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。

**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。

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。

**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

**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：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。

**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**

**十一、消息队列**

**十二、索引**

**十三、面向对象的思想、特性**

**十四、数据库范式**

**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**