

X-Spirit

Always Beyond the Time

BlogJava | 首页 | 发新随笔 | 发新文章 | 联系 | 聚合XML | 管理

随笔：80 文章：1 评论：58 引用：0

Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）【更新重要补疑】

前面我们介绍了Java当中多个线程抢占一个共享资源的问题。但不论是同步还是重入锁，都不能实实在在的解决资源紧缺的情况，这些方案只是靠制定规则来约束线程的行为，让它们不再拼命的争抢，而不是真正从实质上解决他们对资源的需求。

在JDK 1.2当中，引入了java.lang.ThreadLocal。它为我们提供了一种全新的思路来解决线程并发的。但是他的名字难免让我们望文生义：本地线程？

什么是本地线程？
本地线程开玩笑的说：不要迷恋哥，哥只是个传说。

其实ThreadLocal并非Thread at Local，而是LocalVariable in a Thread。

根据WikiPedia上的介绍，ThreadLocal其实是源于一项多线程技术，叫做Thread Local Storage，即线程本地存储技术。不仅仅是Java，在C++、C#、.NET、Python、Ruby、Perl等开发平台上，该技术都已经得以实现。

当使用ThreadLocal维护变量时，它会为每个使用该变量的线程提供独立的变量副本。也就是说，他从根本上解决的是资源数量的问题，从而使得每个线程持有相对独立的资源。这样，当多个线程进行工作的时候，它们不需要纠结于同步的问题，于是性能便大大提升。但资源的扩张带来的是更多的空间消耗，ThreadLocal就是这样一种利用空间来换取时间的解决方案。

说了这么多，来看看如何正确使用ThreadLocal。

通过研究JDK文档，我们知道，ThreadLocal中有几个重要的方法：get()、set()、remove()、initailValue()，对应的含义分别是：
返回此线程局部变量的当前线程副本中的值、将此线程局部变量的当前线程副本中的值设置为指定值、移除此线程局部变量当前线程的值、返回此线程局部变量的当前线程的“初始值”。

还记得我们在第三篇的上半节引出的那个例子么？几个线程修改同一个Student对象中的age属性。为了保证这几个线程能够工作正常，我们需要对Student的对象进行同步。下面我们对这个程序进行一点小小的改造，我们通过继承Thread来实现多线程：

```
/**
 *
 * @author x-spirit
 */
```

< 2010年4月 >						
日	一	二	三	四	五	六
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1
2	3	4	5	6	7	8

常用链接

- 我的随笔
- 我的文章
- 我的评论
- 我的参与
- 最新评论

留言簿(5)

- 给我留言
- 查看公开留言
- 查看私人留言

随笔分类(17)

- Core Java翻译 (rss)
- Java EE (rss)
- Java FX (rss)
- Java SE(6) (rss)
- Java 轻量级企业开发(5) (rss)
- Linux (rss)
- MySQL (rss)
- NetBeans(1) (rss)
- Oracle(3) (rss)
- 其他(1) (rss)
- 开源协议(1) (rss)

随笔档案(79)

- 2010年4月 (6)
- 2010年3月 (3)
- 2010 1 (2)

```
public class ThreadDemo3 extends Thread{

    private ThreadLocal<Student> stuLocal = new ThreadLocal<Student>();

    public ThreadDemo3(Student stu){
        stuLocal.set(stu);
    }

    public static void main(String[] args) {
        Student stu = new Student();
        ThreadDemo3 td31 = new ThreadDemo3(stu);
        ThreadDemo3 td32 = new ThreadDemo3(stu);
        ThreadDemo3 td33 = new ThreadDemo3(stu);
        td31.start();
        td32.start();
        td33.start();
    }

    @Override
    public void run() {
        accessStudent();
    }

    public void accessStudent() {

        String currentThreadName = Thread.currentThread().getName();

        System.out.println(currentThreadName + " is running!");
        Random random = new Random();
        int age = random.nextInt(100);
        System.out.println("thread " + currentThreadName + " set age to:" + age);
        Student student = stuLocal.get();
        student.setAge(age);
        System.out.println("thread " + currentThreadName + " first read age is:" + student.getAge());
        try {
            Thread.sleep(5000);
        } catch (InterruptedException ex) {
            ex.printStackTrace();
        }
        System.out.println("thread " + currentThreadName + " second read age is:" + student.getAge());
    }
}
```

貌似这个程序没什么问题。但是运行结果却显示：这个程序中的3个线程会抛出3个空指针异常。读者一定感到很困惑。我明明在构造器当中把Student对象set进了ThreadLocal里面阿，为什么run起来之后居然在调用stuLocal.get()方法的时候得到的是NULL呢？

年 月

- 2009年11月 (1)
- 2009年8月 (1)
- 2009年7月 (1)
- 2009年6月 (2)
- 2009年5月 (1)
- 2009年4月 (13)
- 2009年3月 (1)
- 2009年2月 (1)
- 2009年1月 (2)
- 2008年12月 (3)
- 2008年11月 (1)
- 2008年10月 (1)
- 2008年9月 (3)
- 2008年5月 (1)
- 2008年4月 (9)
- 2008年3月 (3)
- 2008年1月 (1)
- 2007年12月 (4)
- 2007年10月 (2)
- 2007年9月 (6)
- 2007年8月 (9)
- 2007年7月 (1)
- 2007年4月 (1)

文章分类(1)

- My Voice(1) (rss)

文章档案(1)

- 2009年12月 (1)

收藏夹(4)

- 别人的学习笔记(4) (rss)

牛人牛博

- 22apple
- 22apple
- DaoRu的博客 (rss)
- 有思想的道儒
- raof01 -Haste makes waste
- 很牛的C++ 程序员
- Tim Yang (rss)
- 老大的博客
- 其实我是一个程序员
- 此人今后必火！
- 摇摆巴赫@blog.sina

带着这个疑问，让我们深入到JDK的代码当中，去一看究竟。

原来，在ThreadLocal中，有一个内部类叫做ThreadLocalMap。这个ThreadLocalMap并非java.util.Map的一个实现，而是利用java.lang.ref.WeakReference实现的一个键-值对应的数据结构其中，key是ThreadLocal类型，而value是Object类型，我们可以简单的视为HashMap<ThreadLocal,Object>。

而在每一个Thread对象中，都有一个ThreadLocalMap的引用，即Thread.threadLocals。而ThreadLocal的set方法就是首先尝试从当前线程中取得ThreadLocalMap（以下简称Map）对象。如果取到的不为null，则以ThreadLocal对象自身为key，来取Map中的value。如果取不到Map对象，则首先为当前线程创建一个ThreadLocalMap，然后以ThreadLocal对象自身为key，将传入的value放入该Map中。

```
ThreadLocalMap getMap(Thread t) {
    return t.threadLocals;
}

public void set(T value) {
    Thread t = Thread.currentThread();
    ThreadLocalMap map = getMap(t);
    if (map != null)
        map.set(this, value);
    else
        createMap(t, value);
}
```

而get方法则是首先得到当前线程的ThreadLocalMap对象，然后，根据ThreadLocal对象自身，取出相应的value。当然，如果在当前线程中取不到ThreadLocalMap对象，则尝试为当前线程创建ThreadLocalMap对象，并以ThreadLocal对象自身为key，把initialValue()方法产生的对象作为value放入新创建的ThreadLocalMap中。

```
public T get() {
    Thread t = Thread.currentThread();
    ThreadLocalMap map = getMap(t);
    if (map != null) {
        ThreadLocalMap.Entry e = map.getEntry(this);
        if (e != null)
            return (T)e.value;
    }
    return setInitialValue();
}

private T setInitialValue() {
    T value = initialValue();
    Thread t = Thread.currentThread();
    ThreadLocalMap map = getMap(t);
    if (map != null)
        map.set(this, value);
}
```

- 对服务器开发很有研究的田大师
- 摇摆巴赫@javaeye
- 对服务器开发很有研究的田大师
- 文初的一亩三分地
- 这家伙真酷！高性能网站专家啊！
- 那谁的BLOG
- 那谁的BLOG

最新随笔

- 1. Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）【更新重要补疑】
- 2. Java 多线程同步问题的探究（四、协作，互斥下的协作——Java多线程协作（wait、notify、notifyAll））
- 3. Java 多线程同步问题的探究（三、Lock来了，大家都让开【2. Fair or Unfair? It is a question...】）
- 4. Java 多线程同步问题的探究（三、Lock来了，大家都让开【1. 认识重入锁】）
- 5. Java 多线程同步问题的探究（二、给我一把锁，我能创造一个规矩）
- 6. Java多线程同步问题的探究（一、线程的先来后到）
- 7. [转]关于hibernate的缓存使用
- 8. NetBeans中配置OperaMask+Spring+JPA教程
- 9. Oracle 11g口令过期
- 10. Oracle not in查不到应有的结果(NULL、IN、EXISTS详解)

搜索

```
else
    createMap(t, value);
return value;
}

protected T initialValue() {
    return null;
}
```

这样，我们就明白上面的问题出在哪里：我们在main方法执行期间，试图在调用ThreadDemo3的构造器时向ThreadLocal置入Student对象，而此时，以ThreadLocal对象为key，Student对象为value的Map是被放入当前的活动线程内的。也就是Main线程。而当我们的3个ThreadDemo3线程运行起来以后，调用get()方法，都是试图从当前的活动线程中取得ThreadLocalMap对象，但当前的活动线程显然已经不是Main线程了，于是，程序最终执行了ThreadLocal原生的initialValue()方法，返回了null。

讲到这里，我想不少朋友一定已经看出来了：ThreadLocal的initialValue()方法是需要被覆盖的。

于是，ThreadLocal的正确使用方法是：将ThreadLocal以内部类的形式进行继承，并覆盖原来的initialValue()方法，在这里产生可供线程拥有的本地变量值。这样，我们就有了下面的正确例程：

```
/**
 *
 * @author x-spirit
 */
public class ThreadDemo3 extends Thread{

    private ThreadLocal<Student> stuLocal = new ThreadLocal<Student>(){

        @Override
        protected Student initialValue() {
            return new Student();
        }

    };

    public ThreadDemo3(){

    }

    public static void main(String[] args) {
        ThreadDemo3 td31 = new ThreadDemo3();
        ThreadDemo3 td32 = new ThreadDemo3();
        ThreadDemo3 td33 = new ThreadDemo3();
        td31.start();
        td32.start();
    }
}
```

-
-

最新评论 XML

- 1. re: Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）【更新重要补疑】
- 不错的编程 支持技术贴 --朱少
- 2. re: Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）【更新重要补疑】
- 好详细 了解了镇面貌了 --nauxiaoyao
- 3. re: Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）【更新重要补疑】
- 好全啊 谢谢楼主 --nauxiaoyao
- 4. re: Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）【更新重要补疑】
- 技术的确的支持，顶一下 --朱少
- 5. re: Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）【更新重要补疑】
- 支持技术贴，软件也不错 --圣光永恒

```

        td33.start();
    }

    @Override
    public void run() {
        accessStudent();
    }

    public void accessStudent() {

        String currentThreadName = Thread.currentThread().getName()
;

        System.out.println(currentThreadName + " is running!");
        Random random = new Random();
        int age = random.nextInt(100);
        System.out.println("thread " + currentThreadName + " set age
to:" + age);
        Student student = stuLocal.get();
        student.setAge(age);
        System.out.println("thread " + currentThreadName + " first re
ad age is:" + student.getAge());
        try {
            Thread.sleep(5000);
        } catch (InterruptedException ex) {
            ex.printStackTrace();
        }
        System.out.println("thread " + currentThreadName + " second r
ead age is:" + student.getAge());

    }
}

```

***** 补疑 *****

有的童鞋可能会问：“你这个Demo根本没体现出来，每个线程里都有一个ThreadLocal对象；应该是一个ThreadLocal对象对应多个线程，你这变成了一对一，完全没体现出ThreadLocal的作用。”

那么我们来看一下如何用一個ThreadLocal对象来对应多个线程：

```

/**
 *
 * @author x-spirit
 */
public class ThreadDemo3 implements Runnable{

    private ThreadLocal<Student> stuLocal = new ThreadLocal<Stud
ent>(){

        @Override
        protected Student initialValue() {
            return new Student();
        }
    }
}

```

阅读排行榜

- 1. Java 多线程同步问题的探究（四、协作，互斥下的协作——Java多线程协作（wait、notify、notifyAll））(5344)
- 2. Java 多线程同步问题的探究（三、Lock来了，大家都让开【1. 认识重入锁】）(5254)
- 3. Java 多线程同步问题的探究（二、给我一把锁，我能创造一个规矩）(5212)
- 4. Java多线程同步问题的探究（一、线程的先来后到）(5182)
- 5. Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）【更新重要补疑】(4793)

评论排行榜

- 1. Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）【更新重要补疑】(15)
- 2. Java 多线程同步问题的探究（四、协作，互斥下的协作——Java多线程协作（wait、notify、notifyAll））(9)
- 3. Java 多线程同步问题的探究（三、Lock来了，大家都让开【1. 认识重入锁】）(9)
- 4. Java 多线程同步问题的探究（二、给我一把锁，我能创造一个规矩）(9)
- 5. Java多线程同步问题的探究（一、线程的先来后到）(7)

```
};

public ThreadDemo3(){

}

public static void main(String[] args) {
    ThreadDemo3 td3 = new ThreadDemo3();
    Thread t1 = new Thread(td3);
    Thread t2 = new Thread(td3);
    Thread t3 = new Thread(td3);
    t1.start();
    t2.start();
    t3.start();
}

@Override
public void run() {
    accessStudent();
}

public void accessStudent() {

    String currentThreadName = Thread.currentThread().getName
();
    System.out.println(currentThreadName + " is running!");
    Random random = new Random();
    int age = random.nextInt(100);
    System.out.println("thread " + currentThreadName + " set ag
e to:" + age);
    Student student = stuLocal.get();
    student.setAge(age);
    System.out.println("thread " + currentThreadName + " first r
ead age is:" + student.getAge());
    try {
        Thread.sleep(5000);
    } catch (InterruptedException ex) {
        ex.printStackTrace();
    }
    System.out.println("thread " + currentThreadName + " second
read age is:" + student.getAge());
}
}
```

Powered by: 博客园
模板提供: 沪江博客
Copyright ©2011 X-Spi
rit

这里，多个线程对象都使用同一个实现了Runnable接口的ThreadDemo3对象来构造。这样，多个线程使用的ThreadLocal对象就是同一个。结果仍然是正确的。但是仔细回想一下，这两种实现方案有什么不同呢？

答案其实很简单，并没有本质上的不同。对于第一种实现，不同的线程对象当中ThreadLocalMap里面的KEY使用的是不同的ThreadLocal对象。而对于第二种实现，不同的线程对象当中ThreadLocalMap里面的KEY是同一个ThreadLocal对象。但是从本质上讲，不同的线程对象都是利用其自身的ThreadLocalMap对象来对各自的Student对象进行封装，

用ThreadLocal对象作为该ThreadLocalMap的KEY。所以说，“ThreadLocal的思想精髓就是为每个线程创建独立的资源副本。”这句话并不应当被理解成：一定要使用同一个ThreadLocal对象来对多个线程进行处理。因为真正用来封装变量的不是ThreadLocal。就算是你的程序中所有线程都共用同一个ThreadLocal对象，而你真正封装到ThreadLocalMap中去的仍然是.hashCode()方法返回不同值的不同对象。就好比线程就是房东，ThreadLocalMap就是房东的房子。房东通过ThreadLocal这个中介去和房子里的房客打交道，而房东不管要让房客住进去还是搬出来，都首先要经过ThreadLocal这个中介。

所以提到ThreadLocal，我们不当顾名思义的认为JDK里面提供ThreadLocal就是提供了一个用来封装本地线程存储的容器，它本身并没有Map那样的容器功能。真正发挥作用的是ThreadLocalMap。也就是说，事实上，采用ThreadLocal来提高并发，首先要理解，这不是一种简单的对象封装，而是一套机制，而这套机制中的三个关键因素（Thread、ThreadLocal、ThreadLocalMap）之间的关系是值得我们引起注意的。

***** 补疑完毕 *****

可见，要正确使用ThreadLocal，必须注意以下几点：

1. 总是对ThreadLocal中的initialValue()方法进行覆盖。
2. 当使用set()或get()方法时牢记这两个方法是对当前活动线程中的ThreadLocalMap进行操作，一定要认清哪个是当前活动线程！
3. 适当的使用泛型，可以减少不必要的类型转换以及可能由此产生的问题。

运行该程序，我们发现：程序的执行过程只需要5秒，而如果采用同步的方法，程序的执行结果相同，但执行时间需要15秒。以前是多个线程为了争取一个资源，不得不在同步规则的制约下互相谦让，浪费了一些时间。

现在，采用ThreadLocal机制以后，可用的资源多了，你有我有全都有，所以，每个线程都可以毫无顾忌的工作，自然就提高了并发性，线程安全也得以保证。

当今很多流行的开源框架也采用ThreadLocal机制来解决线程的并发问题。比如大名鼎鼎的 Struts 2.x 和 Spring 等。

把ThreadLocal这样的话题放在我们的同步机制探讨中似乎显得不是很合适。但是ThreadLocal的确为我们解决多线程的并发问题带来了全新的思路。它为每个线程创建一个独立的资源副本，从而将多个线程中的数据隔离开来，避免了同步所产生的性能问题，是一种“以空间换时间”的解决方案。

但这并不是说ThreadLocal就是包治百病的万能药了。如果实际的情况不允许我们为每个线程分配一个本地资源副本的话，同步还是非常有意义的。

好了，本系列到此马上就要划上一个圆满的句号了。不知大家有什么意见和疑问没有。希望看到你们的留言。

下一讲中我们就来对之前的内容进行一个总结，顺便讨论一下被遗忘的volatile关键字。敬

请期待。

发表于 2010-04-24 13:36 **X-SPIRIT** 阅读(4793) 评论(15) 编辑 收藏 所属分类: JAVA SE

评论

re: Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）

"总是对ThreadLocal中的initialValue()方法进行覆盖" ThreadLocal貌似不是这么用的，你现在为每个线程都 new 一个对象当然不会冲突，干脆在 ThreadDemo3 的构造方法中 new 好了，使用 ThreadLocal 干嘛？

pwl2014 评论于 2010-04-08 09:53 回复 更多评论

re: Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）

@pwl2014

这只是一个DEMO，使用new在这里其实就是一个打比方的方法。如果用StudentFactory也许更容易接受一点，呵呵。

在实际的应用中，我们经常可以看到一些用ThreadLocal来封装资源的例子。他们无一例外的都是做一件事情：创建新的资源，供线程使用。例如用ThreadLocal来处理JDBC 的Connection。即使你没有覆盖initialValue()方法，而是用先get再判空，再set的方式，也还是为一个没有获取到connection的线程创建一个connection。

所以，问题的关键不在于ThreadLocal使用的时候采用何种形式。ThreadLocal的思想精髓就是为每个线程创建独立的资源副本。而使用ThreadLocal的时候最重要的就是分清楚当前活动线程是哪个。

至于覆盖initialValue()方法的问题，这个应该是仁者见仁，智者见智的问题，我只是提出一个能够节约代码量的方案。一般情况下，覆盖initialValue()方法已经可以解决问题，这是一种最为经济的编码习惯，它不仅能够达到要求，并且和JDK的原生API结合的很好，不容易出错，当然如果你需要在用ThreadLocal处理资源之前做一些其他的处理，那就另当别论了。

X-Spirit 评论于 2010-04-08 11:07 回复 更多评论

re: Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）

嗯，不错！这一系列文章好。。。

anniezheng 评论于 2010-04-08 19:14 回复 更多评论

re: Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）

@anniezheng

谢谢夸奖啦

X-Spirit 评论于 2010-04-09 00:09 回复 更多评论

re: Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）

@X-Spirit

这句话说的好，ThreadLocal的思想精髓就是为每个线程创建独立的资源副本。你这个Demo根本没体现出来，每个线程里都有一个ThreadLocal对象；应该是一个ThreadLocal对象对应多个线程，你这变成了一对一，完全没体现出ThreadLocal的作用。

路过 评论于 2010-04-23 13:42 回复 更多评论

re: Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）【更新重要补疑】

学习了！

boiledwater 评论于 2010-05-18 18:18 回复 更多评论

re: Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）【更新重要补疑】

最近在网上看些关于线程的东西，开始在一个论坛上找到转载你的一片帖子，然后一路杀过来，找到了你的大本营，一口气看了你写这个6篇文章，感觉收获挺大的，我之前学习了struts2，也发现了它用这个东西，一直没有深究，在这里看到，觉得一下子发现了另一座山似的。但是我的疑问是ThreadLocal这个东西的使用场景，还是不太明确。

康博 评论于 2010-05-28 14:11 回复 更多评论

re: Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）【更新重要补疑】

@康博

感谢你的关注。关于应用场景的问题，我想没有什么非常好的例子可以给你。不过只能告诉你使用ThreadLocal的好处就是可以避免同步带来的性能损耗，并且，当多个线程同时使用同一个类的实例的时候，如果这个实例不是单例模式的一个实现，那么ThreadLocal就是值得考虑的。

X-Spirit 评论于 2010-06-07 00:54 回复 更多评论

re: Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）【更新重要补疑】

@X-Spirit

谢谢你的回复，期待你的下一篇文章。

康博 评论于 2010-06-12 15:23 回复 更多评论

re: Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）【更新重要补疑】

@康博

呵呵，这个系列基本上要结束了。但是由于我最近比较忙，所以本系列的总结还没有时间整理。不过过些日子会整理出来的。请保持关注。

X-Spirit 评论于 2010-06-12 19:55 回复 更多评论

re: Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）【更新重要补疑】

支持技术贴，软件也不错

圣光永恒 评论于 2010-07-07 18:23 回复 更多评论

re: Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）【更新重要补疑】

技术的确的支持，顶一下

朱少 评论于 2010-07-08 16:37 回复 更多评论

re: Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）【更新重要补疑】

好全啊
谢谢楼主

nauxiaoyao 评论于 2010-07-13 19:19 回复 更多评论

re: Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）【更新重要补疑】

好详细
了解了镇面貌了

nauxiaoyao 评论于 2010-07-13 19:39 回复 更多评论

re: Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）【更新重要补疑】

不错的编程 支持技术贴

朱少 评论于 2010-07-17 12:44 回复 更多评论

[新用户注册](#) [刷新评论列表](#)

[Autodesk诚聘C++开发工程师（上海）](#)

IT新闻：

- [MySpace联合创始人安德森盛赞Google+](#)
- [Ubuntu 的辉煌走到尽头了？](#)
- [BitTorrent诞生十周年](#)
- [工商总局将全面核查网店真实身份](#)
- [快递投诉答复期限缩减到7日](#)



京东商城
网购上京东 省钱又放心



百团大战 两百余商品 超值火爆
活动时间6.14-6.22

iike.360buy.com

Google 提供的广告

[博客园](#) [博问](#) [IT新闻](#) [Java程序员招聘](#)

标题

姓名

主页

验证码

*

6599

内容(请不要发表任何与政治相关的内容)

Remember Me?

[登录](#)

[使用Ctrl+Enter键可以直接提交]



推荐职位:

- [北京.NET 软件开发工程师\(澳大利亚SSW有限公司\)](#)
- [北京C# 软件工程师\(科胜永昌软件\)](#)
- [ASP.NET中高级软件工程师\(联展壹步科技\)](#)
- [WPF&Silverlight开发工程师\(北京\)\(盛安德科技\)](#)
- [诚招武汉.NET程序员\(武汉漫山网络\)](#)
- [广州.NET程序员\(南方宜信信息\)](#)
- [上海.NET 开发工程师\(盖世网络技术\)](#)
- [Software Engineer\(Web\)\(Autodesk\)](#)

博客园首页随笔:

- [\(原创\)测试趋势6曲线解读](#)
- [\(译\) iPhone上面的现实增强 \(Augmented Reality \) 入门教程](#)

- 在Orchard中使用Image Gallery模块
 - 《SQL Server 2008从入门到精通》读书笔记2：入门可以，精通差很多（黑体部分为新增内容）
 - 底层和高层-失去的平衡[我们要开发怎样的应用]
- 知识库：
- 快乐与自组织团队
 - 创业公司如何招聘优秀工程师
 - 写给即将入行的程序员的一封信
 - 动起来再调整 - 向项目经理推荐敏捷
 - HTML5之美

最简洁阅读版式：

Java 多线程同步问题的探究（五、你有我有全都有—— ThreadLocal如何解决并发安全性？）【更新重要补疑】

网站导航：

博客园 IT新闻 知识库 博客生活 IT博客网 C++ 博客 博问 管理

相关文章：

Java 多线程同步问题的探究（四、协作，互斥下的协作——Java多线程协作（wait、notify、notifyAll））

Java 多线程同步问题的探究（三、Lock来了，大家都让开【2. Fair or Unfair? It is a question...】）

Java 多线程同步问题的探究（三、Lock来了，大家都让开【1. 认识重入锁】）

Java 多线程同步问题的探究（二、给我一把锁，我能创造一个规矩）

Java多线程同步问题的探究（一、线程的先来后到）