# JDBC连接池&DBUtils

## 使用连接池改造JDBC的工具类：

### 需求：

传统JDBC的操作,对连接的对象销毁不是特别好.每次创建和销毁连接都是需要花费时间.可以使用连接池优化的程序.

\* 在程序开始的时候,可以创建几个连接,将连接放入到连接池中.用户使用连接的时候,可以从连接池中进行获取.用完之后,可以将连接归还连接池.

### 分析:

#### 技术分析：

【自定义连接池】（了解）

\* SUN公司提供了一个连接池的接口.(javax.sql.DataSource).

\* 定义一个连接池:实现这个接口.

\* 使用List集合存放多个连接的对象.

【自定义连接池的代码】

public class MyDataSource implements DataSource{

// 创建一个List集合用于存放多个连接对象.

private List<Connection> list = new ArrayList<Connection>();

// 在程序开始的时候，初始化几个连接,将连接存放到list中.

public MyDataSource() {

// 初始化3个连接:

for(int i=1;i<=3;i++){

Connection conn = JDBCUtils.getConnection();

list.add(conn);

}

}

@Override

// 获得连接的方法：

public Connection getConnection() throws SQLException {

if(list.size() <= 0){

for(int i=1;i<=3;i++){

Connection conn = JDBCUtils.getConnection();

list.add(conn);

}

}

Connection conn = list.remove(0);

return conn;

}

// 归还连接的方法：

public void addBack(Connection conn){

list.add(conn);

}

...

}

【自定义连接池中问题及如何解决】

* **问题?**

1. 如果使用自定义连接池,那么需要额外记住自定义连接池中的API.
2. 能不能使用面向接口的编程方式.

* **解决：**

**不额外提供API方法，就可以解决上述两个问题！！！**

能不能还调用Connection的close方法.能不能增强Connection的close方法,原有的销毁变为归还!!!

* **如何增强Connection的close方法:**

**\* 增强一个Java类中的某个方法有几种方式???**

**\* 一种方式：继承的方式.**

**\* 能够控制这个类的构造的时候,才可以使用继承.**

**\* 二种方式：装饰者模式方式.**

**\* 包装对象和被包装的对象都要实现相同的接口.**

**\* 包装的对象中需要获得到被包装对象的引用.**

**\*\*\*\*\* 缺点：如果接口的方法比较多,增强其中的某个方法.其他的功能的方法需要原有调用.**

**\* 三种方式：动态代理的方式.**

**\* 被增强的对象实现接口就可以.**

【继承和装饰者的案例】

/\*\*

\* 继承的方式增强一个类中某个方法:

\*/

class Man{

public void run(){

System.out.println("跑....");

}

}

class SuperMan extends Man{

public void run(){

// super.run();

System.out.println("飞....");

}

}

/\*\*

\* 使用装饰者的方式完成类的方法的增强

\*/

interface Waiter{

public void server();

}

class Waiteress implements Waiter{

@Override

public void server() {

System.out.println("服务...");

}

}

class WaiteressWrapper implements Waiter{

private Waiter waiter;

public WaiteressWrapper(Waiter waiter) {

this.waiter = waiter;

}

@Override

public void server() {

System.out.println("微笑...");

// this.waiter.server();

}

}

【使用装饰者模式增强Connection的close方法】

public class MyConnection implements Connection{

private Connection conn;

private List<Connection> list;

public MyConnection(Connection conn,List<Connection> list) {

this.conn = conn;

this.list = list;

}

@Override

public void close() throws SQLException {

list.add(conn);

}  
 ...

}

连接池的getConnection方法：  
 @Override

// 获得连接的方法：

public Connection getConnection() throws SQLException {

if(list.size() <= 0){

for(int i=1;i<=3;i++){

Connection conn = JDBCUtils.getConnection();

list.add(conn);

}

}

Connection conn = list.remove(0);

**MyConnection myConn = new MyConnection(conn, list);**

return myConn;

}

【常见的开源的数据库连接池】：

* DBCP：

DBCP(DataBase connection pool),[数据库连接池](http://baike.baidu.com/view/84055.htm)。是 apache 上的一个 java 连接池项目，也是 tomcat 使用的连接池组件。单独使用dbcp需要2个包：commons-dbcp.jar,commons-pool.jar由于建立数据库连接是一个非常耗时[耗资源](http://baike.baidu.com/view/630544.htm)的行为，所以通过连接池预先同数据库建立一些连接，放在内存中，应用程序需要建立数据库连接时直接到连接池中申请一个就行，用完后再放回去。

* C3P0：

C3P0是一个开源的JDBC[连接池](http://baike.baidu.com/view/1528732.htm)，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](http://baike.baidu.com/view/188327.htm)有Hibernate，Spring等。

* Tomcat内置连接池：

【DBCP连接池的使用】

第一步：引入DBCP连接池的jar包.

第二步：编写DBCP代码：

\* 手动设置参数:

\* 配置文件设置参数:

【DBCP连接池的使用】

@Test

/\*\*

\* 手动方式:

\*/

public void demo1(){

Connection conn = null;

PreparedStatement stmt = null;

ResultSet rs = null;

BasicDataSource dataSource = new BasicDataSource();

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

dataSource.setUrl("jdbc:mysql:///web\_07");

dataSource.setUsername("root");

dataSource.setPassword("123");

try{

// 获得连接:

conn = dataSource.getConnection();

// 编写SQL:

String sql = "select \* from category";

// 预编译SQL:

stmt = conn.prepareStatement(sql);

// 执行SQL:

rs = stmt.executeQuery();

while(rs.next()){

System.out.println(rs.getInt("cid")+" "+rs.getString("cname"));

}

}catch(Exception e){

e.printStackTrace();

}finally{

JDBCUtils.release(rs,stmt, conn);

}

}

@Test

/\*\*

\* 配置文件方式：

\*/

public void demo2(){

Connection conn = null;

PreparedStatement stmt = null;

ResultSet rs = null;

Properties properties = new Properties();

try{

properties.load(new FileInputStream("src/dbcpconfig.properties"));

DataSource dataSource = BasicDataSourceFactory.createDataSource(properties);

// 获得连接:

conn = dataSource.getConnection();

// 编写SQL:

String sql = "select \* from category";

// 预编译SQL:

stmt = conn.prepareStatement(sql);

// 执行SQL:

rs = stmt.executeQuery();

while(rs.next()){

System.out.println(rs.getInt("cid")+" "+rs.getString("cname"));

}

}catch(Exception e){

e.printStackTrace();

}finally{

JDBCUtils.release(rs,stmt, conn);

}

}

【C3P0连接池的使用】

第一步：引入C3P0连接池的jar包.

第二步：编写代码：

\* 手动设置参数:

\* 配置文件设置参数：

【C3P0改造工具类】

public class JDBCUtils2 {

private static final ComboPooledDataSource DATA\_SOURCE =new ComboPooledDataSource();

/\*\*

\* 获得连接的方法

\*/

public static Connection getConnection(){

Connection conn = null;

try {

conn = DATA\_SOURCE.getConnection();

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return conn;

}

...

## ResultSetHandler

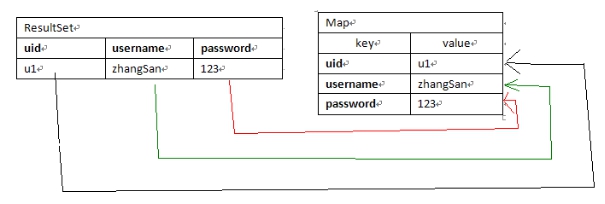
我们知道在执行select语句之后得到的是ResultSet，然后我们还需要对ResultSet进行转换，得到最终我们想要的数据。你可以希望把ResultSet的数据放到一个List中，也可能想把数据放到一个Map中，或是一个Bean中。

DBUtils提供了一个接口ResultSetHandler，它就是用来ResultSet转换成目标类型的工具。你可以自己去实现这个接口，把ResultSet转换成你想要的类型。

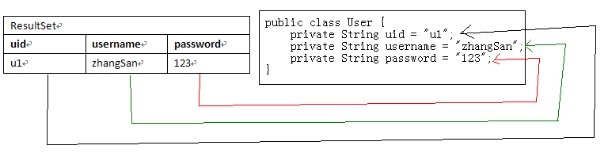
DBUtils提供了很多个ResultSetHandler接口的实现，这些实现已经基本够用了，我们通常不用自己去实现ResultSet接口了。

* MapHandler：单行处理器！把结果集转换成Map<String,Object>，其中列名为键！
* MapListHandler：多行处理器！把结果集转换成List<Map<String,Object>>；
* BeanHandler：单行处理器！把结果集转换成Bean，该处理器需要Class参数，即Bean的类型；
* BeanListHandler：多行处理器！把结果集转换成List<Bean>；
* ColumnListHandler：多行单列处理器！把结果集转换成List<Object>，使用ColumnListHandler时需要指定某一列的名称或编号，例如：new ColumListHandler(“name”)表示把name列的数据放到List中。
* ScalarHandler：单行单列处理器！把结果集转换成Object。一般用于聚集查询，例如select count(\*) from tab\_student。

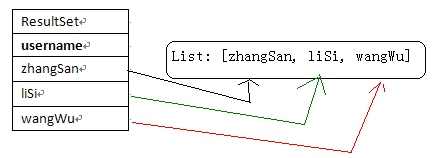
Map处理器



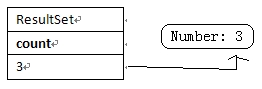
Bean处理器



Column处理器



Scalar处理器



## QueryRunner之查询

QueryRunner的查询方法是：

public <T> T query(String sql, ResultSetHandler<T> rh, Object… params)

public <T> T query(Connection con, String sql, ResultSetHandler<T> rh, Object… params)

query()方法会通过sql语句和params查询出ResultSet，然后通过rh把ResultSet转换成对应的类型再返回。

|  |
| --- |
| @Test  **public** **void** fun1() **throws** SQLException {  DataSource ds = JdbcUtils.*getDataSource*();  QueryRunner qr = **new** QueryRunner(ds);  String sql = "select \* from tab\_student where number=?";  Map<String,Object> map = qr.query(sql, **new** MapHandler(), "S\_2000");  System.*out*.println(map);  }    @Test  **public** **void** fun2() **throws** SQLException {  DataSource ds = JdbcUtils.*getDataSource*();  QueryRunner qr = **new** QueryRunner(ds);  String sql = "select \* from tab\_student";  List<Map<String,Object>> list = qr.query(sql, **new** MapListHandler());  **for**(Map<String,Object> map : list) {  System.*out*.println(map);  }  }    @Test  **public** **void** fun3() **throws** SQLException {  DataSource ds = JdbcUtils.*getDataSource*();  QueryRunner qr = **new** QueryRunner(ds);  String sql = "select \* from tab\_student where number=?";  Student stu = qr.query(sql, **new** BeanHandler<Student>(Student.**class**), "S\_2000");  System.*out*.println(stu);  }    @Test  **public** **void** fun4() **throws** SQLException {  DataSource ds = JdbcUtils.*getDataSource*();  QueryRunner qr = **new** QueryRunner(ds);  String sql = "select \* from tab\_student";  List<Student> list = qr.query(sql, **new** BeanListHandler<Student>(Student.**class**));  **for**(Student stu : list) {  System.*out*.println(stu);  }  }    @Test  **public** **void** fun5() **throws** SQLException {  DataSource ds = JdbcUtils.*getDataSource*();  QueryRunner qr = **new** QueryRunner(ds);  String sql = "select \* from tab\_student";  List<Object> list = qr.query(sql, **new** ColumnListHandler("name"));  **for**(Object s : list) {  System.*out*.println(s);  }  }    @Test  **public** **void** fun6() **throws** SQLException {  DataSource ds = JdbcUtils.*getDataSource*();  QueryRunner qr = **new** QueryRunner(ds);  String sql = "select count(\*) from tab\_student";  Number number = (Number)qr.query(sql, **new** ScalarHandler());  **int** cnt = number.intValue();  System.*out*.println(cnt);  } |