较少，那么两者性能差异不大。但是，当切面太多的话，最好选择AspectJ，它比SpringAOP快很多。

### 12、将一个类声明为Spring的bean的注解有哪些？

我们一般使用@Autowired注解去自动装配bean。而想要把一个类标识为可以用@Autowired注解自动装配的bean，可以采用以下的注解实现：

1.@Component注解。通用的注解，可标注任意类为Spring组件。如果一个Bean不知道属于哪一个层，可以使用@Component注解标注。

2.@Repository注解。对应持久层，即Dao层，主要用于数据库相关操作。

3.@Service注解。对应服务层，即Service层，主要涉及一些复杂的逻辑，需要用到Dao层（注入）。

4.@Controller注解。对应Spring MVC的控制层，即Controller层，主要用于接受用户请求并调用Service层的方法返回数据给前端页面。

### 13、@Component和@Bean的区别是什么？

1.作用对象不同。@Component注解作用于类，而@Bean注解作用于方法。

2.@Component注解通常是通过类路径扫描来自动侦测以及自动装配到Spring容器中（我们可以使用@ComponentScan注解定义要扫描的路径）。@Bean注解通常是在标有该注解的方法中定义产生这个bean，告诉Spring这是某个类的实例，当我需要用它的时候还给我。

3.@Bean注解比@Component注解的自定义性更强，而且很多地方只能通过@Bean注解来注册bean。比如当引用第三方库的类需要装配到Spring容器的时候，就只能通过@Bean注解来实现。

### 14、Spring框架中用到了哪些设计模式？

1.工厂设计模式：Spring使用工厂模式通过BeanFactory和ApplicationContext创建bean对象。

2.代理设计模式：Spring AOP功能的实现。

3.单例设计模式：Spring中的bean默认都是单例的。

4.模板方法模式：Spring中的jdbcTemplate、hibernateTemplate等以Template结尾的对数据库操作的类，它们就使用到了模板模式。

5.包装器设计模式：我们的项目需要连接多个数据库，而且不同的客户在每次访问中根据需要会去访问不同的数据库。这种模式让我们可以根据客户的需求能够动态切换不同的数据源。

6.观察者模式：Spring事件驱动模型就是观察者模式很经典的一个应用。

7.适配器模式：Spring AOP的增强或通知（Advice）使用到了适配器模式、Spring MVC中也是用到了适配器模式适配Controller。

### 15、什么是MVC?

MVC是一个架构，或者说是一个设计模式，它就是强制性使应用程序的输入，处理和输出分开。将一个应用程序分为三个部分：Model，View，Controller。

Model 模型，完成业务逻辑：由javaBean构成，在MVC的三个部件中，模型拥有最多的处理任务。由于应用于模型的代码只需写一次就可以被多个视图重用，所以减少了代码的重复性。

View 视图，就是负责跟用户交互的界面。一般就是由HTML，css元素组成的界面，当然现在还有一些像js，ajax，flex一些也都属于视图层。 在视图层里没有真正的处理发生，只负责数据输入输出。

Controller 控制器，接收请求—>调用模型—>根据结果派发页面并经过模型处理返回相应数据 。

**优点：分工明确，松耦合，复用性高，有利于标准化。**

### 16、Spring MVC的工作原理

流程说明：

1.客户端（浏览器）发送请求，直接请求到DispatcherServlet。

2.DispatcherServlet根据请求信息调用HandlerMapping，解析请求对应的Handler。

3.解析到对应的Handler（也就是我们平常说的Controller控制器）。

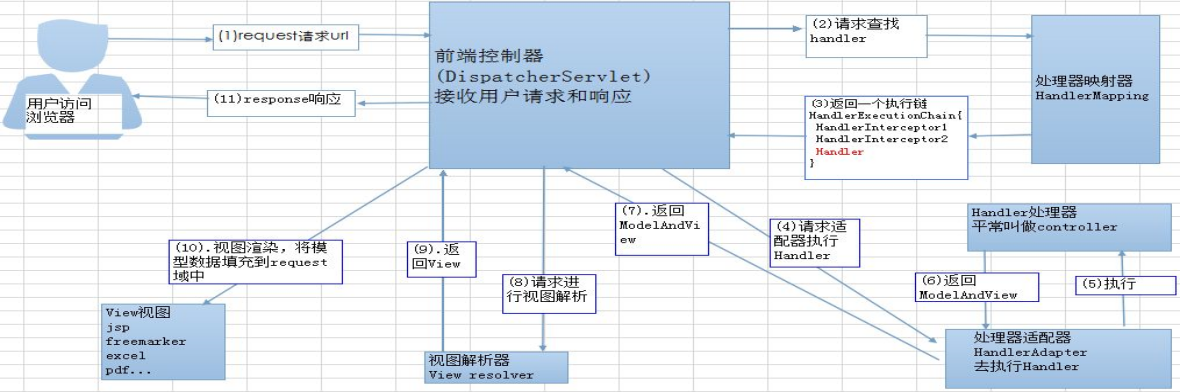
4.HandlerAdapter会根据Handler来调用真正的处理器来处理请求和执行相对应的业务逻辑。

5.处理器处理完业务后，会返回一个ModelAndView对象，Model是返回的数据对象，View是逻辑上的View。

6.ViewResolver会根据逻辑View去查找实际的View。

7.DispatcherServlet把返回的Model传给View（视图渲染）。

8.把View返回给请求者（浏览器）。



### 六、SpringBoot

### 1、springboot是什么？有什么优缺点？

1. Springboot的简介

1) 本身并不提供Spring框架的核心特性以及扩展功能，只是用于快速、敏捷地开发新一代基于Spring框架的应用程序。大部分的Spring Boot应用都只需要非常少量的配置代码，开发者能够更加专注于业务逻辑。Spring Boot 让日益臃肿的 Java 代码又重回简洁。在配合 Spring Cloud 使用时，还可以发挥更大的威力。

2. Springboot的优缺点

1) 优点：注解代替xml配置

1，纯JAVA的配置方式，很简单，很方便

2，配合各种starter使用，基本上可以做到自动化配置

3，配合Maven或Gradle等构件工具打成Jar包后，Java -jar 进行部署运行很简单。

2) 缺点

1，从原来的xml配置方式转换到JAVA配置方式变化有点大，可能一下子转换不过来

2，Spring Boot 比较适合做微服务，不适合做比较大型的项目。

### 2、springboot 的作用？

用来简化spring应用的初始搭建以及开发过程 使用特定的方式来进行配置（properties或yml文件）

创建独立的spring应用程序 main方法运行

嵌入的Tomcat 无需部署war文件

简化maven的配置

自动配置spring添加对应功能starter自动化配置

### 3、springboot常用的starter有哪些？

spring-boot-starter-web 嵌入tomcat和web开发需要的servlet等支持

mybatis-spring-boot-starter 第三方的mybatis集成starter

spring-boot-starter-data-jpa 数据库支持

spring-boot-starter-data-redis redis数据库支持

spring-boot-starter-data-mongodb mongodb数据库支持

spring-boot-starter-thymeleaf thymeleaf模板引擎支持

### 4、springboot自动配置的原理

在spring程序main方法中 添加@SpringBootApplication或者@EnableAutoConfiguration会自动去maven中读取每个starter中的spring.factories文件，该文件里配置了所有需要被创建spring容器中的bean

### 5、springboot读取配置文件的方式？是否可以使用 XML 配置 ?

### springboot默认读取配置文件为application.properties或者是application.yml

springboot推荐使用 Java 配置而非 XML 配置，但是 Spring Boot 中也可以使用 XML 配置，通过 @ImportResource 注解可以引入一个 XML 配置。

### 6、YAML 配置的优势在哪里 ?

YAML 现在可以算是非常流行的一种配置文件格式了，无论是前端还是后端，都可以见到 YAML 配置。那么 YAML 配置和传统的 properties 配置相比到底有哪些优势呢？

配置有序，在一些特殊的场景下，配置有序很关键

支持数组，数组中的元素可以是基本数据类型也可以是对象

简洁，相比 properties 配置文件，YAML 还有一个缺点，就是不支持 @PropertySource 注解导入自定义的 YAML 配置。

### 7、Spring Boot 打成的 jar 和普通的 jar 有什么区别 ?

Spring Boot 项目最终打包成的 jar 是可执行 jar ，这种 jar 可以直接通过 java -jar xxx.jar 命令来运行，这种 jar 不可以作为普通的 jar 被其他项目依赖，即使依赖了也无法使用其中的类。

### 8、Spring Boot 中如何解决跨域问题 ?

跨域可以在前端通过 JSONP 来解决，但是 JSONP 只可以发送 GET 请求，无法发送其他类型的请求，在RESTful 风格的应用中，就显得非常鸡肋，因此我们推荐在后端通过 （CORS，Cross-origin resource sharing） 来解决跨域问题。这种解决方案并非 Spring Boot 特有的，在传统的 SSM 框架中，就可以通过 CORS 来解决跨域问题，只不过之前我们是在 XML 文件中配置 CORS ，现在则是通过 @CrossOrigin 注解来解决跨域问题。@CrossOrigin(origins = "\*",maxAge = 3600)

### 9、什么是Thymeleaf？

### Thymeleaf是一个现代服务器端Java模板引擎，适用于Web和独立环境，能够处理HTML，XML，JavaScript，CSS甚至纯文本。

### Thymeleaf的主要目标是提供一种优雅且高度可维护的模板创建方式。为实现这一目标，它以自然模板的概念为基础，将其逻辑注入模板文件，其方式不会影响模板被用作设计原型。这改善了设计沟通，缩小了设计和开发团队之间的差距。

### Thymeleaf也从一开始就设计了Web标准，特别是HTML5。

### 10、Spring Boot如何添加MVC配置？SpringBoot如何使用监听器、过滤器？

实现接口WebMvcConfigurer，选择性重写要添加的mvc配置，如添加拦截器方法addInterceptors()。

[实现过滤器或监听器接口后使用@ServletComponentScan("com.cssl.web")扫描到容器。](mailto:实现过滤器或监听器接口后使用@ServletComponentScan(\"com.cssl.web\")扫描到容器。)

### 七、NoSQL(not only sql)

### 1、Redis是什么？

Redis是Remote Dictionary Server缩写，它是一个开源的，高性能的键-值存储。Redis出现补偿了关系型数据存在的多种不足。

Redis支持存储多种数据类型：String，list，set，zset，hash，这些类型都支持新增/修改/删除。

Redis是将数据存放到内存中，由于内容存取速度快所以redis被广泛应用在互联网项目中。

Redis持久化：两种持久化方式

①.定时快照方式(RDB)(默认)

②.实时保存，基于语句追加文件的方式（AOF）

注意点：

并不是所有数据都要放进Redis缓存，而是针对一些改动性不大或者访问率大的数据进行缓存来减少关系型数据的压力。

### 2、Redis主从复制：

通过持久化功能，Redis保证了即使在服务器重启的情况下也不会损失（或少量损失）数据，因为持久化会把内存中数据保存到硬盘上，重启会从硬盘上加载数据。

但是由于数据是存储在一台服务器上的，如果这台服务器出现硬盘故障等问题，也会导致数据丢失。为了避免单点故障，通常的做法是将数据库复制多个副本以部署在不同的服务器上，这样即使有一台服务器出现故障，其他服务器依然可以继续提供服务。

在复制的概念中，数据库分为两类，一类是主数据库（master），另一类是从数据库（slave）。主数据库可以进行读写操作，当写操作导致数据变化时会自动将数据同步给从数据库。而从数据库一般是只读的，并接受主数据库同步过来的数据。一个主数据库可以拥有多个从数据库，而一个从数据库只能拥有一个主数据库。

### 3、Redis优点和缺点：

优点：

1）免费开源

2）易于安装使用，操作简单

3）数据结构丰富灵活，适用场景广泛

4）具有事务性，保持一致性

缺点：

1）持久化不成熟，快照文件必然面临数据丢失，AOF很影响性能

2）存储成本高，纯内存操作，不适合海量数据

3）架构扩展不是很简单

### 4、Redis分布式锁问题

Redis在应用集群状态下可以使用分布式锁保障数据的一致性，即使用setnx命令的方式加锁。不过redis也做了集群(主从复制)的情况下也将不能保障数据一致，如：这个master节点由于某些原因发生了主从切换，那么就会出现锁丢失的情况(在Redis的master节点上拿到了锁；但是这个加锁的key还没有同步到slave节点；master故障，发生故障转移，slave节点升级为master节点；导致锁丢失)，正因为如此，Redis基于分布式环境下提出了一种更高级的分布式锁的实现方式：**Redlock(了解)**

### 5、Redis雪崩的问题

目前电商首页以及热点数据都会去做缓存，一般缓存都是定时任务去刷新，或者查不到之后去更新缓存的，定时任务刷新就有一个问题。举个栗子：如果首页所有Key的失效时间都是12小时，中午12点刷新的，我零点有个大促活动大量用户涌入，假设每秒6000个请求，本来缓存可以抗住每秒5000个请求，但是缓存中所有Key都失效了。此时6000个/秒的请求全部落在了数据库上，数据库必然扛不住，真实情况可能DBA都没反应过来直接挂了，此时，如果没什么特别的方案来处理，DBA很着急，重启数据库，但是数据库立马又被新流量给打死了。这就是缓存雪崩。

处理缓存雪崩简单，在批量往Redis存数据的时候，把每个Key的失效时间都加个随机值就好了，这样可以保证数据不会再同一时间大面积失效。

set（key, value, time+Math.random()\*10000）;

如果Redis是集群部署，将热点数据均匀分布在不同的Redis库中也能避免全部失效。或者设置热点数据永不过期，有更新操作就更新缓存就好了，电商首页的数据也可以用这个操作，保险。

### 6、Redis缓存穿透和缓存击穿，和雪崩有什么区别，怎么解决？

先说下缓存穿透吧，缓存穿透是指缓存和数据库中都没有的数据，而用户（黑客）不断发起请求，举个栗子：我们数据库的id都是从1自增的，如果发起id=-1的数据或者id特别大不存在的数据，这样的不断攻击导致数据库压力很大，严重会击垮数据库。

至于缓存击穿，这个跟缓存雪崩有点像，但是又有一点不一样，缓存雪崩是因为大面积的缓存失效，打崩了DB，而缓存击穿不同的是缓存击穿是指一个Key非常热点，在不停地扛着大量的请求，大并发集中对这一个点进行访问，当这个Key在失效的瞬间，持续的大并发直接落到了数据库上，就在这个Key的点上击穿了缓存。

缓存穿透可以在接口层增加校验，比如用户鉴权，参数做校验，不合法的校验直接return，比如id做基础校验，id<=0直接拦截。Redis里还有一个高级用法**布隆过滤器（Bloom Filter）**这个也能很好的预防缓存穿透的发生，他的原理也很简单，就是利用高效的数据结构和算法快速判断出你这个Key是否在数据库中存在，不存在你return就好了，存在你就去查DB刷新KV再return。缓存击穿的话，设置热点数据永不过期，或者加上互斥锁就搞定了。

### 7、Redis是单线程的为什么还那么快？

第一：Redis完全基于内存，绝大部分请求是纯粹的内存操作，非常迅速，数据存在内存中，类似于HashMap，HashMap的优势就是查找和操作的时间复杂度是O(1)。

第二：数据结构简单，对数据操作也简单。

第三：采用单线程，避免了不必要的上下文切换和竞争条件，不存在多线程导致的CPU切换，不用去考虑各种锁的问题，不存在加锁释放锁操作，没有死锁问题导致的性能消耗。

第四：使用多路复用IO模型，非阻塞IO。

### 8、Redis主从复制和哨兵模式是什么？

redis单节点存在单点故障问题，为了解决单点问题，一般都需要对redis配置主从节点，然后使用哨兵来监听主节点的存活状态，如果主节点挂掉，从节点能继续提供缓存功能

主从配置结合哨兵模式能解决单点故障问题，提高redis可用性。从节点仅提供读操作，主节点提供写操作。对于读多写少的状况，可给主节点配置多个从节点，从而提高响应效率。

### 9、Mongodb跟redis的区别：

Mongodb更类似于mysql关系型数据库

A. 擅长查询json数据，数据结构单一，不支持事务

B. MongoDB集群技术比较成熟，只将热点数据存内存

C. 支持丰富的数据表达，索引，更类似于关系型数据库，支持查询的语言比较丰富。

redis非关系型数据库

A. 从3.0支持集群

B. 数据存在内存，定期写入磁盘，如果磁盘不够，可以选择LRU算法删除数据。

C. 数据结构丰富，包括hash,set,list,string,zset

Mongodb适合于冷数据（超过一天）存储海量数据，redis适合于热数据（比如1天内频繁访问的），响应快

### 10、高并发以及高负载

* 1. 服务器常见代理模式
     1. 正向代理：客户端通过代理服务器访问原始服务器。对于客户端而言，正向代理所代理的是客户端，那么客户端必须要进行一些特别的设置。
     2. 反向代理：客户端发送请求到代理服务器， 然后代理服务器将请求转发给内部网络上的其他服务器（原始服务器），并将从原始服务器上得到的结果返回给客户端，此时代理服务器就是代理的服务端，客户端无须进行特别的设置，对外就表现为一个反向代理服务器
  2. JavaWEB高并发的处理措施
     1. 程序设计上运用队列缓存，动态网页静态化(模板)等技术
     2. 项目部署架构上运用服务器集群达到分布式**负载均衡**效果
  3. 负载均衡
     1. 负载均衡是指将请求/数据【均匀】分摊到多个操作单元上执行，负载均衡的关键在于均匀，均衡的分摊压力。而现实最简单的方式就是服务器集群架构(Cluster)。
     2. 常见负载均衡技术：DNS轮询、IP负载均衡和CDN
  4. Web服务器集群
     1. Nginx
        1. Nginx介绍：免费开源的反向代理Web服务器软件
           1. 聚焦于高性能，高并发和低内存消耗问题。
           2. 并且具有多种Web服务器功能特性：负载均衡，缓存，访问控制，带宽控制，以及高效整合各种应用的能力，这些特性使Nginx很适合于现代网站架构。
        2. Nginx详细配置：修改conf 目录下 nginx.conf 文件

### 八、分布式微服务

### 1、什么是微服务

以前的模式是所有的代码在同一个工程中部署在同一个服务器中，同一个项目的不同模块不同功能互相抢占资源。

微服务将工程根据不同的业务规则拆分成微服务，微服务部署在不同的机器上，服务之间进行相互调用

Java微服务的框架有 dubbo（只能用来做微服务），spring cloud（提供了服务的发现，断路器等）

### 2、springcloud如何实现服务的注册和发现

服务在发布时指定对应的服务名，将服务注册到注册中心（eureka或者zookeeper等），这一过程是springcloud自动实现的，只需要在main方法添加@EnableDiscoveryClient，同一个服务修改端口就可以启动多个实例。

调用方法：传递服务名称通过注册中心获取所有的可用实例，通过负载均衡策略调用（ribbon和feign）对应的服务

### 3、微服务中如何实现 session 共享 ?

在微服务中，一个完整的项目被拆分成多个不相同的独立的服务，各个服务独立部署在不同的服务器上，各自的 session 被从物理空间上隔离开了，但是我们需要在不同微服务之间共享 session ，常见的方案就是 Spring Session + Redis 来实现 session 共享。将所有微服务的 session 统一保存在 Redis 上，当各个微服务对 session 有相关的读写操作时，都去操作 Redis 上的 session 。这样就实现了 session 共享。

### 4、Spring Cloud核心组件，在微服务架构中，分别扮演的角色：

• Eureka：各个服务启动时，Eureka Client都会将服务注册到Eureka Server，并且Eureka Client还可以反过来从Eureka Server拉取注册表，从而知道其他服务在哪里

• Ribbon：服务间发起请求的时候，基于Ribbon做负载均衡，从一个服务的多台机器中选择一台

• Feign：基于Feign的动态代理机制，根据注解和选择的机器，拼接请求URL地址，发起请求

• Hystrix：发起请求是通过Hystrix的线程池来走的，不同的服务走不同的线程池，实现了不同服务调用的隔离，避免了服务雪崩的问题

• Zuul：如果前端、移动端要调用后端系统，统一从Zuul网关进入，由Zuul网关转发请求给对应的服务

### 5、Ribbon和Feign的区别：

Ribbon和Feign都是用于调用其他服务的，不过方式不同。

1.启动类使用的注解不同，Ribbon用的是@RibbonClient，Feign用的是@EnableFeignClients。

2.服务的指定位置不同，Ribbon是在@RibbonClient注解上声明，Feign则是在定义抽象方法的接口中使用@FeignClient声明。

3.调用方式不同，Ribbon需要自己构建http请求，模拟http请求然后使用RestTemplate发送给其他服务，步骤相当繁琐。

Feign则是在Ribbon的基础上进行了一次改进，采用接口的方式，将需要调用的其他服务的方法定义成抽象方法即可，不需要自己构建http请求。不过要注意的是抽象方法的注解、方法签名要和提供服务的方法完全一致。

### 6、springcloud断路器（Hystrix）的作用

当一个服务调用另一个服务由于网络原因或者自身原因出现问题时，调用者就会等待被调用者的响应 当更多的服务请求到这些资源时，导致更多的请求等待，这样就会发生连锁效应（雪崩效应），断路器就是解决这一问题。

断路器有完全打开状态: 一定时间内，达到一定的次数无法调用，并且多次检测没有恢复的迹象，断路器完全打开，那么下次请求就不会请求到该服务。

半开: 短时间内有恢复迹象，断路器会将部分请求发给该服务，当能正常调用时断路器关闭。

关闭: 当服务一直处于正常状态，能正常调用，断路器关闭。

### 7、ActiveMQ

ActiveMQ概述： 我们知道JMS只是消息服务的一组规范和接口，并没有具体的实现，而ActiveMQ就是JMS规范的具体实现；它是Apache下的一个项目，采用Java语言开发；是一款非常流行的开源消息服务

基本要素：1、生产者producer ; 2、消费者consumer ; 3、消息服务broker

    Producer客户端使用来发送消息的， Consumer客户端用来消费消息；它们的协同中心就是ActiveMQ broker,broker也是让producer和consumer调用过程解耦的工具，最终实现了异步数据交换的功能。

JMS两种消息传送模式：

点对点（ Point-to-Point）：专门用于使用队列Queue传送消息；基于队列Queue的点对点消息只能被一个消费者消费，如多个消费者都注册到同一个消息队列上，当生产者发送一条消息后，而只有其中一个消费者会接收到该消息，而不是所有消费者都能接收到该消息。

  发布/订阅（Publish/Subscribe）：专门用于使用主题Topic传送消息。基于主题的发布与订阅消息能被多个消费者消费，生产者发送的消息，所有订阅了该topic的消费者都能接收到。

### 8、RabbitMQ

RabbitMQ核心概念：

Message

消息, 消息是不具名的，它由消息头和消息体组成，消息体式不透明的，而消息头则由一系列的可选属性组成，这些属性包括routing-key（路由键）、priority（相对于其他消息的优先权）、delivery-mode（指该消息可能需要持久性存储）等。

Publisher

消息的生产者，也是一个向交换器发布消息的客户端应用程序。

Exchange

交换器，用来接收生产者发送的消息并将这些消息路由给服务器中的队列。Exchange有4种类型：direct（默认）、fanout，topic，和headers，不同类型的Exchange转发消息的策略有所区别

Queue

消息队列，用来保存消息直到发送给消费者。它是消息的容器，也是消息的终点。一个消息可投入一个或多个队列。消息一直在队列里面，等待消费者连接到这个队列将其取走。

Binding

绑定，用于消息队列和交换器之间的关联。一个绑定就是基于路由键将交换器和消息队列连接起来的路由规则，所以可以交换器理解成一个由绑定构成的路由表。Exchange和Queue的绑定可以是多对多的关系。

Connection

网络连接，比如一个TCP连接。

Channel

信道，多路复用连接中的一条独立的双向数据流通道，信道是建立在真实的TCP连接内的虚拟连接，不管是发布消息、订阅队列还是接收消息，这些动作都是通过信道完成。因为对于操作系统来说建立和销毁TCP都是比较昂贵的开销，所以引入了信道的概念，以复用一条TCP连接。

Consumer

消息的消费者，表示一个从消息队列中取得消息的客户端应用程序。

Virtual Host

虚拟主机，表示一批交换器、消息队列和相关对象。

Broker

表示消息队列服务器实体

## **9、[如何解决消息队列数据丢失问题？](https://hacpai.com/article/1542161520849)**

丢数据一般会有 3 种情况：  
1）生产者往 MQ 写数据丢失了，可能因为网络原因数据传输丢失。  
2）MQ 本身数据丢失了，数据放在内存中，MQ 机器宕机了，那么内存中的数据丢失了。  
3）消费者消费的时候丢失了，消费者接受到消息，马上给 MQ 返回已经接受到消息的回复，接着处理逻辑，在处理过程中，消费者宕机了；下次重启的时候消费者会获取到下一条数据消费，这样数据就丢失了。  
  
问题解决：  
**1、解决 rabbitmq 生产者丢数据**

1）事务机制  
在生产者发送消息的时候，加上事务

channel.txSelecttry{//发送消息 如果MQ接收失败会抛异常，那么对事务进行回滚}catch(Exception e){channel.txRollback}channel.txCommit

该方案完全保证发送消息时成功的，但是缺点是：使用了事务之后，消息变成同步了，影响吞吐量。比较耗性能。

2）confirm 模式  
在生产者开启 confirm 模式，每个消息分配唯一 id；生产者发送完消息之后就不用管了，rabbitmq 如果成功接收到消息，那么会调用生产者的 ack(String messageId)方法；rabbitmq 如果接收失败，那么会调用生产者 nack(String messageId)方法，生产者根据情况判断是否要重试；

小结：一般情况下，使用 confirm 模式，性能、吞吐量都要比事务机制高。confirm 是异步的，不会阻塞。

**2、解决 rabbitmq 宕机弄丢了数据**

就是开启持久化，把内存中的数据刷到磁盘中。如果在写入磁盘的时候挂了，这样的话还是会有数据丢失，但是概率很小，你可以等数据持久化到磁盘，在通知生产者 ack 方法，如果没有持久化成功，生产者长时间没收到消息可以尝试在重试一次。

**3、解决 rabbitmq 消费端弄丢了数据**

在消费端这边，需要关闭 autoAck，等自己处理完之后在发送 ack 给 MQ。其他的 MQ，无外乎也都是这些原因，只要在各自端进行一些配置就可以解决数据丢失问题

### 10、分布式事务

### ****单体应用****

单体应用中，一个业务操作需要调用三个模块完成，此时数据的一致性由本地事务来保证。

### ****微服务应用****

随着业务需求的变化，单体应用被拆分成微服务应用，原来的三个模块被拆分成三个独立的应用，分别使用独立的数据源，业务操作需要调用三个服务来完成。此时每个服务内部的数据一致性由本地事务来保证，但是全局的数据一致性问题没法保证。

### ****c48ykaxdk****

### **小结：**在微服务架构中由于全局数据一致性没法保证产生的问题就是分布式事务问题。简单来说，一次业务操作需要操作多个数据源或需要进行远程调用，就会产生分布式事务问题。

### 11、Seata简介(Simple Extensible Autonomous Transaction Architecture)

Seata (简单可扩展自治事务框架，简化版GTS)是一款开源的分布式事务解决方案，致力于提供高性能和简单易用的分布式事务服务。Seata 将为用户提供了 AT、TCC、SAGA 和 XA 事务模式，为用户打造一站式的分布式解决方案。

## ****Seata原理和设计****

### ****定义一个分布式事务****

我们可以把一个分布式事务理解成一个包含了若干分支事务的全局事务，全局事务的职责是协调其下管辖的分支事务达成一致，要么一起成功提交，要么一起失败回滚。此外，通常分支事务本身就是一个满足ACID的本地事务。这是我们对分布式事务结构的基本认识，与 XA 是一致的。

### ****协议分布式事务处理过程的三个组件****

1、Transaction Coordinator (TC)：事务协调器，维护全局事务的运行状态，负责协调并驱动全局事务的提交或回滚；

2、Transaction Manager (TM)：控制全局事务的边界，负责开启一个全局事务，并最终发起全局提交或全局回滚的决议；

3、Resource Manager (RM)：控制分支事务，负责分支注册、状态汇报，并接收事务协调器的指令，驱动分支（本地）事务的提交和回滚。

### ****一个典型的分布式事务过程****

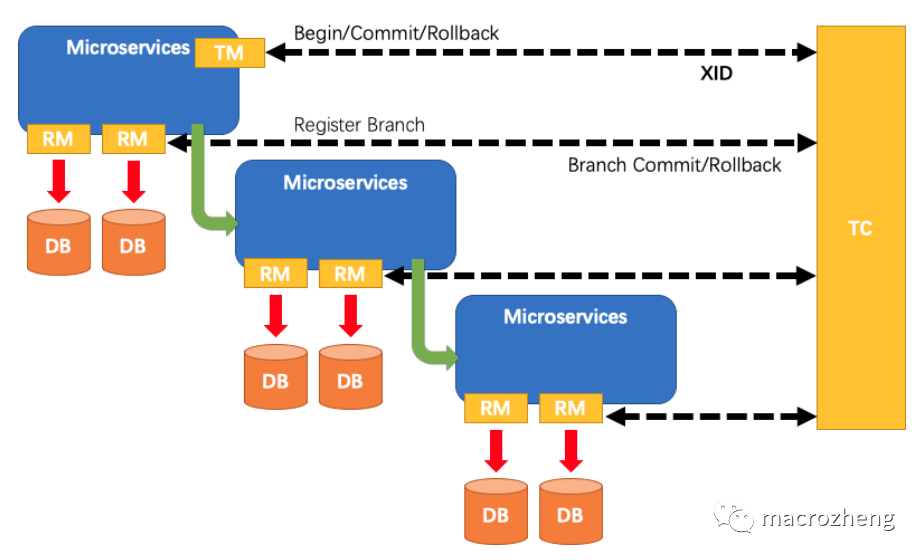
1、TM 向 TC 申请开启一个全局事务，全局事务创建成功并生成一个全局唯一的 XID；

2、XID 在微服务调用链路的上下文中传播；

3、RM 向 TC 注册分支事务，将其纳入 XID 对应全局事务的管辖；

4、TM 向 TC 发起针对 XID 的全局提交或回滚决议；

5、TC 调度 XID 下管辖的全部分支事务完成提交或回滚请求。



# 12、分布式锁

分布式锁，是一种思想，它的实现方式有很多。比如基于数据库、memcached、Redis、zookeeper等。如果我们将沙滩当做分布式锁的组件，那么它看起来应该是这样的：

**加锁：**在沙滩上踩一脚，留下自己的脚印，就对应了加锁操作。其他进程或者线程，看到沙滩上已经有脚印，证明锁已被别人持有，则等待。(redisConnection.setNX(key.getBytes(),new byte[]{1}))

**解锁：**把脚印从沙滩上抹去，就是解锁的过程。(redisTemplate.delete(key);)

**锁超时：**为了避免死锁，我们可以设置一阵风，在单位时间后刮起，将脚印自动抹去。(redisTemplate.expire(key,expire,TimeUnit.SECONDS);)

### 九、其他

### 1、 Shiro

1. 基本介绍：Apache Shiro 是一个强大易用的java安全框架，提供了认证、授权、加密和会话管理等功能。

2. Shiro基本架构：

1) Subject：Subject 代表当前“用户”。与 Subject 的所有交互都会委托给 SecurityManager；Subject 其实是一个门面，SecurityManager 才是实际的执行者；

2) SecurityManager：安全管理器；Shiro框架的核心。Shiro通过SecurityManager来管理内部组件实例，并通过它来提供安全管理的各种服务。

3) Realm：Shiro 从 Realm 获取安全数据（如用户、角色、权限）。从 Realm 获取相应的用户进行比较以确定用户身份是否合法；也需要从 Realm 得到用户相应的角色/权限进行验证用户是否能进行操作。

3. 架构流程

1) 应用代码通过Subject来进行认证和授权，而Subject又委托给SecurityManager；

2) 我们需要给Shiro的SecurityManager注入Realm，从而让SecurityManager能得到合法的用户及其权限进行判断。

### 2、 Eharts

1. 基本介绍：

1) ECharts，一个纯 Javascript 的图表库，可以流畅的运行在 PC 和移动设备上，兼容当前绝大部分浏览器，提供直观，生动，可交互，可高度个性化定制的数据可视化图表。

2. 图表类型：

1) ECharts 提供了常规的折线图，柱状图，散点图，饼图，K线图，用于统计的盒形图，用于地理数据可视化的地图，热力图，线图等等，并且支持图与图之间的混搭。

3. Echarts操作过程

1) 在GitHub或者官网上下载Echarts最新版本

2) 在页面中引入Echarts的文件，即js文件

3) 绘制图表（官网demo中复制代码）

### 3、 在线支付

国内电子商务系统实现的基本流程如下：(以支付宝为例)

客户在系统内下订单 -> 系统根据订单生成支付宝接口url -> 客户通过url使用支付宝（网上银行）付款 -> 支付宝将客户的付款完成信息发送给电子商务系统 -> 系统收到支付宝信息后确定客户订单已经付款 -> 进行发货等后续流程。

### 4、 SOA/WebService

1. 什么是WebService？

1) WebService是一个SOA（面向服务的编程）的架构，它是不依赖于语言，不依赖于平台，可以实现不同的语言间的相互调用，通过Internet进行基于Http协议的网络应用间的交互。

2) WebService实现不同语言间的调用，是依托于一个标准，webservice是需要遵守WSDL（web服务定义语言）/SOAP（简单请求协议）规范的。

2. WebService的实现技术和作用？

1) 在构建和使用Web Service时,主要用到以下几个关键的技术和规则:

a) XML:描述数据的标准方法.

b) SOAP:表示信息交换的协议.

c) WSDL:Web服务描述语言.

d) UDDI(Universal Description, Discovery and Integration):通用描述、发现与集成，它是一种独立于平台的，基于XML语言的用于在互联网上描述商务的协议。

3. WebService的优点：

1) 可以让异构的程序相互访问（跨平台）

2) 松耦合

3) 基于标准协议（通用语言，允许其他程序访问）

### 5、 SVN/GIT

1. SVN和GIT的区别

1) 首先两者都是版本控制系统。但是Git是分布式的，而SVN不是分布式，是集中式的。

2) GIT和SVN的分支概念不同。GIT中任意开启分支，且不会影响其他分支的代码运行。

3) Git没有一个全局版本号，而SVN有：目前为止这是跟SVN相比Git缺少的最大的一个特征。

4) SVN提交（Commit），当直接记录到中央版本库。并且如果和服务器的网络中断，将根本无法提交。而GIT提交完全属于本地版本库的活动。你只需要推送（git push）到主要版本库就可以。

5) 总结：管理项目用svn，管理代码用git。

2. SVN的优缺点

1) 优点

a) 管理方便，逻辑明确，符合一般人思维习惯。

b) 易于管理，集中式服务器更能保证安全性。

c) 代码一致性非常高。

d) 适合开发人数不多的项目开发。缺点

2) 缺点

a) 服务器压力太大，数据库容量暴增。

b) 如果不能连接到服务器上，基本上不可以工作。

c) 不适合开源开发。但是一般集中式管理的有非常明确的权限管理机制（例如分支访问限制），可以实现分层管理，从而很好的解决开发人数众多的问题。

3. GIT的优缺点

1) 优点

a) 适合分布式开发，强调个体。

b) 公共服务器压力和数据量都不会太大。

c) 速度快、灵活。

d) 任意两个开发者之间可以很容易的解决冲突。

e) 离线工作。

2) 缺点

a) 学习周期相对而言比较长。

b) 不符合常规思维。

c) 代码保密性差，一旦开发者把整个库克隆下来就可以完全公开所有代码和版本信息。

### 6、 任务调度开源框架Quartz

各企业应用几乎都会碰到任务调度的需求，就拿论坛来说：每隔半个小时生成精华文章的RSS文件，每天凌晨统计论坛用户的积分排名，每隔30分钟执行锁定用户解锁任务。Quartz 在开源任务调度框架中的翘首，它提供了强大任务调度机制，难能可贵的是它同时保持了使用的简单性。Quartz 允许开发人员灵活地定义触发器的调度时间表，并可以对触发器和任务进行关联映射。

SpringBoot注解任务调度:

@EnableScheduling:开启任务调度的支持

@Scheduled:配置触发任务的触发器

@Scheduled注解可以控制方法定时执行，其中参数可选择：

定时器的任务方法不能有返回值

1、fixedDelay|fixedDelayString控制方法执行的间隔时间，是以上一次方法执行完开始算起，如上一次方法执行阻塞住了，那么直到上一次执行完，并间隔给定的时间后，执行下一次。

2、fixedRate|fixedRateString是按照一定的速率执行，是从上一次方法执行开始的时间算起，如果上一次方法阻塞住了，下一次也是不会执行，但是在阻塞这段时间内累计应该执行的次数，当不再阻塞时，一下子把这些全部执行掉，而后再按照固定速率继续执行。

3、initialDelay|initialDelayString :初次执行任务之前需要等待的时间

4、cron表达式可以定制化执行任务，但是执行的方式是与fixedDelay相近的，也是会按照上一次方法结束时间开始算起。

### 7、WebSocket

一、WebSocket：(Tomcat7+支持,HTML5新特性，取代comet)

WebSocket协议是基于TCP的一种新的网络协议。它实现了浏览器与服务器全双工(full-duplex)通信——允许服务器主动发送信息给客户端。

简单的说，WebSocket协议之前，双工通信是通过不停发送HTTP请求，从服务器拉取更新来实现，这导致了效率低下。WebSocket解决了这个问题。

WebSocket的连接一旦建立，服务器和客户端都可以互推信息。所以在做一些需要即时通信的功能时候，我们首选就是用WebSocket。

二、实现原理：

在实现websocket连线过程中，需要通过浏览器发出websocket连线请求，然后服务器发出回应，这个过程通常称为“握手”。然后，浏览器和服务器之间就形成了一条快速通道。两者之间就直接可以数据互相传送。在此WebSocket 协议中，为我们实现即时服务带来了两大好处：

1. Header

互相沟通的Header是很小的-大概只有 2 Bytes

2. Server Push

服务器的推送，服务器不再被动的接收到浏览器的请求之后才返回数据，而是在有新数据时就主动推送给浏览器。

三、HTML5 Web Socket API：

使用WebSocket有两种方式：1是使用sockjs，2是使用h5的标准。

使用sockjs:前端引入sockjs-x.x.min.js,var websocket = new SockJS("http://...");

在HTML5中内置有一些API，用于响应应用程序发起的请求。

### 8、使用SpringBoot+WebSocket

1、POM

　　核心是@ServerEndpoint这个注解。这个注解是Javaee标准里的注解，Tomcat7以上已经对其进行了实现，如果是用传统方法使用tomcat发布项目，只要在pom文件中引入javaee标准即可使用。

但使用springboot的内置tomcat时，就不需要引入javaee-api了，spring-boot已经包含了。使用springboot的websocket要引入springboot组件。

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-websocket</artifactId>

</dependency>

spring-boot-starter-websocket依赖spring-boot-starter-web和spring-boot-starter

2、使用@ServerEndpoint创立 websocket endpoint

使用Springboot内置容器要注入ServerEndpointExporter，这个bean会自动注册使用了@ServerEndpoint注解声明的Websocket endpoint。

要注意，如果使用独立的servlet容器，就不要注入ServerEndpointExporter，因为它将由容器自己提供和管理。

### 9、Solr和Elasticsearch

Solr(2004年)在搜索领域占据了多年的主导地位。在2010年后，Elasticsearch成为市场上的另一种选择。

Elasticsearch:

Elasticsearch是一个基于[Lucene](https://baike.baidu.com/item/Lucene/6753302" \t "https://baike.baidu.com/item/elasticsearch/_blank)的搜索服务器。它提供了一个分布式多用户能力的全文搜索引擎，基于RESTful web接口。Elasticsearch是用Java语言开发的，并作为Apache许可条款下的开放源码发布，是一种流行的企业级搜索引擎。

Solr:

Solr 是Apache下的一个顶级开源项目，采用Java开发，它也是基于Lucene的全文搜索服务器。Solr提供了比Lucene更为丰富的查询语言，同时实现了可配置、可扩展，并对索引、搜索性能进行了优化。

Elasticsearch 与 Solr 的比较总结:

Solr 利用 Zookeeper 进行分布式管理，而 Elasticsearch 自身带有分布式协调管理功能;

Solr 支持更多格式的数据(xml,json,pdf,txt,doc,ppt)，而 Elasticsearch 仅支持json文件格式；

Solr 官方提供的功能更多，而 Elasticsearch 本身更注重于核心功能，高级功能多有第三方插件提供；

Solr 在传统的搜索应用中表现好于 Elasticsearch，但在处理实时搜索应用时效率明显低于 Elasticsearch。

Solr 是传统搜索应用的有力解决方案，但 Elasticsearch 更适用于新兴的实时搜索应用。

### 10、为什么要分库分表？

单库单表存在的问题：假设你要设计一个电商网站，在一开始，User表、Order表、Product表等等各种表都在同一个数据库中，每个表都包含了大量的字段。在用户量比较少，访问量也比较少的时候，单库单表不存在问题。

但是公司可能发展的比较好，用户量开始大量增加，而且一张表存储的数据还很多，高达几千万数据，更难受的是这样的表还挺多。于是一个数据库的压力就太大了，一张表的压力也比较大。试想一下，我们在一张几千万数据的表中查询数据，压力本来就大，如果这张表还需要关联查询，那压力就更大了。

1. 单库太大：数据库里面的表太多，所在服务器磁盘空间装不下，IO次数多CPU忙不过来。
2. 单表太大：一张表的字段太多，数据太多。查询起来困难。

此时就开始考虑分库分表了。

### 11、如何分库分表

单库单表下越来越不满足需求，此时我们先考虑进行读写分离。我们将数据库的写操作和读操作进行分离，使用多个从库副本（Slaver）负责读，使用主库（Master）负责写，从库从主库同步更新数据，保持数据一致。这在一定程度上可以解决问题，但是用户超级多的时候，比如几个亿用户，此时写操作会越来越多，一个主库（Master）不能满足要求了，那就把主库拆分，这时候为了保证数据的一致性就要开始进行同步。

不管是分库还是分表，都有两种切分方式：水平切分和垂直切分。下面我们分别看看如何切分。

1、分表

（1）垂直分表

表中的字段较多，一般将不常用的、 数据较大、长度较长的拆分到“扩展表“。一般情况加表的字段可能有几百列，此时是按照字段进行数竖直切。注意垂直分是列多的情况。

（2）水平分表

单表的数据量太大。按照某种规则（RANGE,HASH取模等），切分到多张表里面去。 但是这些表还是在同一个库中，所以库级别的数据库操作还是有IO瓶颈。这种情况是不建议使用的，因为数据量是逐渐增加的，当数据量增加到一定的程度还需要再进行切分。比较麻烦。

2、分库

（1）垂直分库

一个数据库的表太多。此时就会按照一定业务逻辑进行垂直切，比如用户相关的表放在一个数据库里，订单相关的表放在一个数据库里。注意此时不同的数据库应该存放在不同的服务器上，此时磁盘空间、内存、TPS等等都会得到解决。

（2）水平分库

水平分库理论上切分起来是比较麻烦的，它是指将单张表的数据切分到多个服务器上去，每个服务器具有相应的库与表，只是表中数据集合不同。 水平分库分表能够有效的缓解单机和单库的性能瓶颈和压力，突破IO、连接数、硬件资源等的瓶颈。

3、分库分表之后的问题

（1）联合查询困难

联合查询不仅困难，而且可以说是不可能，因为两个相关联的表可能会分布在不同的数据库，不同的服务器中。

（2）需要支持事务

分库分表后，就需要支持分布式事务了。数据库本身为我们提供了事务管理功能，但是分库分表之后就不适用了。如果我们自己编程协调事务，代码方面就又开始了麻烦。

（3）跨库join困难

分库分表后表之间的关联操作将受到限制，我们无法join位于不同分库的表，也无法join分表粒度不同的表， 结果原本一次查询能够完成的业务，可能需要多次查询才能完成。 我们可以使用全局表，所有库都拷贝一份。

（4）结果合并麻烦

比如我们购买了商品，订单表可能进行了拆分等等，此时结果合并就比较困难。