# **AtomicInteger**

JDK1.5之后的java.util.concurrent.atomic包里面多了一批原子处理类。AtomicBoolean、AtomicInteger、AtomicLong、AtomicReference。主要用于在高并发环境下的高效程序处理，来帮我们简化并发处理。

AtomicInteger一个提供原子操作的Integer的类。在java语言中i++和++i并不是线程安全的，在使用的时候不可避免使用到synchronized关键字，而通过AtomicInteger类操作Integer类是线程安全的，不必使用synchronized关键字：

public final int get() //获取当前的值  
 public final int getAndSet(int newValue)//获取当前的值，并设置新的值  
 public final int getAndIncrement()//获取当前的值，并自增  
 public final int getAndDecrement() //获取当前的值，并自减  
 public final int getAndAdd(int delta) //获取当前的值，并加上预期的值

public final boolean compareAndSet(int expect, int update)//如果AtomicInteger对象中 的值等于expect，则将AtomicInteger对象的值设置为update。

**AtomicInteger是通过非阻塞算法来实现并发控制的：**

**private static final** sun.misc.Unsafe ***U*** = sun.misc.Unsafe.getUnsafe();  
 **private static final long *VALUE***;  
  
 **static** {  
 **try** {  
 ***VALUE*** = ***U***.objectFieldOffset  
 (AtomicInteger.**class**.getDeclaredField(**"value"**));  
 } **catch** (ReflectiveOperationException e) {  
 **throw new** Error(e);  
 }  
 }  
  
 **private volatile int value**;

***U :***实际上就是AtomicInteger的一个工具类,Unsafe类可以执行以下几种操作:

1. 分配内存，释放内存：在方法allocateMemory，reallocateMemory，freeMemory中，有点类似c

中的malloc，free方法。

1. 可以定位对象的属性在内存中的位置，获得对对象内存地址，可以修改对象的属性值。使用

objectFieldOffset方法。

3、挂起和恢复线程，被封装在LockSupport类中供使用

4、CAS操作(CompareAndSwap，比较并交换，是一个原子操作)

***VALUE:***是用来记录value本身在内存中的地址的，这个记录也主要是为了更新操作

在内存中找到value的位置，方便比较。

***value：***AtomicInteger内部使用变量value表示当前的整型值，这个整型变量还是volatile的，表示内存可见性，一个线程修改value之后保证对其他线程的可见性。

**并发控制的核心算法：**

addAndGet方法举例，addAndGet方法内部使用一个死循环，先得到当前的值value，然后再把当前的值加一，加完之后使用cas原子操作让当前值加一处理正确。当然cas原子操作不一定是成功的，所以做了一个死循环，当cas操作成功的时候返回数据。这里由于使用了cas原子操作，所以不会出现多线程处理错误的问题。比如线程A得到current为1，线程B也得到current为1；线程A的next值为2，进行cas操作并且成功的时候，将value修改成了2；这个时候线程B也得到next值为2，当进行cas操作的时候由于expected值已经是2，而不是1了；所以cas操作会失败，下一次循环的时候得到的current就变成了2；也就不会出现多线程处理问题了：

public final int addAndGet(int delta) {

for (;;) {

int current = get();

int next = current + delta;

if (compareAndSet(current, next))

return next;

}

}