1. mybaties一级缓存、二级缓存

一级缓存是SqlSession级别的缓存。在操作数据库时需要构造 sqlSession对象，在对象中有一个数据结构（HashMap）用于存储缓存数据。不同的sqlSession之间的缓存数据区域（HashMap）是互相不影响的。

二级缓存是mapper级别的缓存，多个SqlSession去操作同一个Mapper的sql语句，多个SqlSession可以共用二级缓存，二级缓存是跨SqlSession的。

1. Hibbernate一级缓存、二级缓存（sessionFactory弱引用）

Hibernate的缓存包括Session的缓存和SessionFactory的缓存，其中SessionFactory的缓存又可以分为两类：内置缓存和外置缓存。Session的缓存是内置的，不能被卸载，也被称为Hibernate的第一级缓存。SessionFactory的内置缓存和Session的缓存在实现方式上比较相似，前者是SessionFactory对象的一些集合属性包含的数据，后者是指Session的一些集合属性包含的数据。SessionFactory的内置缓存中存放了映射元数据和预定义SQL语句，映射元数据是映射文件中数据的拷贝，而预定义SQL语句是在Hibernate初始化阶段根据映射元数据推导出来，SessionFactory的内置缓存是只读的，应用程序不能修改缓存中的映射元数据和预定义SQL语句，因此SessionFactory不需要进行内置缓存与映射文件的同步。SessionFactory的外置缓存是一个可配置的插件。在默认情况下，SessionFactory不会启用这个插件。外置缓存的数据是数据库数据的拷贝，外置缓存的介质可以是内存或者硬盘。SessionFactory的外置缓存也被称为Hibernate的第二级缓存。

内存溢出原因：当你对数据库进行数量级有万级、十万级、百万级甚至千万级别的.hibernate会把操作的对象全部放到自身的内部缓存来进行管理。这样就消耗了大量的内存，导致内存溢出。

解决方法：

1、优化Hibernate，程序上采用分段插入及时清除缓存的方法。

2、绕过Hibernate API,直接通过JDBC API来做批量插入，这个方法性能上是最好的，也是最快的。

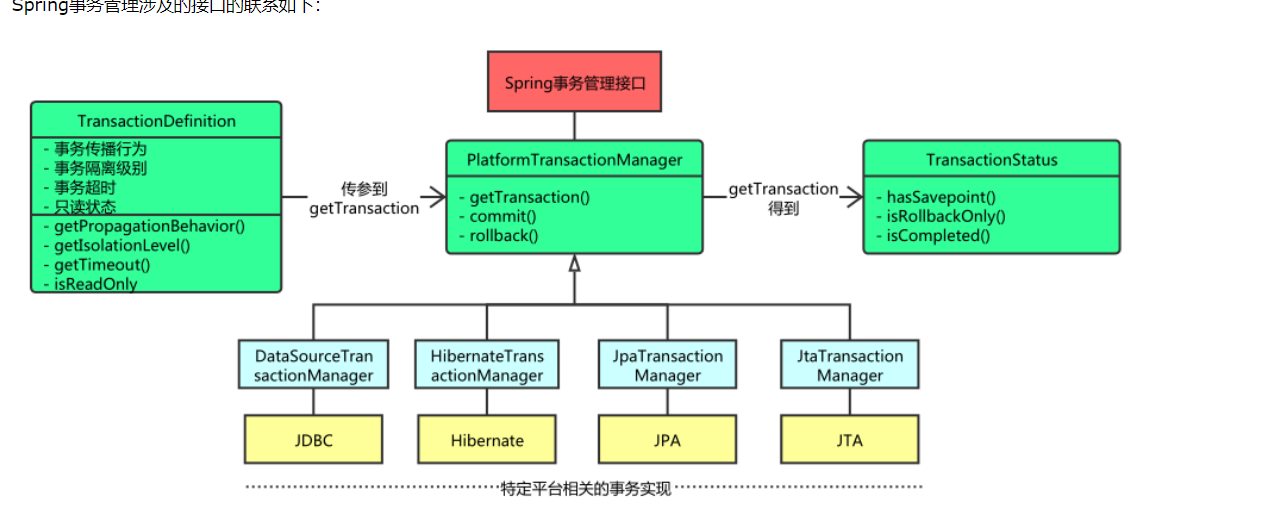
3、事务提交之后直接清除缓存

1. Aop前置增强、后置增强、环绕增强
2. Dubbo 协议

(Dubbo、rmi、http、thrift、webService、hessian、redis、memcached)

Dubbo缺省协议采用单一长连接NIO异步通信，适合小数据量大并发的服务调用，以及服务消费者机器远大于服务提供者机器数的情况。

1. Spring事物（7种、5种）



1. 如何通过反射拿到private的属性和方法（暴力获取）

Reflex reflex = new Reflex();

Field field1=reflex.getClass().getDeclaredField("field1");  
field1.setAccessible(true);

System.*out*.println(field1.get(reflex));

1. 动态代理、静态代理

为了解决每有一个被代理类就需要创建一个代理类的弊端，就出现了动态代理，动态代理就是把所有东西都抽象出来。

1. 泛型被擦除、变形、形变（标识可重用锁）

泛型在编译的时候就被删除

1. I/O字节流、字符流

字节流的抽象流：InputStream、OutputStream

字符流的抽象流: Reader、Writer

1. Spring理解（IOC怎么初始化）
2. Spring的两种代理JDK和CGLAB
3. Java动态代理是利用反射机制生成一个代理接口的匿名类，在调用具体方法前调用InvokeHandler来处理。
4. Cglib动态代理是利用asm开源包，对代理对象的class文件加载进来，通过修改其字节码生成子类来处理。
5. Springboot的加载机制（优先注入spring容器中）
6. Zookeeper什么情况下会出现脑裂（在断网的情况下）会利用选举算法解决
7. HashMap1.8（数组+链表的形式）当链表个数大于8的时候会形成红黑树，线程安全的hashMap（CurrentHashMap、集合类.SynchronizedHashMap）
8. JUC是？并发包

在Java5.0提供Java.util.concurrent包

1. HTTP协议、状态码（请求重定向状态码是302）

超文本传输协议，HTTP采用了请求/响应模型，灵活（HTTP允许传输任意类型的数据对象）、无连接（限制每次链接只处理一个请求）、无状态（协议对于事务处理没有记忆能力）

1XX 100-101 信息提示

2XX 200-206 成功

3XX 300-305 重定向

4XX 400-415 客户端错误

5XX 500-505 服务器错误

1. 类加载机制

类加载机制的生命周期：

1、加载（将类的Class对象加载到内存中）

2、验证（验证Class对象是否符合jvm的要求，即对jvm是否产生安全危害）

3、准备（将静态变量加载到内存中）

4、解析（将符号引用替换为直接引用）

5、初始化(将类的静态变量及静态方法区进行初始化)

BootStrap ClassLoader(根加载器)

Extension ClassLoader（扩展加载器）

Application ClassLoader（应用程序加载器）

UserClassLoader（用户自定义加载器）

1. JVM工作内存模型（每个线程都有自己的工作副本与主线程进行数据交互），怎么保证内存的可见性（利用锁 volatile）
2. ThreadLocal在上下文切换的时候怎么保证值是最新值（利用ThreadLocalMap），ThreadLocal是干什么的？
3. 隔离数据值2、保证每个变量都可以修改各自变量的副本）

ThreadLocalMap<ThreadLocal,Object>的弱引用问题？  
当线程没有结束而ThreadLocalMap<null,Object>的键值对为空的情况，会导致内存溢出(ThreadLocal被回收，ThreadLocal关联的共享变量还存在)

解决：

1. 使用完线程共享变量之后，调用ThreadLocalMap.remove()方法清除线程共享变量
2. JDK建议将ThreadLocal的设置成private static类型这样ThreadLocal的弱引用问题就不存在了
3. Redis常用命令（get、put）数据类型（String、hash、set、list、zset）
4. Redis的持久化方式

RDB和AOF

RDB原理（将redis内存种的数据库定时记录到磁盘上）；

AOF原理（将redis的操作日志以追加的方式写入到文件中）

1. Kafka、RabbitMQ、RocketMQ、ActiveMQ

Kafka是apache的一个开源子项目是分布式发布-订阅消息系统，不支持事务，对消息的重复、丢失、错误没有严格要求，追求高吞吐量。

RabbitMQ是使用Erlang语言开发的开源消息队列系统，对安全及可靠性要求较高，支持事务，相对于kafka来说对并发量要求不高。

RocketMQ是阿里的开源消息中间件，是kafka和RabbitMQ的结合，支持事务及高并发行及高可用性。

ActiveMQ是apache的一个子项目，类似于ZeroMQ能够实现代理人及点对点的技术实现队列。

1. 数据库（索引、试图）
2. StringBuilder和StringBuffer的区别

StringBuilder和StringBuffer相同点都是字符串可变的；不同点StringBuilder是线程不安全的，适用于单线程下在字符缓冲区进行大量操作的情况；StringBuffer是线程安全的适用于在多线程在字符缓冲区进行大量操作的情况。

1. 索引什么情况下会生效、失效

索引生效：当创建索引字段在查询或者表关联的时候就会生效，可以用explain执行计划来查看

如：explain select \* from a

索引失效：

1. 条件中用or，即使其中有条件带索引，也不会使用索引查询。
2. Like的模糊查询以%开头，索引失效
3. 如果列类型是字符串一定要在查询的时候加上‘ ’不然也会引起索引失效。
4. MySQL预计 使用全表扫描比索引快，则不使用索引
5. Spring种Autowrited和Resource的区别

Autowrited是spring的按照类型装配，Resource是JDK的按照名称装配

Autowrited默认按照类型装配，默认情况下它要求依赖对象必须存在，不存在会NullpointException，如果允许为空，可以设置他的required属性为false，如果我们想使用按照名称装配，可以结合@Qualifier注解一起使用；

Resource默认按照类型装配，当找不到与之名称匹配的bean才会按照类型装配，可以通过name属性指定，如果没有指定name的属性，当注解标注在字段上，及默认按照字段的名称作为bean名称寻找依赖对象，当注解标注在属性的setter方法上，即默认取属性名作为bean名称寻找依赖对象。

@Autowrited字段注入会产生警告，并且建议使用构造方法注入而使用@Resource注入不会有警告。

1. 不建议使用字段注入，因为可能会引起NullPointException异常；也不建议使用setter方法注入，因为不能将属性设置为final，从spring4官方建议使用构造方法注入。但是构造方法注入只能使用@Autowrited注入不能使用@Resource注入。
2. 综上要使用字段注入使用@Resource，要使用构造方法注入使用@Autowrited
3. ==和equals的区别

==基本数据类型的比较是用==比较，比较的是堆的内存地址，及比较的是栈的内容；equals比较的是Object对象及引用型对象的值是否相等及堆的内容是否相等。

1. Mybities的sql注入（$和#的区别）

#会自动转换加‘’或“”

$符号传入啥就是啥

1. JDBC的隔离机制

原子性、一致性、隔离性、持久性

1. Spring中bean的加载和获取

从xml读取bean的配置信息加载到spring容器中，通过xml配置的id从spring容器反射得到这个类的实例对象。

将bean存储到beanDefinitionMap中，然后从beanDefinitionMap中获取bean

1. 事务的4中隔离级别

读未提交(RU)

读已提交(RC)

可重复读(RR)

串行

1. MySQL中的InnoDB和MySam区别（https://blog.csdn.net/u013252072/article/details/52912385）
2. InnoDB支持事务，MySam不支持事务
3. InnoDB支持行锁，MySam支持表锁
4. InnoDB支持外键，MySam不支持外键
5. InnoDB支持MVCC，MySam不支持
6. InnoDB不支持全文检索，MySam支持全文检索
7. JDBC连接数据库的步骤
8. JDBC所需的4个参数(user,password,url,driverClass)
9. 加载JDBC驱动程序
10. 创建数据库的连接
11. 创建一个preparedStatement
12. 执行SQL语句
13. 遍历结果集
14. 处理异常，关闭JDBC对象资源
15. Spring中Bean的5种作用域

在spring中，那些组成应用程序的主体及由spring IOC容器所管理的对象，被称之为bean。简单的讲bean就是IOC容器初始化、装配及管理的对象，除此之外，bean就与应用程序的其他对象没有什么区别。而bean的定义以及bean相互间的依赖关系将通过配置元数据来描述。

Singleton：单利模式，在整个Spring IOC容器中，spring默认的作用域。

Prototype：原型模式

Request

Session

Globalsession

1. ThreadLocal有什么用？

简单的说ThreadLocal就是一种以空间换时间的做法，在每个Thread里面维护了一个以开地址法实现的ThreadLocal.ThreadLocalMap,把数据进行隔离，数据不共享，自然就没有线程安全方面的问题了。

1. 为什么要使用线程池

避免频繁的创建和销毁线程，达到线程对象的重用。另外，使用线程池还可以根据项目灵活的控制并发的数目.

1. 静态变量、实例变量、局部变量与线程安全

静态变量：线程不安全，加static关键字的变量只能放到类里不能放到方法里，静态变量表示所有实例共享的一个属性，位于方法区，共享一份内存，而成员变量是对象的特殊描述，不同对象的实例变量被分配在不同的内存空间，一旦静态变量被修改其他对象均对修改可见，故静态变量是线程不安全的。

实例变量：单例模式（只有一个对象实例存在）线程不安全，非单例模式线程安全。实例变量为对象的实例私有，在虚拟机的堆中分配，若在系统中只存在一个对象的实例，在多线程环境中，被某个线程修改后，其他线程对修改均可见，故线程不安全；如果每个线程执行都是不同的对象中，那对象与对象之间的实例变量的修改互不影响，故线程安全。

局部变量：线程安全

由于每个线程执行时将会把局部变量放在各自栈的工作内存中，线程间不共享，故不存在线程安全问题。

1. 如何使用redis实现限制前台点击

方法：前台创建一个uuid+时间戳传给后台，调用redis的自增方法，将key中存储的数字值增1，如果调用中key不存在，那么先把key进行初始化，在调用redis的INCR方法，如果值包含错误的类型或字符串类型的值不能表示为数字，那么返回一个错误。

1. Integer与int的区别（这个题面试会问接口传参的问题，用Integer应为可以判断是否空传的情况；还有就是整型字面量范围【-128~127】）
2. Integer是int提供的封装类，而int是Java的基本数据类型
3. Integer的默认值是null而int的默认值是0
4. 声明为Integer的变量需要实例化，而声明int的变量不需要实例化
5. Integer是对象，是一个引用指向这个对象，而int是基本类型，直接存储数值
6. 网络7层协议

物理层-数据链路层-网络层-传输层-会话层-表示层-应用层

1. JDK1.5的新特性
2. 自动装箱与拆箱
3. 枚举
4. 静态导入
5. 可变参数
6. 内省
7. 泛型
8. For-Each循环
9. JUC
10. JDK1.8新特性
11. 接口中可以有实现方法（但是实现方法必须加default关键字）
12. **Lambda表达式**
13. **函数式接口与静态导入**
14. Lambda作用域
15. 访问局部变量
16. 访问对象字段与静态变量
17. Date API
18. Java代码实现二叉树(大厂子都会问)
19. 查询有2门及以上不及格科目的学生姓名及其平均成绩

Select sname,avg(grade) from s,sc

Where s.s#=sc.s# and grade<60

Group by sname

Having by count(grade)>=2

1. 如何判断一个链表是否有环

设置两个指针(fast,slow),初始值都指向头，slow每次前进一步，fast每次前进二步，如果链表存在环，则fast必定先进入环，而slow后进入环，两个指针必定相遇。（当然，fast先行头到尾部为NULL，则为无环链表）

1. 判断两个链表是否有交点
2. 直接法（采用暴力破解的方法，遍历两个链表，判断第一个链表的每个节点是否在第二个链表中）
3. Hash计数法（把第一个链表放进hash表，对第二个链表进行hash表的查询）
4. SQL语句如何查询评论表里每个人的最新一条评论信息

Select \* from test group by f,t order by time desc limit 1;

1. 线程池如何创建

Java创建线程池的方式

1、ExecutorService cachedThreadPool = Executors.newCachedThreadPool();

cachedThreadPool.execute(new Runnable(){

public void run(){

System.out.println(“”);

}

});

2、Executor Service fixedThread Pool = Executors.newFixedThreadPool(3);

fixedThreadPool.execute(new Runnable(){

…

})；

3、ScheduledExecutorService scheduledThreadPool = Executors.newScheduledThreadPool(5);

scheduledThreadPool.scheduled(new Runnable(){

…

})；

4、ExecutorService singleThreadExecutor=Executors.newSingleThreadExecutor();

49、springboot如何不用默认的tomcat