事务和数据库连接池

今日内容介绍

1.事务

2.连接池

1.连接池概念

2.自定义连接池

3.开源连接池(c3p0)

复习：

1. 使用 jdbc 来操作事务

开启事务， 操作 sql 语句。 关闭事务（commit， rollback）

1. ThreadLocal 类 是什么 ？ 有 什么 功能， 有 什么 方法 ？

Get，set 。

1. 丢失更新，什么 ？
2. 悲观锁， 乐观锁 ？
3. 悲观锁，
4. 共享锁， 排他锁 ？
5. 悲观锁，乐观锁 的区别 。

以前开发， 主要使用 悲观锁。 如果一个 引用 是 高并发引用 。乐观锁 是 肯定 要 比悲观锁 。

1. 数据库连接池的定义 ？
2. 池资源模式 （java 的设计模式）?
3. 自定义 连接池 。
4. 连接 Connection , close（）， 进行加强。
5. 装饰者模式， 修饰模式（ Java 设计模式）对 以前的类进行 增强 。

首先 必须 有一个接口， 接口定义类的实现方法。

装饰者 和 被装饰者。 都是 一个 类。

被装饰者 和 装饰者都 实现同一接口。

装饰者持有 被装饰者的 对象 。装饰者 实现的方法实际上是 调用 被装饰者 对应的方法，并且实现了 加强 。代理模式

1. 动态代理，
2. DBCP， C3P0 的 区别 ？ tomcat 带连接池 ？现在很多开源项目 都是 c3p0.
3. 其他

# 复习:

1.jdbc是什么，它有什么用?

java database connection

让java程序员可以直接通过java程序操作数据.

jdbc是标准，它是由类与接口组成，对于程序员只需要知道标准(Connection Statement PreparedStatement,ResultSet)

不需要了解具休实现就可以操作数据库。

2.jdbc入门代码

导入jar包

1.注册驱动

DriverManager.registDriver(Driver dirver);

不使用这种方式，使用反射

Class.forName("com.mysql.jdbc.Driver");

2.获取连接Connection

Connection con=DriverManager.getConnection(String url,String username,String password);

3.获取操作sql对象 Statement

Statement st=con.createStatement();

如果要得到滚动结果集，可以使用createStatement(int,int);

4.操作sql

DQL语句

ResultSet rs=st.executeQuery(String sql);

DML语句

int row=st.executeUpdate(String sql);

5.遍历结果集

while(rs.next()){

rs.getInt(int coulumnIndex);

rs.getString(String columnName);

}

6.释放资源

rs.close();

st.close();

con.close();

----------------------------------------------

PreparedStatement

1.怎样获取?

PreparedStatement pst=con.prepareStatement(String sql);

2.怎样给占位符"?"赋值

pst.setXxx(int,value)

3.执行

pst.executeQuery();

pst.executeUpdate();

复习：

1. 大数据（blob， clob：text）

了解一下。可以做什么 ？

Blob： 存到 数据库里面 。

如何做 ？

Pstmt.setBinaryStream(1, fis);

Pstmt.executeUpdate();

Rs.getBinaryStream(“content”);

1. Jdbc的批处理

其实 2 种方式 ？

Statement，： 各种 sql 语句都可以

PreparedStatement： 用于处理 sql 相同的 批命令 。

批处理一般 处理 dml 语句 ？

几个常见的方法： statement 已经 创建：

Stmt.addBatch(sql);

executeBatch();

clearBatch();

setString(index, “aaz”);

Pstmt.addBatch()

2 种方式。

1. 什么是 事务 ？
2. 事务有 什么 用 ？
3. 事务 的特点 ？

持久、原子性、一致性、隔离性

1. 事务的隔离级别 （ 面试题 ）

Serializable 串行化

Repeatable read ： 可重复读

Read committed ： 可以读到 提交数据。解决 脏读

Read uncommitted： 3 个问题都没有解决

1. 隔离性出现的问题：

脏读：

不可重复读：（update）

虚读、幻读： （insert）

1. 事务是 如何操作：

Jdbc 下面 事务是如何操作：

Connection：

setAutoCommit(false); start transaction

Dml

Commit(), rollback();,

事务出现问题，执行 rollback（）；

Commit（）

开启事务：

执行我们的 sql 语句 组（ 处理 业务流程）

Commit or rollback 提交 或 回滚

1. 其他

# 事务:

## 问题:事务是什么，有什么用?

事务就是一个事情，组成这个事情可能有多个单元，要求这些单元，要么全都成功，要么全都不成功。

在开发中，有事务的存在，可以保证数据完整性。

## 问题:事务怎样操作

创建表:

create table account(

id int primary key auto\_increment,

name varchar(20),

money double

);

insert into account values(null,'aaa',1000);

insert into account values(null,'bbb',1000);

insert into account values(null,'ccc',1000);

### mysql下怎样操作

方式1:

start transaction 开启事务

rollback 事务回滚

commit 事务提交

方式2:

show variables like '%commit%'; 可以查看当前autocommit值

在mysql数据库中它的默认值是"on"代表自动事务.

自动事务的意义就是:执行任意一条sql语句都会自动提交事务.

测试:将autocommit的值设置为off

1.set autocommit=off 关闭自动事务。

2.必须手动commit才可以将事务提交。

注意:mysql默认autocommit=on oracle默认的autocommit=off;

银行转账的例子，说明

### jdbc下怎样操作

java.sql.Connection接口中有几个方法是用于可以操作事务

1.setAutocommit(boolean flag);

如果flag=false;它就相当于start transaction;

2.rollBack()

事务回滚。

3.commit()

事务提交

* 当Jdbc程序向数据库获得一个Connection对象时，默认情况下这个Connection对象会自动向数据库提交在它上面发送的SQL语句。若想关闭这种默认提交方式，让多条SQL在一个事务中执行，可使用下列语句：
* JDBC控制事务语句
  + Connection.setAutoCommit(false); // 相当于start transaction
  + Connection.rollback(); rollback
  + Connection.commit(); commit
* 示例 ***演示银行转帐案例 （银行转账）***
* 在JDBC代码中使如下转帐操作在同一事务中执行。

update from account set money=money-100 where name=‘a’;

update from account set money=money+100 where name=‘b’;

* 设置事务回滚点
  + Savepoint sp = conn.setSavepoint();
  + Conn.rollback(sp);
  + Conn.commit(); //回滚后必须要提交
* 例子书写步骤：

写一个转账的操作 jdbc

加上事务

加上异常处理

搞点异常

## 事务特性(重点) ACID （面试题）

? 原子性（Atomicity）

原子性是指事务是一个不可分割的工作单位，事务中的操作要么都发生，要么都不发生。

? 一致性（Consistency）

事务前后数据的完整性必须保持一致。

? 隔离性（Isolation）（ 重点， 面试很多题）

事务的隔离性是指多个用户并发访问数据库时，一个用户的事务不能被其它用户的事务所干扰，多个并发事务之间数据要相互隔离。

? 持久性（Durability）

持久性是指一个事务一旦被提交，它对数据库中数据的改变就是永久性的，接下来即使数据库发生故障也不应该对其有任何影响

## 如果不考虑事务的隔离性，会出现什么问题?

1、脏读 ：指一个事务读取另一个事务未提交的数据

A 转账 给B 100，未提交

B 查询账户多了100

A 回滚

B 查询账户那100不见了

2、不可重复读：在一个事务先后两次读取发生数据不一致情况，第二次读取到另一个事务已经提交数据 （强调数据更新 update）

A 查询账户 5000

B 向 A 账户转入 5000

A 查询账户 10000

3、虚读(幻读) ： 在一个事务中，第二次读取发生数据记录数的不同 ，读取到另一个事务已经提交数据 （强调数据记录变化 insert ）

A 第一次读取 存在5条记录

B 向 A 插入一条新的记录

A 第二次读取 存在6条记录

4.丢失更新

两个事务对同一条记录进行操作，后提交的事务，将先提交的事务的修改覆盖了。

-----------------------------------------

演示以上问题，以及问题的解决方案

## 事务的隔离级别

对于以上的问题，我们可以通过设置事务的隔离级别来解决。

### 1.事务的隔离级别有哪些?

1 Serializable：可避免脏读、不可重复读、虚读情况的发生。（串行化）

2 Repeatable read：可避免脏读、不可重复读情况的发生。（可重复读）不可以避免虚读

3 Read committed：可避免脏读情况发生（读已提交）

4 Read uncommitted：最低级别，以上情况均无法保证。(读未提交)

### 怎样设置事务的隔离级别?

1.mysql中设置

1.查看事务隔离级别

select @@tx\_isolation 查询当前事务隔离级别

mysql中默认的事务隔离级别是 Repeatable read.

扩展:oracle 中默认的事务隔离级别是 Read committed

2.mysql中怎样设置事务隔离级别

set session transaction isolation level 设置事务隔离级别

2.jdbc中设置

在jdbc中设置事务隔离级别

使用java.sql.Connection接口中提供的方法

void setTransactionIsolation(int level) throws SQLException

参数level可以取以下值:

level - 以下 Connection 常量之一：

Connection.TRANSACTION\_READ\_UNCOMMITTED、

Connection.TRANSACTION\_READ\_COMMITTED、

Connection.TRANSACTION\_REPEATABLE\_READ

Connection.TRANSACTION\_SERIALIZABLE。

（注意，不能使用 Connection.TRANSACTION\_NONE，因为它指定了不受支持的事务。）

--------------------------------------------------

### 3.演示

1.脏读

一个事务读取到另一个事务的为提交数据

设置A,B事务隔离级别为 Read uncommitted

set session transaction isolation level read uncommitted;

1.在A事务中

start transaction;

update account set money=money-500 where name='aaa';

update account set money=money+500 where name='bbb';

2.在B事务中

start transaction;

select \* from account;

这时，B事务读取时，会发现，钱已经汇完。那么就出现了脏读。

当A事务提交前，执行rollback，在commit， B事务在查询，就会发现，钱恢复成原样

也出现了两次查询结果不一致问题，出现了不可重复读.

2.解决脏读问题

将事务的隔离级别设置为 read committed来解决脏读

设置A,B事务隔离级别为 Read committed

set session transaction isolation level read committed;

1.在A事务中

start transaction;

update account set money=money-500 where name='aaa';

update account set money=money+500 where name='bbb';

2.在B事务中

start transaction;

select \* from account;

这时B事务中，读取信息时，是不能读到A事务未提交的数据的，也就解决了脏读。

让A事务，提交数据 commit;

这时，在查询，这次结果与上一次查询结果又不一样了，还存在不可重复读。

3.解决不可重复读

将事务的隔离级别设置为Repeatable read来解决不可重复读。

设置A,B事务隔离级别为 Repeatable read;

set session transaction isolation level Repeatable read;

1.在A事务中

start transaction;

update account set money=money-500 where name='aaa';

update account set money=money+500 where name='bbb';

2.在B事务中

start transaction;

select \* from account;

当A事务提交后commit;B事务在查询，与上次查询结果一致，解决了不可重复读。

4.设置事务隔离级别 Serializable ,它可以解决所有问题

set session transaction isolation level Serializable;

如果设置成这种隔离级别，那么会出现锁表。也就是说，一个事务在对表进行操作时，

其它事务操作不了。

--------------------------------------------------

总结:

脏读:一个事务读取到另一个事务为提交数据

不可重复读:两次读取数据不一致(读提交数据)---update

虚读:两次读取数据不一致(读提交数据)----insert

事务隔离级别:

read uncommitted 什么问题也解决不了.

read committed 可以解决脏读，其它解决不了.

Repeatable read 可以解决脏读，可以解决不可重复读,不能解决虚读.

Serializable 它会锁表，可以解决所有问题， 一次只能处理一个 事务.

安全性：serializable > repeatable read > read committed > read uncommitted

性能 ：serializable < repeatable read < read committed < read uncommitted

结论： 实际开发中，通常不会选择 serializable 和 read uncommitted ，

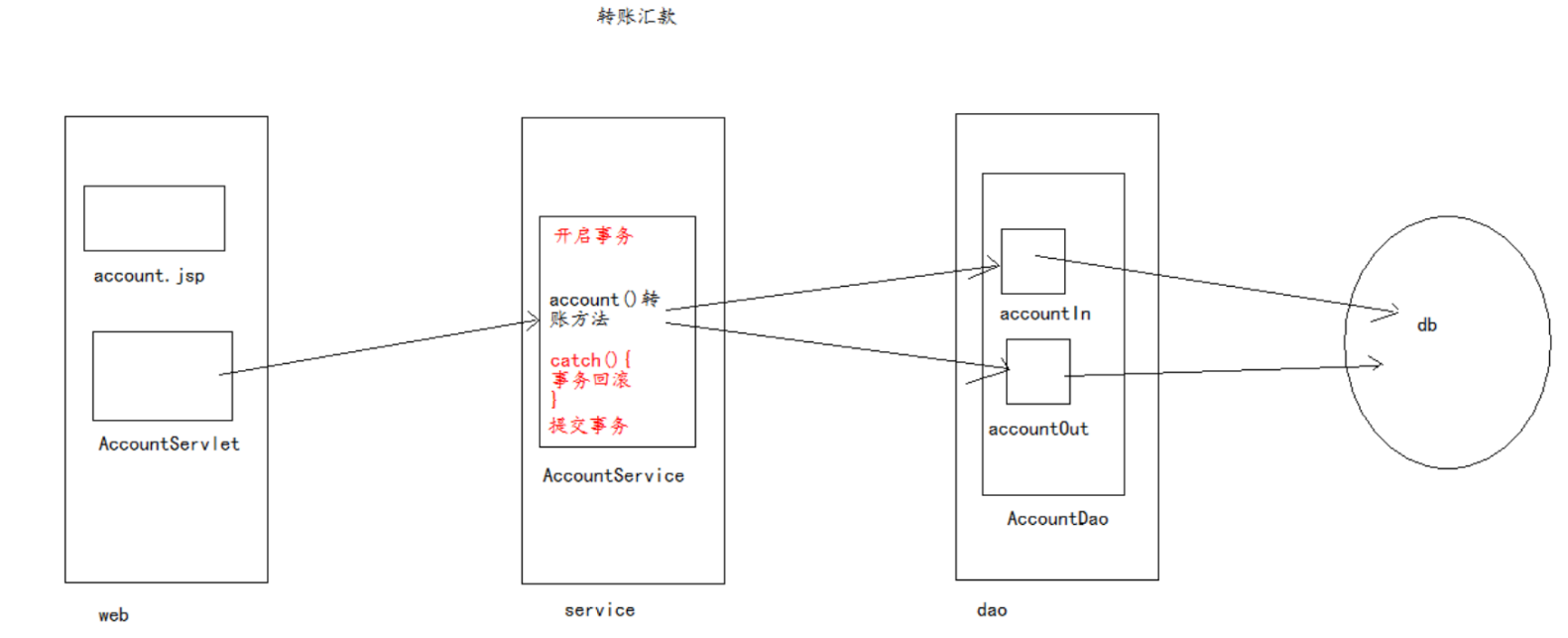
mysql默认隔离级别 repeatable read ，oracle默认隔离级别 read committed

==========================================================================================

## 案例:转账汇款----使用事务

### 正常流程

写一个 web -> service -> dao 的完整例子



问题:service调用了dao中两个方法完成了一个业务操作，如果其中一个方法执行失败怎样办?

需要事务控制

问题:怎样进行事务控制?

我们在service层进行事务的开启，回滚以及提交操作。

问题:进行事务操作需要使用Connection对象，那么，怎样保证，在service中与dao中所使用的是同一个Connection.

在service层创建出Connection对象，将这个对象传递到dao层.

注意:Connecton对象使用完成后，在service层的finally中关闭

而每一个PreparedStatement它们在dao层的方法中用完就关闭.

关于程序问题

1.对于转入与转出操作，我们需要判断是否成功，如果失败了，可以通过抛出自定义异常在servlet中判断，

进行信息展示 。

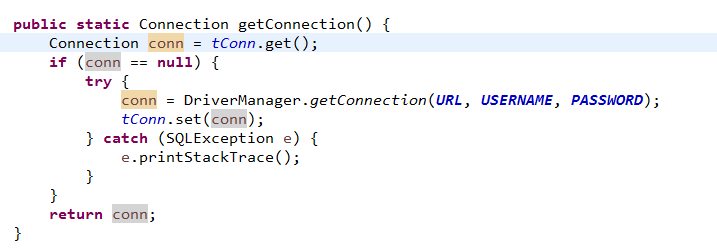
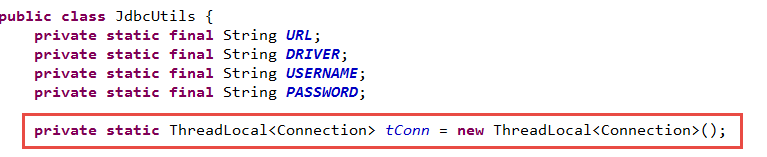
需要对 sql语句的操作 是否成功进行判断， 如果操作 不成功， 要考虑处理

----------------------------------------------------------

### 使用 threadlocal

问题:

在设置dao层时，



那么我们自己去实现这个接口时，怎样处理，同一个Connection对象问题?

使用ThreadLocal

ThreadLocal可以理解成是一个Map集合

Map<Thread,Object>

set方法是向ThreadLocal中存储数据，那么当前的key值就是当前线程对象.

get方法是从ThreadLocal中获取数据，它是根据当前线程对象来获取值。

如果我们是在同一个线程中，只要在任意的一个位置存储了数据，在其它位置上，就可以获取到这个数据。

关于JdbcUtils中使用ThreadLocal

1.声明一个ThreadLocal

private static final ThreadLocal<Connection> tl = new ThreadLocal<Connection>();

2.在getConnection()方法中操作

Connection con = tl.get(); 直接从ThreadLocal中获取，第一次返回的是null.

if (con == null) {

// 2.获取连接

con = DriverManager.getConnection(URL, USERNAME, PASSWORD);

tl.set(con); //将con装入到ThreadLocal中。

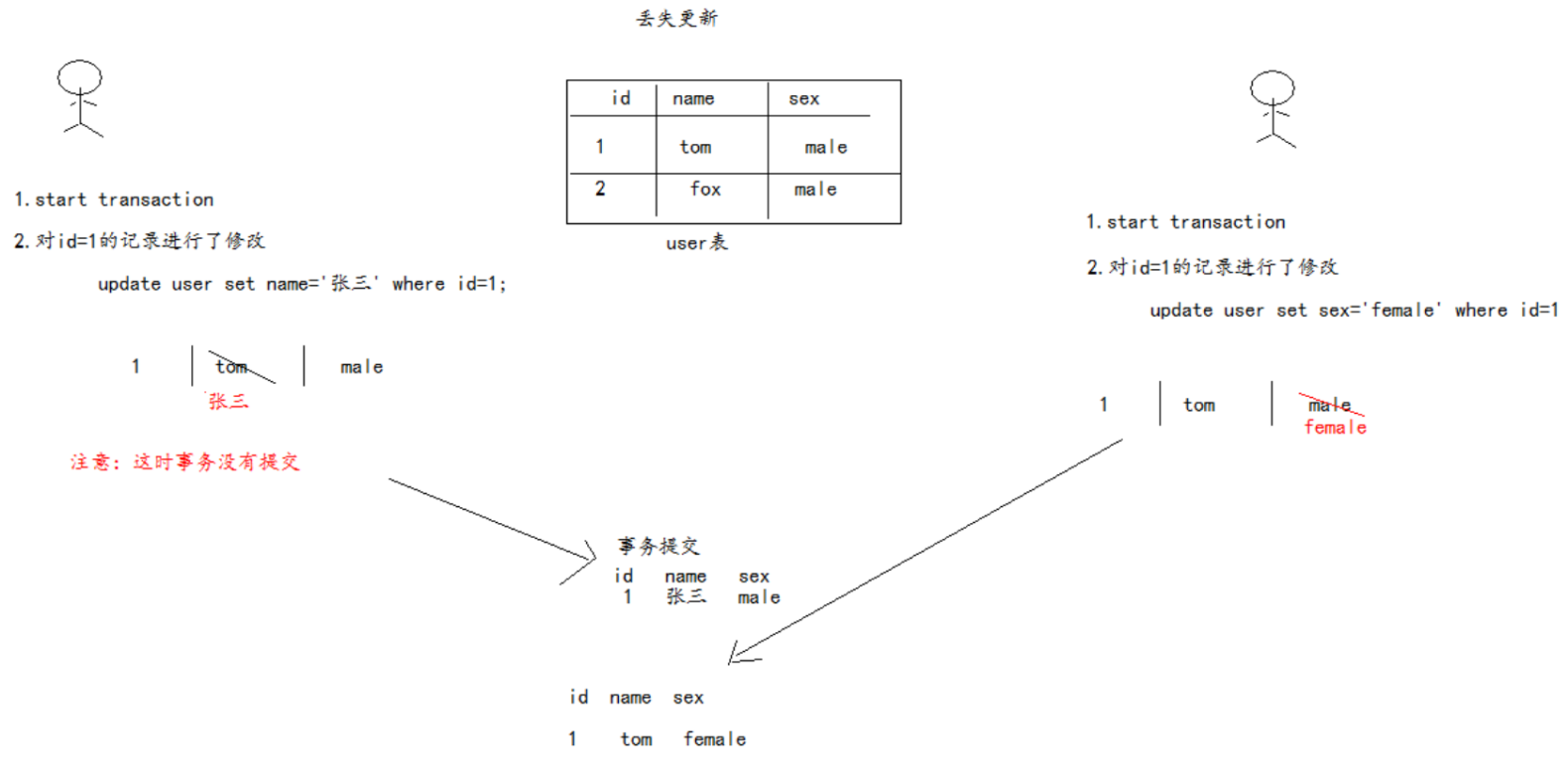
}

## 丢失更新 (工作中实用性很强，并发性）

### 什么是丢失更新

多个事务对同一条记录进行了操作，后提交的事务将先提交的事务操作覆盖了。

查看图.



问题：怎样解决丢失更新问题?

解决丢失更新可以采用两种方式:

### 悲观锁

悲观锁 （假设丢失更新一定会发生 ） ----- 利用数据库内部锁机制，管理事务

提供的锁机制

1.共享锁

select \* from table lock in share mode（读锁、共享锁）锁

2.排它锁

select \* from table where id = 1 for update （写锁、排它锁）

update语句默认添加排它锁

#### 共享锁：（目的， 修改数据）

又称为读锁，简称S锁，顾名思义，共享锁就是多个事务对于同一数据可以共享一把锁，都能访问到数据，但是只能读不能修改，加锁的事务可以进行修改，其他事务只能读不能修改，其他事务还可以加 共享锁， 如果 有 多把共享锁， 大家都 不能修改 。

#### 排他锁：

又称为写锁，是行级锁。简称X锁，顾名思义，排他锁就是不能与其他锁并存，如一个事务获取了一个数据行的排他锁，其他事务就不能再获取该行的其他锁，包括共享锁和排他锁，但是获取排他锁的事务是可以对数据就行读取和修改。在 mysql 里面，修改语句缺省加排他锁

#### 两种锁的理解

对于共享锁大家可能很好理解，就是多个事务只能读数据不能改数据，对于排他锁大家的理解可能就有些差别，我当初就犯了一个错误，以为排他锁锁住一行数据后，其他事务就不能读取和修改该行数据，其实不是这样的。排他锁指的是一个事务在一行数据加上排他锁后，其他事务不能再在其上加其他的锁。mysql InnoDB引擎默认的修改数据语句，update,delete,insert都会自动给涉及到的数据加上排他锁，select语句默认不会加任何锁类型，如果加排他锁可以使用select ...for update语句，加共享锁可以使用select ... lock in share mode语句。所以加过排他锁的数据行在其他事务种是不能修改数据的，也不能通过for update和lock in share mode锁的方式查询数据，但可以直接通过select ...from...查询数据，因为普通查询没有任何锁机制。

### 乐观锁

乐观锁 （假设丢失更新不会发生）------- 采用程序中添加版本字段解决丢失更新问题

create table product (

id int,

name varchar(20),

updatetime timestamp

);

insert into product values(1,'冰箱',null);

update product set name='洗衣机' where id = 1;

解决丢失更新：在数据表添加版本字段，每次修改过记录后，版本字段都会更新，如果读取是版本字段，

与修改时版本字段不一致，说明别人进行修改过数据 （重改）

乐观锁（ Optimistic Locking ） 相对[悲观锁](https://baike.baidu.com/item/%E6%82%B2%E8%A7%82%E9%94%81" \t "_blank)而言，乐观锁机制采取了更加宽松的加锁机制。悲观锁大多数情况下依靠数据库的锁机制实现，以保证操作最大程度的独占性。但随之而来的就是数据库性能的大量开销，特别是对长[事务](https://baike.baidu.com/item/%E4%BA%8B%E5%8A%A1" \t "_blank)而言，这样的开销往往无法承受。而乐观锁机制在一定程度上解决了这个问题。乐观锁，大多是基于数据版本（ Version ）记录机制实现。何谓数据版本？即为数据增加一个版本标识，在基于数据库表的版本解决方案中，一般是通过为数据库表增加一个 “version” 字段来实现。读取出数据时，将此版本号一同读出，之后更新时，对此版本号加一。此时，将提交数据的版本数据与数据库表对应记录的当前版本信息进行比对，如果提交的数据版本号大于数据库表当前版本号，则予以更新，否则认为是过期数据。

#### 例子：

读取出数据时，将此版本号一同读出，之后更新时，对此版本号加一。此时，将提交数据的版本数据与数据库表对应记录的当前版本信息进行比对，如果提交的数据版本号大于数据库表当前版本号，则予以更新，否则认为是过期数据。

Update set num = 200 where id = 1 and version < 3;

对于上面修改用户帐户信息的例子而言，假设数据库中帐户信息表中有一个 version 字段，当前值为 1 ；而当前帐户余额字段（ balance ）为 $100 。

1 操作员 A 此时将其读出（ version=1 ），并从其帐户余额中扣除 $50（ $100-$50 ）。

2 在操作员 A 操作的过程中，操作员B 也读入此用户信息（ version=1 ），并从其帐户余额中扣除 $20 （ $100-$20 ）。

3 操作员 A 完成了修改工作，将数据版本号加一（ version=2 ），连同帐户扣除后余额（ balance=$50 ），提交至数据库更新，此时由于提交数据版本大于数据库记录当前版本，数据被更新，数据库记录 version 更新为 2 。

4 操作员 B 完成了操作，也将版本号加一（ version=2 ）试图向数据库提交数据（ balance=$80 ），但此时比对数据库记录版本时发现，操作员 B 提交的数据版本号为 2 ，数据库记录当前版本也为 2 ，不满足 “ 提交版本必须大于记录当前版本才能执行更新 “ 的乐观锁策略，因此，操作员 B 的提交被驳回。

这样，就避免了操作员 B 用基于 version=1 的旧[数据修改](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E4%BF%AE%E6%94%B9" \t "_blank)的结果覆盖操作员A 的操作结果的可能。

# 连接池

## 问题：连接池是什么，有什么用?

连接池:就是创建一个容器，用于装入多个Connection对象，在使用连接对象时，从容器中获取一个Connection，

使用完成后，在将这个Connection重新装入到容器中。这个容器就是连接池。(DataSource)

也叫做数据源.

我们可以通过连接池获取连接对象.

优点:

节省创建连接与释放连接 性能消耗 ---- 连接池中连接起到复用的作用 ，提高程序性能

-----------------------------------------------------------------------------------

## 自定义连接池

1.创建一个MyDataSource类，在这个类中创建一个LinkedList<Connection>

2.在其构造方法中初始化List集合，并向其中装入5个Connection对象。

3.创建一个public Connection getConnection();从List集合中获取一个连接对象返回.

4.创建一个 public void readd(Connection) 这个方法是将使用完成后的Connection对象重新装入到List集合中.

代码问题:

1.连接池的创建是有标准的.

在javax.sql包下定义了一个接口 DataSource

简单说，所有的连接池必须实现javax.sql.DataSource接口，

我们的自定义连接池必须实现DataSource接口。

2.我们操作时，要使用标准，怎样可以让 con.close()它不是销毁，而是将其重新装入到连接池.

要解决这个问题，其本质就是将Connection中的close()方法的行为改变。

怎样可以改变一个方法的行为(对方法功能进行增强)

1.继承

2.装饰模式

1.装饰类与被装饰类要实现同一个接口或继承同一个父类

2.在装饰类中持有一个被装饰类引用

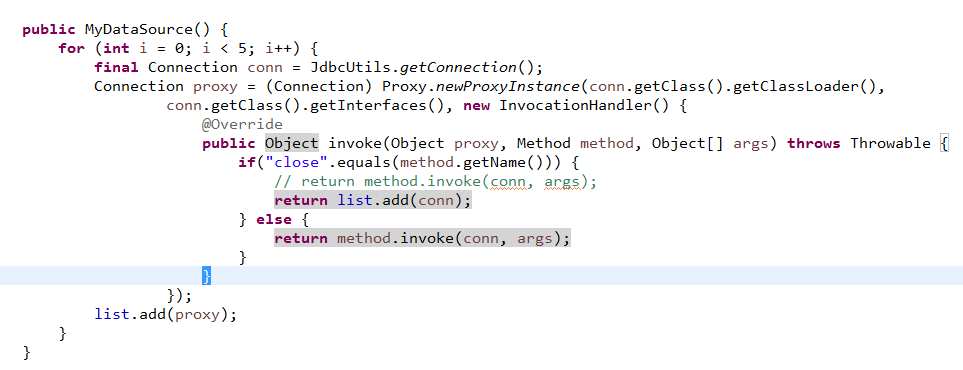
3.对方法进行功能增强。

3.动态代理

可以对行为增强

Proxy.newProxyInstance(ClassLoacer ,Class[],InvocationHandler);

结论:Connection对象如果是从连接池中获取到的，那么它的close方法的行为已经改变了，不在是销毁，而是重新装入到连接池。



--------------------------------------------------------------------

1.连接池必须实现javax.sql.DataSource接口。

2.要通过连接池获取连接对象 DataSource接口中有一个 getConnection方法.

3.将Connection重新装入到连接池 使用Connection的close()方法。

开源连接池

## 1.dbcp(了解)

dbcp是apache的一个开源连接池。

要想使用DBCP连接池，要下载jar包

导入时要导入两个

commons-dbcp-1.4.jar

commons-pool-1.5.6.jar

关于dbcp连接池使用

1.手动配置(手动编码)

BasicDataSource bds = new BasicDataSource();

// 需要设置连接数据库最基本四个条件

bds.setDriverClassName("com.mysql.jdbc.Driver");

bds.setUrl("jdbc:mysql:///day18");

bds.setUsername("root");

bds.setPassword("abc");

// 得到一个Connection

Connection con = bds.getConnection();

2.自动配置(使用配置文件)

Properties props = new Properties();

FileInputStream fis = new FileInputStream("D:\\java1110\\workspace\\day18\_2\\src\\dbcp.properties");

props.load(fis);

DataSource ds = BasicDataSourceFactory.createDataSource(props);

## 2.c3p0(必会)

C3P0是一个开源的JDBC连接池，它实现了数据源和JNDI绑定，支持JDBC3规范和JDBC2的标准扩展。

目前使用它的开源项目有Hibernate，Spring等。

c3p0与dbcp区别

dbcp没有自动回收空闲连接的功能

c3p0有自动回收空闲连接功能

c3p0连接池使用

1.导包

c3p0-0.9.1.2.jar

使用

1.手动

ComboPooledDataSource cpds = new ComboPooledDataSource();

cpds.setDriverClass("com.mysql.jdbc.Driver");

cpds.setJdbcUrl("jdbc:mysql:///day18");

cpds.setUser("root");

cpds.setPassword("abc");

1. 自动(使用配置文件)

By default, c3p0 will look for an XML configuration file in its classloader's resource path under the name "/c3p0-config.xml". That means the XML file should be placed in a directly or jar file directly named in your applications CLASSPATH, in WEB-INF/classes, or some similar location.

c3p0的配置文件可以是properties也可以是xml.

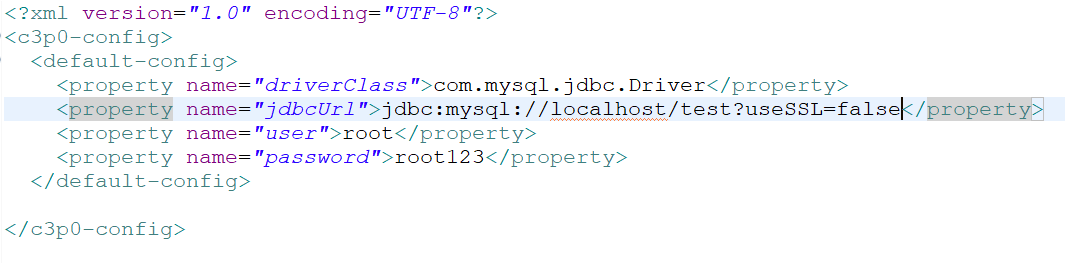


c3p0的配置文件如果名称叫做 c3p0.properties or c3p0-config.xml 并且放置在classpath路径下(对于web应用就是classes目录)

那么c3p0会自动查找。

注意：我们其时只需要将配置文件放置在src下就可以。

c3p0-config.xml 文件内容：



使用：

ComboPooledDataSource cpds = new ComboPooledDataSource();

它会在指定的目录下查找指定名称的配置文件，并将其中内容加载。