一.数据库连接池

1.数据库连接实际上是客户端和远程数据库服务器建立了一个socket连接, 创建过程比较耗性能和时间, 因此出现了数据库连接池技术。

2.数据库连接池负责分配、管理和释放数据库连接, 它允许应用程序重复使用一个现有的数据库连接，而不是再重新建立一个。这项技术能明显提高对数据库操作的性能。

3.连接池基本的思想是在系统初始化的时候, 将数据库连接作为对象存储在内存中, 当用户需要访问数据库时, 并非建立一个新的连接, 而是从连接池中取出一个已建立的空闲连接对象。使用完毕后, 用户也并非将连接关闭, 而是将连接放回连接池中, 以供下一个请求访问使用。而连接的建立、断开都由连接池自身来管理。同时, 还可以通过设置连接池的参数来控制连接池中的初始连接数、连接的上下限数以及每个连接的最大使用次数、最大空闲时间等等。也可以通过其自身的管理机制来监视数据库连接的数量、使用情况等。

4.在Java中常用的开源的数据库连接池有以下几种:

(1).C3P0: 是一个开放源代码的JDBC连接池, 包括了实现jdbc3和jdbc2扩展规范说明的Connection和Statement池的DataSources 对象。

(2).Proxool: 是一个Java SQL Driver驱动程序, 提供了对选择的其它类型的驱动程序的连接池封装。可以非常简单的移植到现存的代码中, 完全可配置, 快速、成熟、健壮。可以透明地为现存的JDBC驱动程序增加连接池功能。

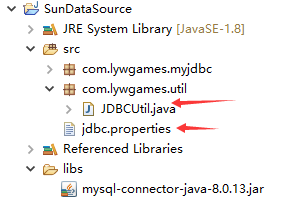
(3).Jakarta DBCP: DBCP是一个依赖Jakartacommons-pool对象池机制的数据库连接池。DBCP可以直接的在应用程序中使用。

(4).目前Proxool和DBCP以及C3P0一起, 最为常见的三种JDBC连接池技术。

(5).后来, Hibernate官方宣布由于Bug太多不再支持DBCP, 而推荐使用C3P0或Proxool。

5.自己实现Java的数据库连接池接口(DataSource)

(1).创建一个名称为SunDataSource的Java工程, 使用之前的JDBCUtil.java和jdbc.properties属性文件



(2).创建一个MyDataSource.java实现Java的DataSource接口

|  |
| --- |
| package com.lywgames.myjdbc;  import java.io.PrintWriter;  import java.sql.Connection;  import java.sql.SQLException;  import java.sql.SQLFeatureNotSupportedException;  import java.util.ArrayList;  import java.util.List;  import java.util.logging.Logger;  import javax.sql.DataSource;  import com.lywgames.util.JDBCUtil;  /\*\*  \* 实现java的DataSource接口  \*/  public class MyDataSource implements DataSource {  private List<Connection> list = new ArrayList<Connection>();    /\*\*  \* 对象一创建就创建10个数据库连接对象, 放到list对象  \*/  public MyDataSource() {  for (int i = 0; i < 10; i++) {  Connection conn = JDBCUtil.getConn();  list.add(conn);  }  }    // 该连接池对外公布的获取连接的方法  @Override  public Connection getConnection() throws SQLException {  // 发现连接池的连接数量为0, 自动扩容  if(list.size() == 0) {  for (int i = 0; i < 10; i++) {  Connection conn = JDBCUtil.getConn();  list.add(conn);  }  }    return list.remove(0);  }    /\*\*  \* 归还数据库连接对象  \* @param conn  \*/  public void backConnection(Connection conn) {  System.out.println("归还连接前, 连接数:" + list.size());  list.add(conn);  System.out.println("归还连接后, 连接数:" + list.size());  }    @Override  public Connection getConnection(String username, String password) throws SQLException {  return null;  }  @Override  public PrintWriter getLogWriter() throws SQLException {  return null;  }  @Override  public int getLoginTimeout() throws SQLException {  return 0;  }  @Override  public Logger getParentLogger() throws SQLFeatureNotSupportedException {  return null;  }  @Override  public void setLogWriter(PrintWriter arg0) throws SQLException {}  @Override  public void setLoginTimeout(int arg0) throws SQLException {}  @Override  public boolean isWrapperFor(Class<?> arg0) throws SQLException {  return false;  }  @Override  public <T> T unwrap(Class<T> arg0) throws SQLException {  return null;  }  } |

(3).使用MyDataSource

|  |
| --- |
| package com.lywgames.myjdbc;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.SQLException;  import java.sql.Timestamp;  import com.lywgames.util.JDBCUtil;  public class MyJDBC {  public static void main(String[] args) {  Connection conn = null;  PreparedStatement ps = null;  // 创建自己的连接池对象  MyDataSource ds = new MyDataSource();    try {  conn = ds.getConnection();  ps = conn.prepareStatement("insert into user values(null,?,?,?)");  ps.setString(1, "lvbu");  ps.setString(2, "123456");  ps.setTimestamp(3, new Timestamp(System.currentTimeMillis()));  ps.executeUpdate();  } catch (SQLException e) {  e.printStackTrace();  } finally {  JDBCUtil.release(ps);  }    // 归还连接  ds.backConnection(conn);  }  } |

(4).运行程序, 查看结果

