1. ***线程怎么保持同步***

**什么是线程同步？**

当使用多个线程来访问同一个数据时，非常容易出现线程安全问题(比如多个线程都在操作同一数据导致数据不一致),所以我们用同步机制来解决这些问题。

1同步方法synchronized

2同步代码块synchronized

3在变量上加volatile（jvm在多线程操作该共享数据时就会判断它是否正在被其他线程使用和操作）

4ThreadLocal管理对象

5ReenTrantlock的lock()unlock方法

1. ***spring中bean的周期是怎样的***

***随着spring容器的创建而创建，然后实例化，然后初始化，然后使用，然后销毁***

1. ***给了一个场景，在一个数组中怎么找出两个相加是是10的数***
2. ***dubbo但是对使用方式和分布式架构原理***

dubbo的使用方式：

**节点角色说明：**

Provider: 暴露服务的服务提供方。

Consumer: 调用远程服务的服务消费方。

Registry: 服务注册与发现的注册中心。

Monitor: 统计服务的调用次调和调用时间的监控中心。

Container: 服务运行容器。spring容器

**调用关系说明：**

1. 服务提供者在启动时，向注册中心注册自己提供的服务。

2. 服务消费者在启动时，向注册中心订阅自己所需的服务。

3. 注册中心返回服务提供者地址列表给消费者，如果有变更，注册中心将基于长连接推送变更数据给消费者。

4. 服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，如果调用失败，再选另一台调用。

5. 服务消费者和提供者，在内存中累计调用次数和调用时间，定时每分钟发送一次统计数据到监控中心。

分布式架构原理：

分布式是为了解决项目的模块之间耦合性高，单台机器的性能上限较低的问题。分布式就是将一个系统拆分成不同的子系统并部署在不同的服务器上。分布式解耦合之后还方便单个模块的扩展，也就是将单个模块设置成集群。分布式系统之间进行通信的方式是dubbo服务中间件和mq消息中间件。

1. ***抽象类和接口区别，多线程及其同步方式***

抽象类和接口的区别：

抽象类和接口的对比：

|  |  |  |
| --- | --- | --- |
| **参数** | **抽象类** | **接口** |
| 默认的方法实现 | 它可以有默认的方法实现 | 接口完全是抽象的。它根本不存在方法的实现 |
| 实现 | 子类使用**extends**关键字来继承抽象类。如果子类不是抽象类的话，它需要提供抽象类中所有声明的方法的实现。 | 子类使用关键字**implements**来实现接口。它需要提供接口中所有声明的方法的实现 |
| 构造器 | 抽象类可以有构造器 | 接口不能有构造器 |
| 与正常Java类的区别 | 除了你不能实例化抽象类之外，它和普通Java类没有任何区别 | 接口是完全不同的类型 |
| 访问修饰符 | 抽象方法可以有**public**、**protected**和**default**这些修饰符 | 接口方法默认修饰符是**public**。你不可以使用其它修饰符。 |
| main方法 | 抽象方法可以有main方法并且我们可以运行它 | 接口没有main方法，因此我们不能运行它。 |
| 多继承 | 抽象方法可以继承一个类和实现多个接口 | 接口只可以继承一个或多个其它接口 |
| 速度 | 它比接口速度要快 | 接口是稍微有点慢的，因为它需要时间去寻找在类中实现的方法。 |
| 添加新方法 | 如果你往抽象类中添加新的方法，你可以给它提供默认的实现。因此你不需要改变你现在的代码。 | 如果你往接口中添加方法，那么你必须改变实现该接口的类。 |

什么时候使用抽象类什么时候使用接口：

如果你拥有一些方法并且想让它们中的一些有默认实现，那么使用抽象类吧。

如果你想实现多重继承，那么你必须使用接口。由于**Java不支持多继承**，子类不能够继承多个类，但可以实现多个接口。因此你就可以使用接口来解决它。

如果基本功能在不断改变，那么就需要使用抽象类。如果不断改变基本功能并且使用接口，那么就需要改变所有实现了该接口的类。（其实也是说的默认实现）

多线程：线程是由进程创建的，进程是一个正在运行的软件。一个进程可以创建多个线程，多个线程共享进程的资源。任何一个进程中至少要有一个线程。多线程的意义在于可以让计算机软件同时执行多个任务，比如说qq有很多功能，这些功能可以同时使用，每个功能都是一个线程。在单核处理器下，如果他的单个处理线程占用CPU为100%的话，那是用多线程只会导致效率降低。  
现在是多核处理器，使用多线程可以提高CPU使用率，既提高效率。  
还有一种情况就是，单个线程处理数据时有等待现象，导致不能满负荷使用CPU，也可以使用多线程。

***面试***：

1. ***服务器线程还在，但是服务器无响应，你的解决思路？***

说明线程进入了阻塞态，第一种原因是线程所在的进程抢不到cpu，第二个原因是无法获得同步锁（或死锁）。抢不到的解决方案是关闭一些进程减小服务器压力，死锁的解决方案是编程时尽量避免死锁，或者将优先级较低的线程终止。

1. ***系统设计，架构设计，JVM原理，classloader原理***

***架构设计：***

架构就是软件的结构，架构应该由需求决定，比如分哪几个功能模块；架构也包括技术选型，不同的应用场景需要用到不同的技术。架构的设计要有前瞻性，不仅要满足目前的需求还要方便日后的业务扩展。

***jvm原理：***

jvm包含堆，栈，方法区，类加载器，垃圾回收器等，类加载有三种，垃圾回收机制有两种。栈里面是变量，形参，基本类型的数据和变量，堆里面是引用类型的数据，方法区里面是类信息、静态变量、final类型的常量、属性和方法信息等只需要加载一次或不需要修改的内容。JVM用持久代来存放方法区。当局部变量生命周期结束后，栈空间立刻被回收，堆空间区域等待GC回收。

***classloader类加载器原理：***

类加载器是用来将硬盘中的class文件加载到内存中生成class对象。类加载的过程是加载---连接---初始化。

1. ***aop的实现原理***

也就是切面的原理，见其他笔记。它是面向切面编程，用于业务层对事务的处理。

1. ***sevlet实现原理***

1.Servlet介绍

Servlet它是JavaWEB三大组件(Filter/Listener)之一，也是最重要的一个！ 它是一个接口(规范)，servlet是运行在 Web服务器中的小型 Java程序

servlet 通常通过 HTTP（超文本传输协议）接收和响应来自 Web客户端的请求。 要想你编写的一个类成为一个servlet，那么必须实现这个Servlet接口(必须在web.xml文件中进行配置)

2.Servlet作用

Servlet主要用于接收用户的请求，并对请求进行相关的处理(具体的业务需求)，最终将处理的结果响应给客户端！

3.实现Servlet的方式(有3种)

第一种方式：

实现Servlet接口，并实现该接口下指定的5个方法！

第二种方式：

继承Servlet的已知实现类GenericServlet(抽象类)，那么必须实现抽象方法 service(ServletRequest request,ServletResponse response)

第三种实现方式：

继承Servlet的已知实现类HttpServlet(抽象类)，根据需要(如果是get请求，书写doGet方法;如果是post请求，书写doPost方法)重写doXXX方法！

4.Servlet的生命周期

生命周期：一个对象的创建到销毁

创建的问题：但凡与Servlet相关的对象，都不是程序员自己创建的，而是由服务器容器tomcat来负责创建的！

默认情况，用户第一次发送请求，由服务器容器创建Servlet实例(一种类型的Servlet只会创建一个实例对象)，会调用初始化方法init(ServletConfig config)

该方法在整个Servlet生命周期中只会被执行一次！紧接着会调用服务方法service(ServletRequest request,ServletResponse response),该方法在整个

Servlet生命周期中会被调用多次(用户发送一次请求，调用一次)，当正常关闭服务器或者将这个项目从服务器容器中移除的时候，调用销毁方法destory(),该方法在整个

Servlet生命周期中只会被调用一次(在关闭服务器之前调用)

# SpringMVC和Struts2中是并发访问否会存在线程安全问题。

对于使用过SpringMVC和Struts2的人来说，大家都知道SpringMVC是基于方法的拦截，而Struts2是基于类的拦截。

对于Struts2来说，因为每次处理一个请求，struts就会实例化一个对象；这样就不会有线程安全的问题了;

而Spring的controller默认是Singleton的，这意味着每一个request过来，系统都会用原有的instance去处理，这样导致两个结果：

一是我们不用每次创建Controller，二是减少了对象创建和垃圾收集的时间;由于只有一个Controller的instance，当多个线程调用它的时候，它里面的instance变量就不是线程安全的了，会发生窜数据的问题。

有几种解决方法：  
1、在Controller中使用ThreadLocal变量  
2、在spring配置文件Controller中声明 scope="prototype"，每次都创建新的controller。

***5、设计模式***

其他笔记中有。

1. ***继承和实现的理解***

继承是对父类的继承，实现是对接口的实现。继承偏向于分类，也就是该类属于哪种类；接口偏向于动态添加功能，需要哪个功能就加上哪个接口。接口需要实现具体方法，继承的复用性更强，可以直接复用方法。接口效率高于继承父类，因为继承父类的话就要先实例化父类。继承只能继承一个父类，接口可以实现很多个。

***项目组是偏开放性的问题，更注重于解决问题的思路和方法***

***Round1：***

1. ***ConcurrentHashMap内部原理？它是如何实现多线程的***

要想用线程安全的hashmap，可以用hashtable。它是对整个hash表上锁，所以效率不高。ConcurrentHashMap是将哈希表分成很多个segment,分别对每个segment进行上锁，所以修改效率比较高，等待时间较短。

1. ***线程池，线程池里面核心线程数和最大线程数有什么区别***

线程池是为了解决线程频繁开启和关闭所产生的资源消耗。线程可以放在线程池里面，用的时候拿出来，不用的时候还回去。线程池中的线程处于阻塞态，不会占用cpu。

如果当前线程池任务线程数量小于核心线程池数量，执行器总是优先创建一个任务线程，

而不是从线程队列中取一个空闲线程。

如果当前线程池任务线程数量大于核心线程池数量，执行器总是优先从线程队列中取一个空闲线程， 而不是创建一个任务线程。

如果当前线程池任务线程数量大于核心线程池数量，且队列中无空闲任务线程，将会创建 一个任务线程，直到超出maximumPoolSize，如果超时maximumPoolSize，则任务将会被拒绝。

1. ***spring和springBoot比较，springboot使用场景***

springboot的优点是不用配置xml文件，有默认配置，假如需要自定义功能的话可以修改默认配置项。开箱即用（将本来需要做的其他工作做成了依赖包），搭建速度快。springboot可以搭建独立的spring应用程序。内嵌了tomcat，既可以用传统方式部署又可以直接跑起来（java -jar）。提供了一些例如表单验证等的第三方功能。简化Maven的配置。

***4、springboot的部署如何区分开发环境和生产环境（application.properties）***

***（对于已上线的版本，如何将生产环境修改成为开发环境）***

为不同的环境设置不同的配置文件，更改配置即可。

***5、分布式（没用过的话不会细问）***

***Round2：***

***1、int类型数组，取到两两相加结果等于某给定变量的所有组合方式***

***直接for循环，说出代码（时间复杂度）***

0.5\*n\*(n+1)

***HashMap/HashSet方式（假定数组里面的数值没有重复项）***

***排序方式***

为了排除同时出现ab.ba的情况，首先将取到的键值对从大到小排列，然后放入hashmap，自动保证了唯一性，然后计算键值对的和。

1. ***volatile关键字***

用于给变量加锁，jvm会判断该变量是否存在多线程问题，如果存在就加锁。避免了多线程数据的安全问题。

1. ***序列化/反射***

反射见其他笔记。

序列化就是将对象转换成字节流。序列化有两个作用，一是将对象转换成字节流在网络中进行传输，第二个是将对象存储到磁盘中。

***4、多线程单例（创建一个多线程单例，说出大致代码）***

见其他笔记

***5、针对数据查询量过大进行优化操作***

1创建索引（缺点是插入变慢，数据量变大）。

2将常用信息放到redis缓存。

3slow\_query\_log分析：当你设置slow\_query\_log为on的时候，server端会对每次的查询进行记录，当超过你设置的慢查询时间(long\_query\_time)的时候就把该条查询记录到日志。而你对性能进行优化的时候，就可以分析慢查询日志，对慢查询的查询语句进行有目的的优化。可以通过创建各种索引，可以通过分表等操作。那为什么要分库分表那，当不分库分表的时候那个地方是限制性能的地方啊。

4分库分表：

分库分表应该算是查询优化的杀手锏了。上述各种措施在数据量达到一定等级之后，能起到优化的作用已经不明显了。这个时候就必须对数据量进行分流。分流一般有分库与分表两种措施。而分表又有垂直切分与水平切分两种方式。下面我们就针对每一种方式简单介绍。

  对于mysql，其数据文件是以文件形式存储在磁盘上的。当一个数据文件过大的时候，操作系统对大文件的操作就会比较麻烦与耗时，而且有的操作系统就不支持大文件，所以这个时候就必须分表了。另外对于mysql常用的存储引擎是Innodb，它的底层数据结构是B+树。当其数据文件过大的时候，B+树就会从层次和节点上比较多，当查询一个节点的时候可能会查询很多层次，而这必定会导致多次IO操作进行装载进内存，肯定会耗时的。除此之外还有Innodb对于B+树的锁机制。对每个节点进行加锁，那么当更改表结构的时候，这时候就会树进行加锁，当表文件大的时候，这可以认为是不可实现的。

  所以综上我们就必须进行分表与分库的操作。

5子查询优化

在查询中经常会用到子查询，在子查询的时候一般使用in或者exist关键词。针对in和exist在查询的时候当数据量大到一定程度以后，查询执行时间就差别比较大。但是，为了避免此类情况出现，最好的方式是使用join查询。因为在绝大多数情况下，服务器对join的查询优化要远远高于子查询优化。在比较高的版本5.6，mysql查询会自动把in查询优化成joint查询，就不会出现子查询比较慢的问题。有时候也可以采用distinct关键词来限制子查询的数量，但是需要注意的是distinct很多时候会转化为group by，这个时候就会出现一个**临时表**，就会出现copy数据到临时表的时延。

***熟练线程池，知晓核心线程数和最大线程数的区别；知晓4种保持线程同步的方式（Synchronized,volatile, ReentrantLock，ThreadLocal ），知晓java.util.ConcurrentHashMap 的原理，熟练使用DUBBO，但没有阅读过DUBBO代码。 有重构项目的经历。***

1. ***arraylist和linkedlist的区别。Linkedlist是双向的吗？***

arraylist是数组，查找快增删慢；linkedllist是链表，增删快查找慢。

双向链表。next属性指向后一个元素，previous指向前一个元素

1. ***hashmap的原理（先hashcode还是先equal）。***

先hashcode后equal

1. ***线程的sleep()和wait()的区别（我是结合生产者和消费者问题回答的）。***

sleep() 和 wait() 的区别就是 调用sleep方法的线程不会释放对象锁，而调用wait() 方法会释放对象锁。前者可以设定时间，后者需要nodify手动唤醒。

1. ***线程池的原理与使用场景。***

例如数据库连接池。

1. ***Hibernante和mybatis的区别，mybatis的主键回填。***

selectKey元素防止返回的id为null。

1. ***Dubbo的zk集群挂掉，dubbo还能使用吗（PS：dubbo有缓存）。***

可以的，启动dubbo时，消费者会从zk拉取注册的生产者的地址接口等数据，缓存在本地。每次调用时，按照本地存储的地址进行调用。

1. ***Zk的节点数量为什么是奇数个（考察ZK原理）。***

zookeeper有这样一个特性：集群中只要有过半的机器是正常工作的，那么整个集群对外就是可用的。也就是说如果有2个zookeeper，那么只要有1个死了zookeeper就不能用了，因为1没有过半，所以2个zookeeper的死亡容忍度为0；同理，要是有3个zookeeper，一个死了，还剩下2个正常的，过半了，所以3个zookeeper的容忍度为1；同理你多列举几个：2->0;3->1;4->1;5->2;6->2会发现一个规律，2n和2n-1的容忍度是一样的，都是n-1，所以为了更加高效，何必增加那一个不必要的zookeeper呢。

1. ***消息中间件RabbitMQ，消息的丢失是怎么处理的，监听器没有收到消息，MQ会丢掉数据吗？MQ会不会将消息缓存到硬盘？***

关闭自动确认，改成消费者手动确认。但是消费者忘记确认的话会产生内存泄漏。不会缓存到硬盘，会缓存到内存中。

***9、 算法，数组排序。***

***10、 Sql的left join和right join的区别。***

***11、 Mybatis的join标签好像是，记不太清楚了，考察mybatis标签。***

1. ***Zk分布式锁实现原理。***

**分布式锁获取思路**

**获取分布式锁的总体思路**

        在获取分布式锁的时候在locker节点下创建临时顺序节点，释放锁的时候删除该临时节点。客户端调用createNode方法在locker下创建临时顺序节点，

然后调用getChildren(“locker”)来获取locker下面的所有子节点，注意此时不用设置任何Watcher。客户端获取到所有的子节点path之后，如果发现自己在之

前创建的子节点序号最小，那么就认为该客户端获取到了锁。如果发现自己创建的节点并非locker所有子节点中最小的，说明自己还没有获取到锁，

此时客户端需要找到比自己小的那个节点，然后对其调用exist()方法，同时对其注册事件监听器。之后，让这个被关注的节点删除，则客户端的Watcher会

收到相应通知，此时再次判断自己创建的节点是否是locker子节点中序号最小的，如皋是则获取到了锁，如果不是则重复以上步骤继续获取到比自己小的一个

节点并注册监听。当前这个过程中还需要许多的逻辑判断。

***2、currenthashmap和hashmap 的区别。***

***3、spring aop 原理。***

***4、你觉得你最值得骄傲的项目，什么架构。分布式事务方案，怎么解决分布式事务。***

基于消息中间件的两阶段提交往往用在高并发场景下，将一个分布式事务拆成一个消息事务（A系统的本地操作+发消息）+B系统的本地操作，其中B系统的操作由消息驱动，只要消息事务成功，那么A操作一定成功，消息也一定发出来了，这时候B会收到消息去执行本地操作，如果本地操作失败，消息会重投，直到B操作成功，这样就变相地实现了A与B的分布式事务。

***1、 arraylist和linkedlist的区别。Linkedlist是双向的吗？***

***2、 hashmap的原理（先hashcode还是先equal）。***

***3、 线程的sleep()和wait()的区别（我是结合生产者和消费者问题回答的）。***

***4、 线程池的原理与使用场景。***

***5、 Hibernante和mybatis的区别，mybatis的主键回填。***

***6、 Dubbo的zk集群挂掉，dubbo还能使用吗（PS：dubbo有缓存）。***

***7、 Zk的节点数量为什么是奇数个（考察ZK原理）。***

***8、 消息中间件RabbitMQ，消息的丢失是怎么处理的，监听器没有收到消息，MQ会丢掉数据吗？MQ会不会将消息缓存到硬盘？***

***9、 算法，数组排序。***

***10、 Sql的left join和right join的区别。***

***11、 Mybatis的join标签好像是，记不太清楚了，考察mybatis标签。***

***12、Zk分布式锁实现原理。***

***13、currenthashmap和hashmap 的区别。***

***14、spring aop 原理。***

***15、你觉得你最值得骄傲的项目，什么架构。分布式事务方案，怎么解决分布式事务。***

***（主管偏向于解决实际问题和全局性的掌握～偏架构）***

***第一问最近的项目设计思路及适用场景***

***第二问，项目采用soa设计架构所使用的技术***

***第三问，分布式服务架构框架dubbo的实现原理***

***第四问，dubbo中采用服务注册发布zookeeper实现原理***

***第五问，Spring实现原理***

***第六问，项目中采用消息队列的适用场景***

***第七问，多线程变成适用场景***

***第八问，项目中涉及大数据量的处理方式***

***第九问，大数据量情况下分表操作的设计方式***

***第十问，针对分表后的数据，如何适应业务需要，如统计，数据浏览等应对策略***