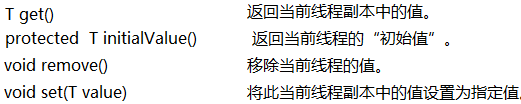
# 绑定本地线程—线程池

## ThreadLocal类

### 构造方法



### 常用方法



### MyThreadLocal<T>

**public** **class** MyThreadLocal<T> {

**private** Map<Thread, T> map;

**private** MyThreadLocal() {

map = **new** HashMap<Thread, T>();

}

**public** **void** set(T t) {

map.put(Thread.*currentThread*(), t);

}

**public** T get() {

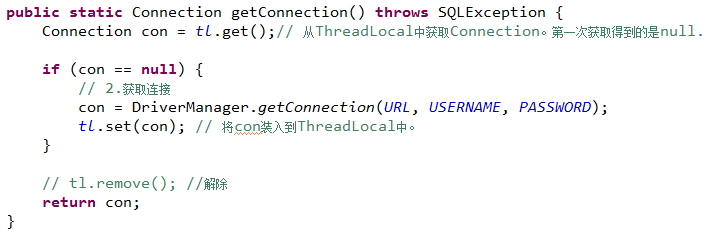
**return** map.get(Thread.*currentThread*());

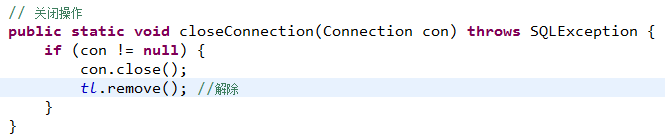
}

}

## 关于JdbcUtils中使用ThreadLocal







# 实例

## Druid连接池，线程事务

com.prosay.utils.DruidUtils

public class DruidUtils {  
 // 线程局部变量，保存某个线程使用的连接  
 private static ThreadLocal<Connection> *t* = new ThreadLocal<>();  
 // 无参构造方法会去获取c3p0-config.xml配置文件中的默认配置信息default-config  
 //static DataSource ds = new ComboPooledDataSource();  
 // 获取配置文件中named-config name="oracle11gorcl"对应的配置  
 static DruidDataSourceFactory *f* = new DruidDataSourceFactory();  
 static DataSource *ds* = null;  
 static{  
 Properties properties = new Properties();  
 try {  
 properties.load(DruidUtils.class.getClassLoader().getResourceAsStream("db.properties"));  
 properties.setProperty("driverClassName", properties.getProperty("jdbc.driver"));  
 properties.setProperty("url", properties.getProperty("jdbc.url"));  
 properties.setProperty("username", properties.getProperty("jdbc.user"));  
 properties.setProperty("password", properties.getProperty("jdbc.password"));  
 *ds* = *f*.*createDataSource*(properties); // 由此倒推  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 public static Connection getConnection(){  
 // 首先从线程局部变量中获取连接，如果线程局部变量中没有保存有连接，说明是第一次获取  
 Connection conn = *t*.get();  
 if (conn == null) {  
 try {  
 // 第一次获取，从数据源，数据库连接池中获取  
 conn = *ds*.getConnection();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 return conn;  
 }  
  
 public static void closeConnection(){  
 Connection conn = *t*.get();  
 if (conn != null) {  
 try {  
 conn.close();  
 *t*.remove();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 */\*\*  
 \* 开启事务  
 \*/* public static void openTransaction(){  
 try {  
 // 设置自动提交为false，执行SQL语句时并没有立即生效，需要手动调用commit方法  
 *getConnection*().setAutoCommit(false);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 */\*\*  
 \* 提交事务  
 \*/* public static void commitTransaction(){  
 try {  
 *getConnection*().commit();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 */\*\*  
 \* 回滚事务，发生异常时，应该撤销的操作  
 \*/* public static void rollbackTransaction(){  
 try {  
 *getConnection*().rollback();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
   
 /\*public static void main(String[] args) {  
 for (int i = 0; i < 4; i++) {  
 System.out.println(DruidUtils.getConnection());  
 }  
 }\*/  
}