



作者	正文
<b>mxdba321123</b> 等级: 初级会员    性别:  文章: 23 积分: 40 来自: 杭州 	<div>                         发表时间: 2010-10-06 最后修改: 2010-10-06                     </div> <hr/> <div>                         相关知识库:                          <a href="#">Android知识库</a>  <a href="#">语音识别与合成知识库</a>  <a href="#">计算机视觉知识库</a>  <a href="#">自然语言理解和处理知识库</a> </div> <div> <a href="#">Java综合</a> </div> <p>ThreadLocal&lt;T&gt;类在Spring, Hibernate等框架中起到了很大的作用, 对于其工作原理, 很多网上的文章分析的不够彻底, 甚至有些误解。</p> <p>首先, 为了解释ThreadLocal类的工作原理, 必须同时介绍与其工作甚密的其他几个类(内部类)</p> <ol style="list-style-type: none"> <li>1.ThreadLocalMap</li> <li>2.Thread</li> </ol> <div>  <b>ThreadLocal&lt;T&gt;</b> <ul style="list-style-type: none"> <li>◆ initialValue() : T</li> <li>● ThreadLocal()</li> <li>● get() : T</li> <li>■ setInitialValue() : T</li> <li>● set(T) : void</li> <li>● remove() : void</li> <li>▲ getMap(Thread) : ThreadLocalMap</li> <li>▲ createMap(Thread, T) : void</li> <li>▲ childValue(T) : T</li> <li>▲ ThreadLocalMap</li> <li>▶ Entry</li> </ul> </div> <p>可能有人会觉得Thread与ThreadLocal有什么关系, 其实真正的奥秘就在Thread类中的一行:</p> <div> <b>Java代码</b> ☆                 <pre>1. ThreadLocal.ThreadLocalMap threadLocals = null;</pre> </div> <p>其中ThreadLocalMap的定义是在ThreadLocal类中, 真正的引用却是在Thread类中</p> <p>那么ThreadLocalMap究竟是什么呢?</p> <p>可以看到这个类应该是一个Map,JDK的解释是</p> <p>写道</p> <div>                 ThreadLocalMap is a customized hash map suitable only for maintaining thread local values             </div> <p>接下来的重点是ThreadLocalMap中用于存储数据的entry</p> <div> <b>Java代码</b> ☆                 <pre>1. static class Entry extends WeakReference&lt;ThreadLocal&gt; { 2.     /** The value associated with this ThreadLocal. */</pre> </div>

相关文章:

- [正确理解ThreadLocal](#)
- [深入浅出ThreadLocal](#)
- [关于ThreadLocal的一些概念](#)

 推荐群组: [struts2](#)  
[更多相关推荐](#)

作者

Entry(ThreadLocal k, Object v) { 正文

```
5.
6.         super(k);
7.         value = v;
8.     }
9. }
```

从中我们可以发现这个Map的key是ThreadLocal变量，value为用户的值，并不是网上大多数的列子key是线程的名字或者标识

到这里，我们就可以理解ThreadLocal究竟是如何工作的了

- 1.Thread类中有一个成员变量叫做ThreadLocalMap，它是一个Map，他的Key是ThreadLocal类
- 2.每个线程拥有自己的申明为ThreadLocal类型的变量,所以这个类的名字叫'ThreadLocal': 线程自己的（变量）
- 3.此变量生命周期是由该线程决定的，开始于第一次初始（get或者set方法）
- 4.由ThreadLocal的工作原理决定了：每个线程独自拥有一个变量，并非共享或者拷贝

Java代码



```
1.  /**
2.   * @author mxdba
3.   *
4.   */
5.  public class ThreadLocalSample {
6.
7.      public static void main(String[] args) {
8.          ThreadTest test1 = new ThreadTest(10);
9.          ThreadTest test2 = new ThreadTest(20);
10.         test1.start();
11.         test2.start();
```

作者

正文

```
14.     }
15.
16.     /**
17.      * 此线程有两个ThreadLocal变量，但是由于ThreadLocal是延迟初始的，
18.      * 所以在debug时可以看到线程名为“线程20”的线程的ThreadLocalMap中没有thLcal2这个entry
19.      * @author mxdba
20.      *
21.      */
22.     class ThreadTest extends Thread {
23.
24.         public static ThreadLocal<Integer> thLocal = new ThreadLocal<Integer>();
25.         public static ThreadLocal<String> thLocal2 = new ThreadLocal<String>();
26.
27.         public Integer num;
28.
29.
30.
31.         public ThreadTest(Integer num) {
32.             super("线程" + num);
33.             this.num = num;
34.         }
35.
36.         @Override
37.         public void run() {
38.             Integer n = thLocal.get();
39.             if(num != 20) {
40.                 String s = thLocal2.get();
41.             }
42.
43.             if(n == null) {
44.                 thLocal.set(num);
45.             }
46.             System.out.println(thLocal.get());
47.         }
48.
49.     }
```

接下来分析一下源码，就更加清楚了

关键方法代码



```
1.     /**
2.      * 关键方法，返回当前Thread的ThreadLocalMap
3.      * [[[每个Thread返回各自的ThreadLocalMap，所以各个线程中的ThreadLocal均为独立的]]]
4.      */
5.     ThreadLocalMap getMap(Thread t) {
6.         return t.threadLocals;
7.     }
```

Threadlocal的get方法代码



```
1.     public T get() {
2.         Thread t = Thread.currentThread();
3.         /**
4.          * 得到当前线程的ThreadLocalMap
5.          */
6.         ThreadLocalMap map = getMap(t);
7.         if (map != null) {
8.             /**
9.              * 在此线程的ThreadLocalMap中查找key为当前ThreadLocal对象的entry
10.             */
11.             ThreadLocalMap.Entry e = map.getEntry(this);
12.             if (e != null)
13.                 return (T)e.value;
14.         }
15.         return setInitialValue();
16.     }
```

作者

正文

初始化方法代码 ☆

```
1. private T setInitialValue() {
2.     /**
3.      * 默认返回null, 这个方法为protected可以继承
4.      */
5.     T value = initialValue();
6.     Thread t = Thread.currentThread();
7.     ThreadLocalMap map = getMap(t);
8.     if (map != null)
9.         map.set(this, value);
10.    else
11.        /**
12.         * 初次创建
13.         */
14.        createMap(t, value);
15.    return value;
16. }
```

Java代码 ☆

```
1. /**
2.  * 给当前thread初始ThreadlocalMap
3.  */
4. void createMap(Thread t, T firstValue) {
5.     t.threadLocals = new ThreadLocalMap(this, firstValue);
6. }
```

通过上边的分析,我们发现, ThreadLocal类的使用虽然是用来解决多线程的问题的,但是还是有很明显的针对性

1.最明显的, ThreadLocal变量的活动范围为某线程,并且我的理解是该线程“专有的,独自霸占”,对该变量的所有操作均有该线程完成!也就是说, ThreadLocal不是用来解决共享,竞争问题的。典型的应用莫过于Spring, Hibernate等框架中对于多线程的处理了

Java代码 ☆

```
1. private static final ThreadLocal threadSession = new ThreadLocal();
2.
3. public static Session getSession() throws InfrastructureException {
4.     Session s = (Session) threadSession.get();
5.     try {
6.         if (s == null) {
7.             s = getSessionFactory().openSession();
8.             threadSession.set(s);
9.         }
10.    } catch (HibernateException ex) {
11.        throw new InfrastructureException(ex);
12.    }
13.    return s;
14. }
```

这段代码,每个线程有自己的ThreadLocalMap,每个ThreadLocalMap中根据需要初始加载threadSession,这样的好处就是介于singleton与prototype之间,应用singleton无法解决线程,应用prototype开销又太大,有了ThreadLocal之后就好了,对于需要线程“霸占”的变量用ThreadLocal,而该类实例的方法均可以共享。

2.关于内存泄漏:

虽然ThreadLocalMap已经使用了weakReference,但是还是建议能够显示的使用remove方法。

作者

声明: ITeYe 文章版权属于作者, 受法律保护, 没有作者书面许可不得转载。正文

推荐链接

返回顶楼

[主页](#) [资料](#) [短信](#) [留言](#) [关注](#)

躁动的绵羊

等级: 初级会员



性别:

文章: 85

积分: 50

来自: 北京

我现在离线

发表时间: 2010-10-07

必须得支持

对我有帮助的东西, 我一般都投精华票来回报贴主

返回顶楼

[主页](#) [资料](#) [短信](#) [留言](#) [关注](#) 回帖地址

0 0 请登录后投票

xiarilian12

等级: 初级会员



性别:

文章: 8

积分: 40

来自: 西安

我现在离线

发表时间: 2010-10-08

先shoucang一下。。。

返回顶楼

[主页](#) [资料](#) [短信](#) [留言](#) [关注](#) 回帖地址

0 0 请登录后投票

ikeycn

等级: 初级会员



性别:

文章: 10

积分: 0

来自: 杭州

我现在离线

发表时间: 2010-10-08

昨天刚看了一些关于ThreadLocal的东西, 再加上这个总结, 更了解了

返回顶楼

[主页](#) [资料](#) [短信](#) [留言](#) [关注](#) 回帖地址

0 0 请登录后投票

kswwhyk

等级: 初级会员



发表时间: 2010-10-08

看了楼主的帖子让我对ThreadLocal有了更深的了解!

性别: 

锁定老帖子 主题: ThreadLocal-分析-总结

精华帖 (18):: 良好帖 (7):: 新手帖 (2):: 隐藏帖 (0)

文章: 13 作者  
积分: 30  
来自: 深圳

正文

 我现在离线

返回顶楼

 主页

 资料

 短信

 留言

 关注

回帖地址



0



0

请登录投票

windzhq

等级: 初级会员

发表时间: 2010-10-08

ITEYE

性别: 

文章: 2

积分: 30

来自: 成都

 我现在离线

说得简单易懂，让人一下就理清Thread和Threadlocal的关系

返回顶楼

 主页

 资料

 短信

 留言

 关注

回帖地址



0



0

请登录投票

41897179

等级: 初级会员

发表时间: 2010-10-08

很好，刚看了下struts里的相关应用，再看这个很受用啊




性别: 

文章: 1

积分: 30

来自: 深圳

 我现在离线

 主页

 资料

 短信

 留言

 关注

回帖地址



0



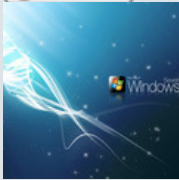
0

请登录投票

msi110

等级: 初级会员

发表时间: 2010-10-08



貌似ThreadLocal在JDK1.5以后才出现的么？

性别: 

文章: 23

积分: 0

来自: 成都

 我现在离线

返回顶楼

 主页

 资料

 短信

 留言

 关注

回帖地址



0



0

请登录投票

mxdba321123

等级: 初级会员

发表时间: 2010-10-08

ITEYE

msi110 写道

貌似ThreadLocal在JDK1.5以后才出现的么？

ThreadLocal 在JDK1.2时就已经出现了吗

锁定老帖子 主题: ThreadLocal 分析-总结

精华帖 (18):: 良好帖 (7):: 新手帖 (2):: 隐藏帖 (0)

性别:

文章: 23

积分: 40

来自: 杭州

我现在离线

返回顶楼

正文

主页

资料

短信

留言

关注

回帖地址

0

0 请登录后投票

jiangzhouyun

等级: 初级会员

性别:

文章: 54

积分: 30

来自: 杭州

我现在离线

发表时间: 2010-10-08

抄别人的吧

主页

资料

短信

留言

关注

回帖地址

0

0 请登录后投票