途牛面试：（答案不知道对不对）

自我介绍

1.Mybatis的#和$有什么区别？两者都有什么弊端?

"#" 是将传入的参数都当做一个字符串，会对传入的参数加一个双引号。相当于PreparedStatement;

如group by #myname# ，如果传入的值是et,那么解析成sql时就为group by "et"，

如果传入的值是id,则解析成SQL就是 group by "id"

所以在select \* from ？ 或者 truncate table ?时这样传入数据库对象（表名或属性名）时不能使用

"$" 是将传入的参数直接显示生成在SQL中,相当于是普通的Statement。

如 Order by $user\_id$,如果传入的值是111,那么解析成sql时的值为order by user\_id,

如果传入的值是id，则解析成的sql为order by id.

$方式一般用于传入数据库对象，例如传入表名.

2.JAVA内存模式？

一种是栈内存，另一种是堆内存

(1)在函数中定义的基本类型变量（即基本类型的局部变量）和对象的引用变量（即对象的变量名）都在函数的栈内存中分配；

(2)堆内存用来存放由new创建的对象和数组以及对象的实例变量（即全局变量）。

在函数（代码块）中定义一个变量时，java就在栈中为这个变量分配内存空间，当超过变量的作用域后，java会自动释放掉为该变量所分配的内存空间；

在堆中分配的内存由java虚拟机的自动垃圾回收器来管理

堆和栈的优缺点

堆的优势是可以动态分配内存大小，生存期也不必事先告诉编译器，因为它是在运行时动态分配内存的。

缺点就是要在运行时动态分配内存，存取速度较慢；

栈的优势是，存取速度比堆要快，仅次于直接位于CPU中的寄存器。

3。说说ioc和aop的底层实现原理？

工厂模式（ioc容器+反射）

代理模式（cglib和jdk动态代理）

4。ioc有两种功能都是什么

管理对象和控制反转

5.cilib代理和jdk动态代理的区别？

<1.JDK动态代理

此时代理对象和目标对象实现了相同的接口，目标对象作为代理对象的一个属性，具体接口实现中，可以在调用目标对象相应方法

前后加上其他业务处理逻辑。代理模式在实际使用时需要指定具体的目标对象，如果为每个类都添加一个代理类的话，会导致类很多

，同时如果不知道具体类的话，怎样实现代理模式呢？这就引出动态代理。

JDK动态代理只能针对实现了接口的类生成代理。

<2.CGLIB代理

CGLIB（CODE GENERLIZE LIBRARY）代理是针对类实现代理，主要是对指定的类生成一个子类，覆盖其中的所有方法，所以该类或

方法不能声明称final的。

6.quartz定时任务机制？表达式？

任务的调度过程大致分为三步:获取待触发trigger、触发trigger、实例化并执行Job

cron表达式用于配置cronTrigger的实例。cron表达式实际上是由七个子表达式组成。这些表达式之间用空格分隔。

(1.Seconds （秒）

(2.Minutes（分）

(3.Hours（小时）

(4.Day-of-Month （天）

(5.Month（月）

(6.Day-of-Week （周）

(7.Year（年）

7.说一说线程锁吧？有什么区别

在Java5中，专门提供了锁对象，利用锁可以方便的实现资源的封锁，用来控制对竞争资源并发访问的控制，这些内容主要集中在java.util.concurrent.locks 包下面，里面有三个重要的接口Condition、Lock、ReadWriteLock。

Condition:

Condition 将 Object 监视器方法（wait、notify 和 notifyAll）分解成截然不同的对象，以便通过将这些对象与任意 Lock 实现组合使用，为每个对象提供多个等待 set （wait-set）。

Lock:

Lock 实现提供了比使用 synchronized 方法和语句可获得的更广泛的锁定操作。

ReadWriteLock:

ReadWriteLock 维护了一对相关的锁定，一个用于只读操作，另一个用于写入操作。

8.项目中用到线程了吗？怎么用的？

用到了。线程池

9.。线程池用的哪个,什么接口,使用的哪种队列？

//可重用的线程池 其中参数代表预留多少活跃线程

ExecutorService es = Executors.newFixedThreadPool(3);

//带缓存机制的线程池 也就是在任务执行结束60秒内线程不消亡

ExecutorService es = Executors.newCachedThreadPool();

//单一实例的线程执行器

ExecutorService es = Executors.newSingleThreadExecutor();

搜了下有三种队列：直接提交，有界队列，无界队列

排队有三种通用策略：

直接提交。工作队列的默认选项是 SynchronousQueue，它将任务直接提交给线程而不保持它们。在此，如果不存在可用于立即运行任务的线程，则试图把任务加入队列将失败，因此会构造一个新的线程。此策略可以避免在处理可能具有内部依赖性的请求集时出现锁。直接提交通常要求无界 maximumPoolSizes 以避免拒绝新提交的任务。当命令以超过队列所能处理的平均数连续到达时，此策略允许无界线程具有增长的可能性。

无界队列。使用无界队列（例如，不具有预定义容量的 LinkedBlockingQueue）将导致在所有 corePoolSize 线程都忙时新任务在队列中等待。这样，创建的线程就不会超过 corePoolSize。（因此，maximumPoolSize的值也就无效了。）当每个任务完全独立于其他任务，即任务执行互不影响时，适合于使用无界队列；例如，在 Web页服务器中。这种排队可用于处理瞬态突发请求，当命令以超过队列所能处理的平均数连续到达时，此策略允许无界线程具有增长的可能性。

有界队列。当使用有限的 maximumPoolSizes时，有界队列（如 ArrayBlockingQueue）有助于防止资源耗尽，但是可能较难调整和控制。队列大小和最大池大小可能需要相互折衷：使用大型队列和小型池可以最大限度地降低 CPU 使用率、操作系统资源和上下文切换开销，但是可能导致人工降低吞吐量。如果任务频繁阻塞（例如，如果它们是 I/O边界），则系统可能为超过您许可的更多线程安排时间。使用小型队列通常要求较大的池大小，CPU使用率较高，但是可能遇到不可接受的调度开销，这样也会降低吞吐量。

10.redis应用场景是什么，怎么用的

高并发查询数据

redis存储的数据类型是什么？

redis是作为一个key-value型的数据存储仓库，因此存储的数据类型是键值对类型

其中 key 使用字符串表示

value 可以是普通的字符数据（字符串），也可以是基本数据类型；当然也可以存储 List、Set、Map类型 ， 自定义类型对象通过实现序列化也可以存储在redis中。

redis怎样使用的？怎样存储数据的？主要存储的什么类型？（不是说以键值对的形式）

首先在服务器中启动redis服务 : redis-server.exe redis.conf

连接redis可以采用spring提供的spring-data-redis组件，该组件封装了jedis访问redis服务器的过程 ：

首先在ioc容器中配置redis连接池(JedisPoolConfig)、连接工厂(JedisConnectionFactory)和模板类对象(RedisTemplate)

然后将RedisTemplate注入给业务对象，通过使用模板类实现对redis服务器的连接和操作。

存储数据 ：

redisTemplate.set(key , value) get(key)

rt.opsForList() -> ListOperations -> rightPush() leftPush() range()

rt.opsForSet() -> SetOperations -> add() members()

rt.opsForHash() -> HashOperations -> put(key,value-key,value) entries(key)

get(key,value-key)

11.restful接口怎么调用的

HttpClient调的,通过url，post,get方法实例什么的

12.为什么用的是webservice调接口？选择这种方式的原因是什么？有跟其他方式比较过吗

因为人家用的webservice发布的接口啊，我又不能选，md

13.说说反射？

reflection是java语言提供的一套用来在运行期动态获得类中的信息的API。

通过反射，我们可以在运行期动态的获得类中的属性和方法；

通过反射，我们可以在运行期动态的执行类中的方法；

通过反射，我们可以在运行期动态的判断对象的类型；

通过反射，我们可以在运行期动态创建类的对象；

Class:用来表示类，一个Class对象代表一个具体的类型

\* Field:用来标示属性 一个Field对象代表一个具体的属性

\* Method:用来标示方法，一个Method对象代表一个具体的方法

14.插入一个对象除了equals()方法还需要注意什么方法

hashCode() 和 ==

15.equals()方法和==区别？

==是比较地址，equals()可以通过覆盖自定义比较规则

16.&和&&的区别？

&可以按位运算替代%

&&有短路特性