通过代理模式来保持用户关闭连接的习惯

1. **数据库连接池**：可以获取连接**Connection对象**，但是由于操作习惯，可能习惯地用完连接后，会调用close方法，从而把此连接关闭掉。为了保持这个习惯，且调用close方法时，执行的是把此连接放回**连接池**。
2. 方法：创建一个新类，去实现接口**Connection**，然后把**close方法**覆盖掉。**用MyConnection类去代理Connection，这就是代理模式。**有两个属性，一个是真正的Connection对象realConn，一个是数据库连接池。

private Connection realConn;

private ConnectionPool conns;

realConn的作用是**调用Connection的方法**来实现新的类的方法的重写，其实是没有重写，如

@Override

public void setAutoCommit(boolean autoCommit) throws SQLException {

**this.realConn.setAutoCommit(autoCommit);**

}

其他方法都是利用这种假重写的方法进行重写方法。

在这里，仅仅只是对**close方法**实现了真正意义上的重写，如

@Override

**public** **void** close() **throws** SQLException {

**this.conns.connPool.addLast(this);**

}

1. ConnectionPool类的修改：

public class ConnectionPool {

private static int maxCount = 9;

private static int currentCount = 0;

public LinkedList<Connection> connPool = new LinkedList<Connection>();

public ConnectionPool() {

for (int i = 0; i < 5; i++) {

**connPool.addLast(this.creatNewConn());//创建的是MyConnection的对象**

this.currentCount++;

}

}

public Connection getConnection() throws SQLException {

synchronized (connPool) {

if (connPool.size() > 0)

return connPool.removeFirst();

else if (this.currentCount < this.maxCount) {

this.currentCount++;

return this.creatNewConn();

}

else

throw new SQLException("已径达到了最大的限度。");

}

}

public void free(Connection conn) {

connPool.addLast(conn);

}

public Connection creatNewConn() {

Connection realConn = DriverUtil.getConnection();//这个是真正的Connection对象realConn。

**MyConnection myConn = new MyConnection(realConn, this);//创建一个Connection代理，对realConn进行了包装，实质上只改变了close方法，其他的都没有发生改变。**

**真正意义上的连接也就是一个：realConn，myConn就像是给realConn穿上的一件衣服。这就是代理模式。**

**return myConn;**

}

}

1. 对此进行进一步的改进：

**对每一个连接的使用次数进行限制，当一个连接的使用次数超过了一定的限度，就真正的关闭掉。**

**MyConnection类：**

**private Connection realConn;**

**private ConnectionPool conns;**

**private int maxUseCount = 3;**

**private int currentUseCount = 0;**

**@Override**

**public void close() throws SQLException {**

**this.currentUseCount++;**

**if (this.currentUseCount < this.maxUseCount) {**

**this.conns.connPool.addLast(this);**

**}**

**else {**

**this.realConn.close();**

**this.conns.currentCount--;**

**} }**

**其他的都不变。**

**ConnectionPool：没有修改。**

**public class ConnectionPool {**

**private int maxCount = 8;**

**int currentCount = 0;**

**public LinkedList<Connection> connPool = new LinkedList<Connection>();**

**public ConnectionPool() {**

**for (int i = 0; i < 5; i++) {**

**connPool.addLast(this.creatNewConn());**

**this.currentCount++;**

**}**

**}**

**public Connection getConnection() throws SQLException {**

**synchronized (connPool) {**

**if (connPool.size() > 0)**

**return connPool.removeFirst();**

**else if (this.currentCount < this.maxCount) {**

**this.currentCount++;**

**return this.creatNewConn();**

**}**

**else**

**throw new SQLException("已径达到了最大的限度。");**

**} }**

**public void free(Connection conn) {**

**connPool.addLast(conn);**

**}**

**public Connection creatNewConn() {**

**Connection realConn = DriverUtil.getConnection();**

**MyConnection myConn = new MyConnection(realConn, this);**

**return myConn;**

**} }**

**Test类：**

**public static void main(String[] args) throws SQLException {**

**ConnectionPool connpool = new ConnectionPool();**

**for (int i = 0; i < 29; i++) {**

**Connection connection = connpool.getConnection();**

**System.out.println(connection);**

**connection.close();**

**} }**

**这样，一个连接使用一定次数后，就关闭掉，连接池在需要的时候，就可以重新创建新的连接。**