面试宝典

**一、JAVA**

一、Json和XMl的区别

Json是一种轻量级的数据交互格式。

优点：

1、数据格式比较简单，易于解析、便于维护

2、压缩过的，网络传输占带宽较小。

3、支持多种语言

4、简化了代码量

缺点：不通用，暂时只在web service应用。

Xml是一种扩展标记语言。

优点：

1、格式统一，符合标准，便于与其他系统进行交互。

缺点：

1、文件格式复杂，难以解析和维护。

2、XML文件庞大，网络传输占带宽

**二、DOM和SAX**

SAX解析方式：逐行扫描文档，一边扫描一边解析。相比于DOM，SAX可以在解析文档的任意时刻停止解析，是一种速度更快，更高效的方法。

优点：解析可以立即开始，速度快，没有内存压力

缺点：不能对结点做修改

DOM解析方式：DOM解析器在解析XML文档时，会把文档中的所有元素，按照其出现的层次关系，解析成一个个Node对象（节点）

优点：把XML文件在内存中构建属性结构，可以遍历和修改节点。

缺点：如果文件比较大，内存有压力，解析的时间会比较长。

**三、数据库乐观锁和悲观锁**

乐观锁：乐观的锁，它的思想是获取数据到提交前数据是没有被修改的。适用于多读少写场景。

乐观锁实现机制有两种：版本号机制和CAS算法

版本号机制：在数据库表中加一个版本号字段，表示当前数据被修改的次数，每当数据被修改时版本号就加1。核心思想是提交后的版本号永远比当前数据的版本号大，否则提交失败。

CAS算法（非阻塞同步）：核心思想就是比较与交换。其中涉及到三个操作数，内存值、预期值、新值。当且仅当内存值和预期值相等的时候将新值赋值给内存值，否则一直自旋。

1、CAS会涉及到ABA问题即数据的值从A修改到B再修改回A，CAS检测不出来，会认为数据没有被修改过逻辑不严谨。

2、多线程抢夺资源严重的时候自旋的记录很大，从而会浪费很多的CPU资源，效率低下。

悲观锁：它的思想就是拿数据的时候总认为别人已经修改了数据。因此悲观锁会机制会在操作前上锁，Java中synchronized和ReentrantLock等独占锁就是悲观锁思想的实现。悲观锁适用于少读多写的场景。

\*、共享锁、排他锁、行级锁、表级锁

数据库的增、删、改操作默认都会加排他锁，而查询不会加任何锁。

共享锁：也称为读锁，对某一资源加共享锁，自身可以读该资源，其他人也可以读该资源，共享锁可以添加多个。

select \* from table lock in share mode;

排他锁：也称为写锁，对某一资源加排他锁，只能自身修改该资源，排他锁需要等共享锁释放才能添加。

select \* from table for update;

行级锁：行级锁不是锁记录而是锁索引，如果条件是主键则会锁定该记录对应的主键索引，如果是其他唯一索引则会先锁定索引再锁定主键。如果是普通字段则会对整个表进行扫描因此会锁定整个表也就是表级锁。

**四、Volatile修饰词**

volatile ：不能体现原子性，但是可以体现可见性和有序性。volatile修饰的变量被修改后会立即刷新到主内存中当其他线程需要获取该变量时会重新到主内存中读取

java内存模型

java内存模型规定所有变量都存储在主内存中，每个线程拥有独立的工作内存，每个线程只能和自己的工作内存进行交互，工作内存可以和主内存进行交互，线程之间的变量传递只能通过主内存来完成。这种模式在单线程是没有什么问题的，但是在多线程情况下就会出现数据不同步的问题。

工作内存和主内存之间的交互过程：

read（读取）：作用于主内存变量，把一个变量值从主内存传输到线程的工作内存中，以便随后的load动作使用

load（载入）：作用于工作内存的变量，它把read操作从主内存中得到的变量值放入工作内存的变量副本中。

use（使用）：作用于工作内存的变量，把工作内存中的一个变量值传递给执行引擎，每当虚拟机遇到一个需要使用变量的值的字节码指令时将会执行这个操作。

assign（赋值）：作用于工作内存的变量，它把一个从执行引擎接收到的值赋值给工作内存的变量，每当虚拟机遇到一个给变量赋值的字节码指令时执行这个操作。

store（存储）：作用于工作内存的变量，把工作内存中的一个变量的值传送到主内存中，以便随后的write的操作。

write（写入）：作用于主内存的变量，它把store操作从工作内存中一个变量的值传送到主内存的变量中。

volatile修饰的变量能保证：

read、load、use为不可分割操作即读取的永远是最新的数据。

assign、store、write为不可分割操作即修改其他线程可以看见。

java并发特性：

原子性：不可分割即当前操作不会受到其他线程的干扰。

可见性：当前线程修改的数据其他线程也可以看的到。

有序性：程序执行顺序按照代码顺序执行。

注：jvm是即时编译会有可能重新排序，重新排序原则是没有依赖关系的代码可能会改变顺序。重新排序在单线程是没有

问题的，但是多线程就有问题。假设有123三条语句，13,12有关联，23是没有关联的所以23的顺序可以被重新排列即132，假设第二条语句很重要，线程1运行到3语句后被线程2抢到资源执行，它此时需要语句2的条件进行判断此时程序的走向就会不一样了。

**五、进程和线程的区别**

1、进程是资源分配的最小单元，线程是程序执行的最小单元。

2、进程有自己的独立地址空间，每启动一个进程，系统就会为它分配地址空间，建立数据表来维护代码段、堆栈段和数据段。而线程是共享进程中的数据的，使用相同的地址空间，因此CPU切换一个线程的花费远比进程要小很多，同时创建一个线程的开销也比进程要小很多

3、线程之间的通信更方便。

**六、spring依赖**

Setter方式依赖注入,构造函数依赖注入，两者同时存在先构造

**七、Autowired、Inject、Resource注解的区别**

依赖注入时执行顺序：

@Autowired and @Inject

1、Matches by Type(默认)

2、Restricts by Qualifiers

3、Matches by Name

@Resource

1、Matches by Name(默认)

2、Matches by Type

3、Restricts by Qualifiers (ignored if match is found by name)

上述Name是指类型的默认名称即首字母小写。

@Qualifier注解默认指定的是限定名称即你自定义的名称如果匹配不上就会根据Name的规则去匹配，当给定@Qualifier则不会从默认开始执行依赖注入（当然@Resource配合@Qualifier比较特殊规则如上括号中注释）。

区别：

@Autowired：是Spring框架才有的注解

@Resource和@Inject：不是Spring框架的注解

**八、死锁具体体现，举例生活例子**

假设线程A、B，线程A运行需要线程B的锁，线程B运行需要线程A的锁因此就会产生死锁。

例如：1、魔塔游戏，游戏中有许多房间，房间中有钥匙。假设房间A中有房间B去往房间C的钥匙，房间B中有房间A去往房间B的钥匙，此时就无法通过了也就死锁。

2、两个线程A、B,两个数据库操作,第一个数据库操作加了共享锁，第二个数据库操作添加了排他锁。当A线程执行了共享锁操作，此时线程B获得执行权限,它也执行共享锁,但是执行到排他锁的时候会阻塞因为A线程中的加了共享锁。排他锁需要等共享锁全部释放了才能添加。此时就会产生死锁

**九、软件生命周期**

1、问题定义

2、可行性研究

3、需求分析

4、总体设计

5、详细设计

6、编码和单元测试

7、综合测试

8、软件维护

**十、集群和分布式的区别**

1、分布式是指将不同的业务分布在不同的地方。 而集群指的是将几台服务器集中在一起，实现同一业务。分布式中的每一个节点，都可以做集群。 而集群并不一定就是分布式的。分布式是并联工作的，集群是串联工作的

2、分布式是以缩短单个任务的执行时间来提升效率的，而集群则是通过提高单位时间内执行的任务数来提升效率。

集群特性：

可扩展性：集群的性能不限于单一的服务实体，新的服务实体可以动态地加入到集群，从而增强集群的性能。

高可用性：集群通过服务实体冗余使客户端免于轻易遇到out of service的警告。在集群中，同样的服务可以由多个服务实体提供。如果一个服务实体失败了，另一个服务实体会接管失败的服务实体。集群提供的从一个出 错的服务实体恢复到另一个服务实体的功能增强了应用的可用性。

集群能力：

负载均衡,错误恢复

集群技术：

集群地址,内部通信

**十一、消息队列**

**十二、索引**

数据库索引是用来提升数据查询的速度。创建索引也会创建表，会占内存空间。常见的索引有主键索引、唯一索引、普通索引、组合索引。查询字段中有索引则会先去索引表中查找出主键然后通过主键再快速的查找出记录。如果查询字段在索引表中就有则直接会返回记录。

创建索引：create index[indexname] on table(fieldname);

优点：提高查询速度

缺点：

1. 增删改的时候需要同时修改索引表结构影响效率
2. 创建索引占空间，如果索引表很大会很耗时间

**十三、面向对象的思想、特性**

封装：在java的世界里万事万物都可以看做为对象。将属性和行为封装在对象中。封装可以隐藏对象内部细节，对象只向外部曝露接口，可以控制属性和行为的读写，提高了代码的安全性，但是降低了耦合度。

继承：继承在java中是一种关系，只适用于父子类并且只支持单继承。java类中有属性和行为，为了避免相同属性和行为重复编写使用继承可以很好的减少代码冗余。同时继承和实现也是多肽实现的一个关键步骤。

多肽：一种行为多种表现形式。具体实现方法是首先具有继承或实现关系，其次子类或实现类需要重写相同方法最后是子类对象指向父类引用。多肽会损失子类新增的行为。

**十四、数据库范式**

第一范式：属性不可分割如数据库表中有一个电话属性则不能在该属性下再分为座机、手机等属性。不满足第一范式的数据库，不是关系数据库。

第二范式：前提满足一范式，非主属性完全依赖于码（唯一关键字）

第三范式：前提满足第二范式：消除传递依赖

**十五、大学计算机教程**

计算机科学导论、数字电路与逻辑设计、计算机组成原理、数据结构与算法、操作系统原理、汇编语言程序设计、数据库系统原理、编译原理、软件工程导论、计算机网络

**十六、List集合可以添加null值吗**

可以

**十七、selep和wait区别**

1. selep是Thread类中的方法；wait是Object中的方法。
2. 线程调用selep不会释放对象锁休眠时间结束后继续运行；wait会释放对象锁并进入等待池中并需要调用notify进行唤醒才会继续执行。

**十八、数据库优化**

1. 适当使用索引可以增加查询速度
2. 语句优化尽量不使用in like null !< or等语句；使用UNION合并相同查询条件的查询语句
3. 使用事务

**十九、JVM原理**

我们开发者编写程序需要安装jdk。Jdk包含jre和一些开发工具。Jre是java运行的环境。Jre包含jvm和核心类库。因此从组成上来说它就是必不可缺的。Jvm是用来解析class文件并将解析内容分配在内存区域。虚拟机内存区域主要可分为6个部分：

1. 程序计数器，代码行号指示器，这个是线程私有的。
2. Java虚拟机栈：空间比较小，同样也是线程私有。主要存放局部变量，引用地址，方法入口出口等信息。
3. 本地方法栈：为非Java编写的本地定义的栈空间和java虚拟机栈类似。
4. 方法区：方法区是线程共享的，在JVM启动时创建。它存储了运行时常量池、字段和方法信息、静态变量以及被JVM载入的所有类和接口的方法的字节码。
5. 常量池：常量池也是线程共享的，一个存储了类文件格式中的常量池表的内存空间，这部分空间存在于方法区内。
6. Java虚拟机堆：这个空间比较大，是线程共享的。是实际存放数据的地方。同时也是jvm虚拟机进行垃圾回收的地方。

\*、java垃圾回收机制

设计的知识点：新生代、老年代、持久代、Eden区、Survive区

新生代：比较活跃，对象被经常创建和死亡，包含Eden区和2个Survive区，比例为8:1:1。

新创建的较小的对象都会被放入Eden区（很大的对象会直接被放入老年代中）当Eden区空间不够分配给新创建的对象时它就会触发minGc将Eden区和一个Survive区的还存活的对象放入另外一个Survive区此时还存活的对象年龄+1，当年龄到达老年区的限制时候就会放入老年代区。

老年代：主要存放应用程序中生命周期长的内存对象，当内存不够分配给新加入到老年代的对象时会触发majorGc，清理死亡的对象。

永久代：存放的是类信息，GC不会清理此区域的数据。

**二十、谈谈对Spring的理解**

首先，Spring是一个轻量级的框架，旨在方便开发。Spring框架中有很多模板如数据获取、事务机制、面向切面、web支持、IOC容器等。其中最重要的就是IOC容器模块，它包含Beans、core、context三个主要的组件。Core包含的是一些公共的类我们常称为Utils其他类都要依赖于它。Beans组件也是IOC中最重要的一部分因为它是用来创建Bean的，Bean是这个容器的主角就像一个盛水的缸中的水没有水这个缸也就没有意义了。Context是上下文的意思，它是这个容器的环境同时也提供了获取bean的途径。IOC容器的思想就是控制翻转和依赖注入即可以将传统的主动new对象交个spring框架来处理，我们只需要给它一个引用即可。IOC的好处是动态产生（区别于工厂模式）、创建的对象是单例模式，同时容器还会帮我管理对象的生命周期。

**二十一、B-树、B+树（难）**

**二十二、负载均衡算法与实现**

一、算法

1、轮询

轮询算法把每个请求轮流发送到每个服务器上。

2、加权轮询

在轮询基础上根据服务器的性能进行加大或缩小触发的概率

3、最少连接数

某个服务器当前最少连接那么下一个连接就发送给最少连接的服务器上。

4、加权最小连接数

同上

5、随机算法

随机发送给服务器，适用于服务器性能相似。

二、实现

1、DNS解析

2、修改MAC地址（链路层）

3、修改IP地址（网络层）

4、HTTP重定向

5、反向代理

**二十三、正向代理和反向代理**

代理服务器的优点：

1. 提高访问速度

客户端访问的数据会缓存在代理服务器上，下次访问相同节点可以直接读取。

1. 防火墙作用

因为所用请求都要从代理服务器上经过，因此可以做一些限制起到防火墙的作用。

1. 突破访问限制

本身不可访问，但是代理服务器可以访问，因此可以做到突破访问限制

正向代理和反向代理的区别：

正向代理是一个位于客户端和原始服务器之间的服务器，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。

适用场景：访问外网、隐藏自己信息、缓存

反向代理是代理服务器架设在服务器端，客户端根本不知道代理服务器的存在。

适用场景：安全、负载均衡

区别：正向代理是客户端和代理服务器相对于服务器端是一个整体。反向代理是代理服务器和服务器端是一个整体。

二十三、Hibernate和Mybatis

ORM：对象映射关系即表字段和实体类字段进行映射。

Mybatis：Mybatis是一个半自动ORM框架即Sql语句还是需要自己来写，因此数据库移植性较低。Mybatis有固定的格式，按照配置Bean-Interface-xml进行配置就可以使用，入门比较简单，但是相对而言它修改起来也比较麻烦因为需要同时修改多个文件，相对开发速度比较慢。二级缓存机制不佳，日志系统欠缺。

Hibernate：Hibernate是全自动ORM框架即只要表和实体类即可，对于简单的数据库操作Hibernate已经封装好了可以直接调用，可以大大提高开发速度，但是对于复杂的数据操作需要自己写Sql和映射比较麻烦。Hibernate拥有完整的日志系统

二十四、事务

事务特性：原子性、一致性、隔离性、持久性

并发事务会产生的问题：

1. 脏读：即事务A读取事务B未提交的数据
2. 不可重复读：即两次读取的数据不一致
3. 幻读：即读取了不存在的数据

事务隔离级别：

1. read\_unCommitted：读未提交，三个问题都会发生，级别最低
2. read\_Committed:读并提交，这种隔离级别可以解决脏读的问题，但是不能解决不可重复读和幻读的问题。
3. Repeatable\_Read：重复读，不能解决幻读的问题，在mysql中此级别可以解决幻读问题。
4. SERLALIZABLE：顺序读取，级别最高。一个一个事务按顺序进行执行。

事务隔离级别越高性能越差，安全性越高。