# 连接池

起因:频繁的连接和关闭影响程序的运行效率

连接管理:预先创建一组连接，需要的时候每次取出一个,用完后放回.

程序实现思路:

1. 指定“初始化连接数目”
2. 指定”最大连接数”
3. 指定”当前使用的连接个数”(不能超出最大连接数)

代码实现:

1. MyPool.java连接池类
2. 指定全局参数:初始化数目,最大连接数,当前连接,连接池集合
3. 构造函数:循环创建多个连接
4. 写一个创建连接的方法
5. 获取连接

判断:池中有连接,直接拿

池中没有连接,判断是否到达最大连接数,达到就抛出异常,没有,则创建新的连接

1. 释放连接

连接放回集合中

代理模式:

如果对某个接口中的某个指定的方法功能进行扩展,而有不想重写这个接口,那就使用（动态）代理模式

java中代理模式:静态/动态/Cglib代理(spring)

自定义连接池代码实现:

|  |
| --- |
| import java.lang.reflect.InvocationHandler;  import java.lang.reflect.Method;  import java.lang.reflect.Proxy;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.SQLException;  import java.util.LinkedList;  public class MyPool {  private int init\_count = 3; // 初始化连接数目  private int max\_count = 6; // 最大连接数  private int current\_count = 0; // 记录当前使用连接数  // 连接池 （存放所有的初始化连接）  private LinkedList<Connection> pool = new LinkedList<Connection>();  //1. 构造函数中，初始化连接放入连接池  public MyPool() {  // 初始化连接  for (int i=0; i<init\_count; i++){  // 记录当前连接数目  current\_count++;  // 创建原始的连接对象  Connection con = createConnection();  // 把连接加入连接池  pool.addLast(con);  }  }  //2. 创建一个新的连接的方法  private Connection createConnection(){  try {  Class.forName("com.mysql.jdbc.Driver");  // 原始的目标对象  final Connection con = DriverManager.getConnection("jdbc:mysql://47.96.27.61", "lw", "linwei");  /\*\*\*\*\*\*\*\*\*\*对con对象代理\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  // 对con创建其代理对象  Connection proxy = (Connection) Proxy.newProxyInstance(  con.getClass().getClassLoader(), // 类加载器  // con.getClass().getInterfaces(), // 当目标对象是一个具体的类的时候  new Class[]{Connection.class}, // 上面一行和本行差不多,最好上面一行  new InvocationHandler() { // 当调用con对象方法的时候， 自动触发事务处理器  @Override  public Object invoke(Object proxy, Method method, Object[] args)  throws Throwable {  // 方法返回值  Object result = null;  // 当前执行的方法的方法名  String methodName = method.getName();  // 判断当执行了close方法的时候，把连接放入连接池  if ("close".equals(methodName)) {  System.out.println("begin:当前执行close方法开始！");  // 连接放入连接池  pool.addLast(con);  System.out.println("end: 当前连接已经放入连接池了！");  } else {  // 调用目标对象方法  result = method.invoke(con, args);  }  return result;  }  }  );  return proxy;  } catch (Exception e) {  throw new RuntimeException(e);  }  }  //3. 获取连接  public Connection getConnection(){  // 3.1 判断连接池中是否有连接, 如果有连接，就直接从连接池取出  if (pool.size() > 0){  return pool.removeFirst();  }  // 3.2 连接池中没有连接： 判断，如果没有达到最大连接数，创建；  if (current\_count < max\_count) {  // 记录当前使用的连接数  current\_count++;  // 创建连接  return createConnection();  }  // 3.3 如果当前已经达到最大连接数，抛出异常  throw new RuntimeException("当前连接已经达到最大连接数目 ！");  }  //4. 释放连接  public void realeaseConnection(Connection con) {  // 4.1 判断： 池的数目如果小于初始化连接，就放入池中  if (pool.size() < init\_count){  pool.addLast(con);  } else {  try {  // 4.2 关闭  current\_count--;  con.close();  } catch (SQLException e) {  throw new RuntimeException(e);  }  }  }  public static void main(String[] args) throws SQLException {  MyPool pool = new MyPool();  System.out.println("当前连接: " + pool.current\_count); // 3  // 使用连接  pool.getConnection();  pool.getConnection();  Connection con4 = pool.getConnection();  Connection con3 = pool.getConnection();  Connection con2 = pool.getConnection();  Connection con1 = pool.getConnection();  // 释放连接, 连接放回连接池  // pool.realeaseConnection(con1);  /\*  \* 希望：当关闭连接的时候，要把连接放入连接池！【当调用Connection接口的close方法时候，希望触发pool.addLast(con);操作】  \* 把连接放入连接池  \* 解决1：实现Connection接口，重写close方法  \* 解决2：动态代理  \*/  con1.close();  // 再获取  pool.getConnection();  System.out.println("连接池：" + pool.pool.size()); // 0  System.out.println("当前连接: " + pool.current\_count); // 3  }  } |

# 开源的连接池技术

概述:sun公司规定:如果是连接池技术,需要实现一个接口,

javax.sql.DataSource

连接池:

1. DBCP数据库连接池(tomcat)
2. C3P0数据库连接池(hibernate)

## DBCP连接池

DBCP是apache基金会组织的开源连接池实现,

需要两个jar文件

Commons-dbcp.jar;//连接池的实现

Commons-pool.jar;//连接池实现的依赖库

tomcat的连接池正式采用了连接池来实现的,该数据库连接池可以与应用服务器整合使用

核心类:BasicDataSource

两种方式进行使用:

1. 硬编码方式
2. 配置文件方式

代码:

|  |
| --- |
| public class App\_Dbcp {  public static void main(String args[]) throws Exception{  f2();  }  //方式1:硬编码方式  public static void f1() throws Exception{  //DBCP连接池核心类  BasicDataSource dataSource = new BasicDataSource();  //连接池参数配置:初始化连接数,最大连接数,连接字符串,驱动,用户,密码  dataSource.setUrl("jdbc:mysql://47.96.27.61:3306/test");  dataSource.setDriverClassName("com.mysql.jdbc.Driver");//数据库驱动  dataSource.setUsername("lw");  dataSource.setPassword("linwei");  dataSource.setInitialSize(3);// 初始化连接  dataSource.setMaxTotal(6);//最大连接数  dataSource.setMaxIdle(3000);//最大空闲时间  //获取连接  Connection con = dataSource.getConnection();  con.prepareStatement("delete from admin WHERE id=2").executeUpdate();  con.close();  }  //方式2:配置文件实现连接池配置(这种方式为主要)  public static void f2() throws Exception{  //加载properties配置文件  Properties prop = new Properties();  //加载属性配置文件  InputStream inputStream = App\_Dbcp.class.getResourceAsStream("dbcp.properties");  prop.load(inputStream);  //根据  BasicDataSource dataSource = BasicDataSourceFactory.createDataSource(prop);  Connection con = dataSource.getConnection();  con.prepareStatement("delete from admin WHERE id=2").executeUpdate();  con.close();  }  } |

dbcp.properties文件:

|  |
| --- |
| url=jdbc:mysql://47.96.27.61:3306/test  driverClassName=com.mysql.jdbc.Driver  username=lw  password=linwei  initialSize=3  maxTotal=6  maxIdle=3000 |

注意:配置文件中的key和BaseDataSource中的属性一样

## 2.2C3P0连接池

C3P0连接池:

最常用的连接池技术,Spring框架,默认支持C3P0连接池

核心类:CombopooledDataSource

需要2个jar文件:

c3p0-0.9.5.2.jar

mchange-commons-java-0.2.11.jar

使用方式:

1. 硬编码方式
2. 配置文件方式(xml)

代码:

|  |
| --- |
| import com.mchange.v2.c3p0.ComboPooledDataSource;  import java.sql.\*;  public class Main{  public static void main(String args[]) throws Exception{  f2();  }  public static void f1() throws Exception{  //创建连接池核心工具类  ComboPooledDataSource dataSource = new ComboPooledDataSource ();  //设置连接参数:url,驱动,用户,密码,初始连接数,最大连接数  dataSource.setJdbcUrl("jdbc:mysql://47.96.27.61:3306/test");  dataSource.setDriverClass("com.mysql.jdbc.Driver");  dataSource.setUser("lw");  dataSource.setPassword("linwei");  dataSource.setInitialPoolSize(3);  dataSource.setMaxIdleTime(3000);  dataSource.setMaxPoolSize(6);  //从连接池对象中获取连接对象  Connection conn = dataSource.getConnection();  //执行更新  conn.prepareStatement("delete from admin where id=2").executeUpdate();  //关闭  conn.close();  }  //方式2:xml配置文件方式  public static void f2() throws Exception{  //创建连接池核心工具类  //自动加载src下c3p0配置文件c3p0-config.xml  ComboPooledDataSource dataSource = new ComboPooledDataSource ();  //当然还可以传递字符串,这个字符串在配置文件中  //ComboPooledDataSource dataSource = new ComboPooledDataSource ("oracle-config");  //从连接池对象中获取连接对象  Connection conn = dataSource.getConnection();  //执行更新  conn.prepareStatement("delete from admin where id=2").executeUpdate();  //关闭  conn.close();  }  } |

c3p0-config.xml:

|  |
| --- |
| <c3p0-config>  <!-- 使用默认的配置读取连接池对象 -->  <default-config>  <!-- 连接参数 -->  <property name="driverClass">com.mysql.jdbc.Driver</property>  <property name="jdbcUrl">jdbc:mysql://47.96.27.61:3306/test</property>  <property name="user">lw</property>  <property name="password">linwei</property>    <!-- 连接池参数 -->  <property name="initialPoolSize">5</property>  <property name="maxPoolSize">8</property>  <property name="checkoutTimeout">3000</property>  </default-config>  <named-config name="mysql\_day18">  <!-- 连接参数 -->  <property name="driverClass">com.mysql.jdbc.Driver</property>  <property name="jdbcUrl">jdbc:mysql://47.96.27.61:3306/day18</property>  <property name="user">lw</property>  <property name="password">linwei</property>    <!-- 连接池参数 -->  <property name="initialPoolSize">5</property>  <property name="maxPoolSize">8</property>  <property name="checkoutTimeout">3000</property>  </named-config>  </c3p0-config> |