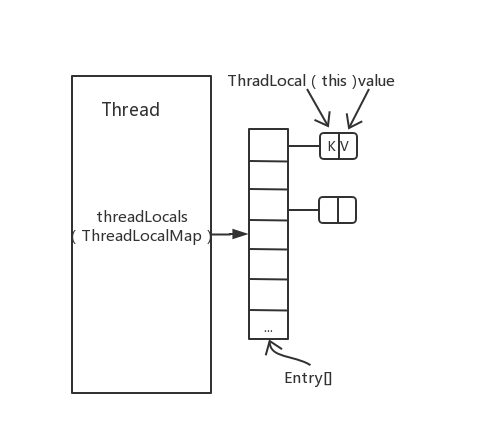
# ThreadLocal简介

ThreadLocal 可以使多线程间数据读写隔离，因此 ThreadLocal 解决的是线程局部变量安全性问题，并不是多线程间共享变量安全性问题。ThreadLocal 在使用时必须先初始化 value，否则会报空指针异常，你可以通过 set 方法与重写 initialValue 方法两种方式初始化 value。



## ThreadLocal 源码

### 1.1.1内部相关属性

|  |
| --- |
| /\*  \* ThreadLocal 的哈希值通过一个原子类计算  \*/  private final int threadLocalHashCode = nextHashCode();  /\*\*  \* 用于计算 ThreadLocal 哈希值的原子类  \*/  private static AtomicInteger nextHashCode = new AtomicInteger();  /\*\*  \* 计算 ThreadLocal 哈希值的魔数  \* 该值生成出来的值可以较为均匀地分布在 2 的幂大小的数组中  \* 与斐波那契散列有关...  \*/  private static final int HASH\_INCREMENT = 0x61c88647; |

ThreadLocalMap 的结构是通过纯数组实现的，因此 ThreadLocal 计算哈希值的方式也比较特殊，通过 nextHashCode() 方法生成哈希值，下面是具体实现。

|  |
| --- |
| private static int nextHashCode() {  return nextHashCode.getAndAdd(HASH\_INCREMENT);  } |

生成哈希值时每次加上 0x61c88647，据了解通过 0x61c88647 计算出来的哈希值能够均匀的分布在 2 的幂大小的数组中，有兴趣的可以网上查一下进行详细的了解。

### 1.1.2 set 方法