**ThreadLocal**

平时我们在解决线程问题的时候我们都是用锁将多个线程都要访问的数据放在锁对象中，但是还有另外一种思想就是最原始的思想，在每一次线程中都有一个要使用的对象，每个线程都是用自己线程中的那个对象这样也就避免了线程问题。

1.概念：

ThreadLocal类用来提供线程内部的局部变量。这些变量在多线程环境下访问(通过get或set方法访问)时能保证各个线程里的变量相对独立于其他线程内的变量，ThreadLocal实例通常来说都是private static类型。

总结：ThreadLocal不是为了解决多线程访问共享变量，而是为每个线程创建一个单独的变量副本，提供了保持对象的方法和避免参数传递的复杂性。

2.实现原理：

ThreadLocal可以看做是一个容器，容器里面存放着属于当前线程的变量。ThreadLocal类提供了四个对外开放的接口方法，这也是用户操作ThreadLocal类的基本方法：   
1)void set(Object value)设置当前线程的线程局部变量的值。   
2)public Object get()该方法返回当前线程所对应的线程局部变量。   
3) public void remove()将当前线程局部变量的值删除，目的是为了减少内存的占用，该方法是JDK 5.0新增的方法。需要指出的是，当线程结束后，对应该线程的局部变量将自动被垃圾回收，所以显式调用该方法清除线程的局部变量并不是必须的操作，但它可以加快内存回收的速度。   
4) protected Object initialValue()返回该线程局部变量的初始值，该方法是一个protected的方法，显然是为了让子类覆盖而设计的。这个方法是一个延迟调用方法，在线程第1次调用get()或set(Object)时才执行，并且仅执行1次，ThreadLocal中的缺省实现直接返回一个null。

可以通过上述的几个方法实现ThreadLocal中变量的访问，数据设置，初始化以及删除局部变量，那ThreadLocal内部是如何为每一个线程维护变量副本的呢？

其实在ThreadLocal类中有一个静态内部类ThreadLocalMap(其类似于Map)，用键值对的形式存储每一个线程的变量副本，ThreadLocalMap中元素的key为当前ThreadLocal对象，而value对应线程的变量副本，每个线程可能存在多个ThreadLocal。

3.优点与缺点：

ThreadLocal使用场合主要解决多线程中数据因并发产生不一致问题。ThreadLocal为每个线程的中并发访问的数据提供一个副本，通过访问副本来运行业务，这样的结果是耗费了内存，大大减少了线程同步所带来性能消耗，也减少了线程并发控制的复杂度。

ThreadLocal的使用比synchronized要简单得多。

ThreadLocal和Synchonized都用于解决多线程并发访问。但是ThreadLocal与synchronized有本质的区别。

a. synchronized是利用锁的机制，使变量或代码块在某一时该只能被一个线程访问。

b. ThreadLocal为每一个线程都提供了变量的副本，使得每个线程在某一时间访问到的并不是同一个对象，这样就隔离了多个线程对数据的数据共享。而Synchronized却正好相反，它用于在多个线程间通信时能够获得数据共享。

Synchronized用于线程间的数据共享，而ThreadLocal则用于线程间的数据隔离。

当然ThreadLocal并不能替代synchronized,它们处理不同的问题域。Synchronized用于实现同步机制，比ThreadLocal更加复杂