事务

1 什么是事务？ （事务相关的概念）

事务就是一组有逻辑操作单元的SQL集合,要不都成功,要么全部失败回滚

1、原子性（Atomicity）

　　原子性是指事务包含的所有操作要么全部成功，要么全部失败回滚，因此事务的操作如果成功就必须要完全应用到数据库，如果操作失败则不能对数据库有任何影响。

2、一致性（Consistency）

　　一致性是指事务必须使数据库从一个一致性状态变换到另一个一致性状态，也就是说一个事务执行之前和执行之后都必须处于一致性状态。

　　拿转账来说，假设用户A和用户B两者的钱加起来一共是5000，那么不管A和B之间如何转账，转几次账，事务结束后两个用户的钱相加起来应该还得是5000，这就是事务的一致性。

3、隔离性（Isolation）

　　隔离性是当多个用户并发访问数据库时，比如操作同一张表时，数据库为每一个用户开启的事务，不能被其他事务的操作所干扰，多个并发事务之间要相互隔离。

　　即要达到这么一种效果：对于任意两个并发的事务T1和T2，在事务T1看来，T2要么在T1开始之前就已经结束，要么在T1结束之后才开始，这样每个事务都感觉不到有其他事务在并发地执行。

4、持久性（Durability）

　　持久性是指一个事务一旦被提交了，那么对数据库中的数据的改变就是永久性的，即便是在数据库系统遇到故障的情况下也不会丢失提交事务的操作。

　　例如我们在使用JDBC操作数据库时，在提交事务方法后，提示用户事务操作完成，当我们程序执行完成直到看到提示后，就可以认定事务以及正确提交，

即使这时候数据库出现了问题，也必须要将我们的事务完全执行完成，否则就会造成我们看到提示事务处理完毕，但是数据库因为故障而没有执行事务的重大错误。

事务的传播特性:

传播特性

1. PROPAGATION\_REQUIRED(瑞快润特)

如果事务不存在则创建，如果存在使用存在的事务

1. PROPAGATION\_SUPPORTS(死普特死)

如果当前存在事务，则使用存在的事务，如果不存在则使用非事务方式处理。

1. PROPAGATION\_MANDATORY(慢的脆)

如果当前存在事务，则使用存在的事务，如果不存在则抛出异常。

4. PROPAGATION\_REQUIRES\_NEW(瑞快润)

永远创建新的事务，如果当前存在事务则挂起

5. PROPAGATION\_NOT\_SUPPORTED

永远使用非事务方式执行，如果当前存在事务则挂起

6. PROPAGATION\_NEVER

不支持事务执行方式，如果存在事务则抛出异常

7. PROPAGATION\_NESTED

如果存在一个活动的事务，则运行在一个嵌套的事务中. 如果没有活动事务, 安按照PROPAGATION\_REQUIRED 事务属性执行

事务的隔离级别:

参考地址:https://blog.csdn.net/zhouym\_/article/details/90381606

第一种隔离级别：Read uncommitted(读未提交)

如果一个事务已经开始写数据，则另外一个事务不允许同时进行写操作，但允许其他事务读此行数据，该隔离级别可以通过“排他写锁”，但是不排斥读线程实现。这样就避免了更新丢失，却可能出现脏读，也就是说事务B读取到了事务A未提交的数据

解决了更新丢失，但还是可能会出现脏读

第二种隔离级别：Read committed(读提交)

如果是一个读事务(线程)，则允许其他事务读写，如果是写事务将会禁止其他事务访问该行数据，该隔离级别避免了脏读，但是可能出现不可重复读。事务A事先读取了数据，事务B紧接着更新了数据，并提交了事务，而事务A再次读取该数据时，数据已经发生了改变。

解决了更新丢失和脏读问题

第三种隔离级别：Repeatable read(可重复读取)

可重复读取是指在一个事务内，多次读同一个数据，在这个事务还没结束时，其他事务不能访问该数据(包括了读写)，这样就可以在同一个事务内两次读到的数据是一样的，因此称为是可重复读隔离级别，读取数据的事务将会禁止写事务(但允许读事务)，写事务则禁止任何其他事务(包括了读写)，这样避免了不可重复读和脏读，但是有时可能会出现幻读。(读取数据的事务)可以通过“共享读镜”和“排他写锁”实现。

解决了更新丢失、脏读、不可重复读、但是还会出现幻读

第四种隔离级别：Serializable(可序化)

提供严格的事务隔离，它要求事务序列化执行，事务只能一个接着一个地执行，但不能并发执行，如果仅仅通过“行级锁”是无法实现序列化的，必须通过其他机制保证新插入的数据不会被执行查询操作的事务访问到。序列化是最高的事务隔离级别，同时代价也是最高的，性能很低，一般很少使用，在该级别下，事务顺序执行，不仅可以避免脏读、不可重复读，还避免了幻读

解决了更新丢失、脏读、不可重复读、幻读(虚读)

以上四种隔离级别最高的是Serializable级别，最低的是Read uncommitted级别，当然级别越高，执行效率就越低，像Serializeble这样的级别，就是以锁表的方式(类似于Java多线程中的锁)使得其他线程只能在锁外等待，所以平时选用何种隔离级别应该根据实际情况来，在MYSQL数据库中默认的隔离级别是Repeatable read（可重复读）。

Aop:

什么是aop？

**aop：面向切面编程是软件编程思想发展到一定阶段的产物，是面向对象编程(OOP)的有益补充。AOP一般适用于具有横切逻辑的场合，如安全控制、事务管理、日志记录、性能统计等。面向切面编程简单地说就是在不改变源程序的基础上为代码段增加新的功能，对代码段进行增强处理。**

**aop的代理通过jdk动态代理，也可以通过cglib实现，默认是通过jdk动态代理实现的。JDK动态代理需要接口的支持，如果没有接口只有类，则使用cglib实现。**

用aop做过哪些?

**安全控制、事务管理、日志记录、性能统计**

**日志记录的实现:**

**我们日志记录用到了注解和反射这两个技术,**

首先我们先创建一个类加上@Component 声明把类交给IOC管理 在加上@Aspect切面类 @Before 前置通知 参数加上一个joinpoint参数

然后构建日志对象 获取对象签名 在获取方法对象 在获取注解 判断注解是否为空 不为空的话我们将数据添加到日志表里

mybatis # $ 区别

分库分表（数据库优化）

反射

对类来说 我能知道你的所有属性和方法

对于对象来说 我能得到所有的属性值和调用你的方法

获取类的三种方式

方法一：直接通过一个class的静态变量class获取：

**Class cls = String.class;**

方法二：如果我们有一个实例变量，可以通过该实例变量提供的getClass()方法获取：

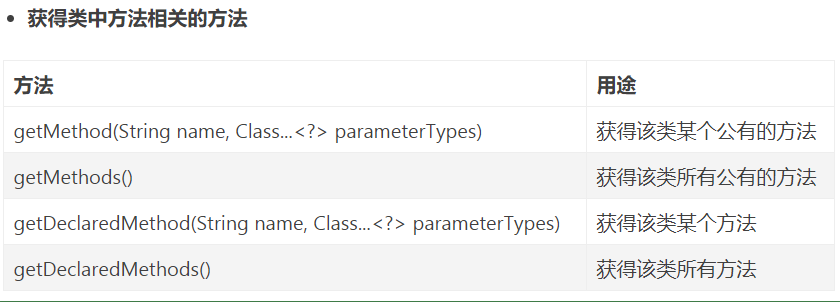
String s = "Hello";

Class cls = s.getClass();

方法三：如果知道一个class的完整类名，可以通过静态方法Class.forName()获取：

Class cls = Class.forName("java.lang.String");





注解

元注解：是java提供的修饰自定义注解的四种注解，叫元注解，

包括以下四种：

@Target:用于指定自定义注解的使用范围。

ElementType.FIELD：应用于全局属性

ElementType.METHOD：应用于方法

ElementType.PARAMETER：应用于方法的参数

ElementType.TYPE：应用于类、接口或者枚举声明

@Retention：用于修饰自定义注解的生存周期，或者可以保留多久

RetentionPolicy.SOURCE：源代码时保留，其他直接丢弃

RetentionPolicy.CLASS：默认值，编译器将把注解记录在class文件中，程序运行时，不会留下

RetentionPolicy.RUNTIME：编译器将把注解记录在class文件中，当运行java程序时，虚拟机保留注解，程序可以通过反射获取该注解；

@Documented：执行javadoc命令时，被该元注解修饰的自定义注解也会生成在文档中

@Inherited：如果父类所使用的注解有@Inherited修饰，则子类可以继承该注解，否则不能继承。