# Threadlocal原理

ThreadLocal 为解决多线程程序的并发问题提供了一种新的思路。使用这个工具类可以很简洁地编写出优美的多线程程序。当使用 ThreadLocal 维护变量时，ThreadLocal 为每个使用该变量的线程提供独立的变量副本，所以每一个线程都可以独立地改变自己的副本，而不会影响其它线程所对应的副本。

每个线程中都保有一个ThreadLocalMap的成员变量，ThreadLocalMap内部采用WeakReference数组保存，数组的key即为ThreadLocal内部的Hash值。

## 内存泄漏

ThreadLocalMap 使用 ThreadLocal 的弱引用作为 key ，如果一个 ThreadLocal 没有外部强引用来引用它，那么系统 GC 的时候，这个 ThreadLocal 势必会被回收，这样一来，ThreadLocalMap 中就会出现 key 为 null 的 Entry ，就没有办法访问这些 key 为 null 的 Entry 的 value，如果当前线程再迟迟不结束的话，这些 key 为 null 的 Entry 的 value 就会一直存在一条强引用链：Thread Ref -> Thread -> ThreaLocalMap -> Entry -> value 永远无法回收，造成内存泄漏。

static class Entry extends WeakReference<ThreadLocal<?>> {
 /\*\* The value associated with this ThreadLocal. \*/
 Object value;

 Entry(ThreadLocal<?> k, Object v) {
 super(k);
 value = v;
 }
}

其实，ThreadLocalMap 的设计中已经考虑到这种情况，也加上了一些防护措施：在 ThreadLocal 的 get(),set(),remove()的时候都会清除线程 ThreadLocalMap 里所有 key 为 null 的 value