

联系我们



请扫描二维码联系客服
✉ webmaster@csdn.net
☎ 400-660-0108
💬 QQ客服 🗨 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心



等级：**博客 4** 访问量：12.31万
积分：1702 排名：2.81万

文章搜索

- 文章分类
- java (58)

算法 (11)

object C (10)

SQL (11)

设计模式 (2)

spring (19)

JavaScript (4)

IntelliJ (4)

Git (2)

hive (2)

UML (2)

HTTP (7)

Linux (3)

web-xml (4)

感悟 (2)

guava (1)

redis (1)

- 文章存档
- 2018年1月 (1)

2017年10月 (1)



2017年9月 (2)

2017年8月 (1)

2017年7月 (2)
- 展开



- 阅读排行
- java 异常处理 Throwable Err... (14320)

Java Lombok @Data @Buil... (10874)

☰ 目录视图 ☰ 摘要视图  **1**  订阅

java 深入理解ThreadLocal

🔖 标签：**java** **多线程**

2017年01月07日 13:49:14 964人阅读 评论(0)  收藏  举报

分类：

java (57) ▾

目录(?) [\[+\]](#)

相信读者在网上也看了很多关于ThreadLocal的资料，很多博客都这样说：ThreadLocal为解决多线程程序的并发问题提供了一种新的思路；ThreadLocal的目的是为了解决多线程访问资源时的共享问题。如果你也这样认为的，那现在给你10秒钟，清空之前对ThreadLocal的错误认知！

看看JDK中的源码是怎么写的：

This class provides thread-local variables. These variables differ from their normal counterparts in that each thread that accesses one (via its {`@code get`} or {`@code set`} method) has its own, independently initialized copy of the variable. {`@code ThreadLocal`} instances are typically private static fields in classes that wish to associate state with a thread (e.g., a user ID or Transaction ID).

翻译过来大概是这样的(英文不好，如有更好的翻译，请留言说明)：

ThreadLocal类用来提供线程内部的局部变量。这种变量在多线程环境下访问(通过get或set方法访问)时能保证各个线程里的变量相对独立于其他线程内的变量。**ThreadLocal实例通常来说都是private static类型的，用于关联线程和线程的上下文。**

可以总结为一句话：ThreadLocal的作用是提供线程内的局部变量，这种变量在线程的生命周期内起作用，减少同一个线程内多个函数或者组件之间一些公共变量的传递的复杂度。

举个例子，我出门需要先坐公交再做地铁，这里的坐公交和坐地铁就好比是同一个线程内的两个函数，我就是个线程，我要完成这两个函数都需要同一个东西：公交卡（北京公交和地铁都使用公交卡），那么我为了不向这两个函数都传递公交卡这个变量（相当于不是一直带着公交卡上路），我可以这么做：**将公交卡事先交给一个机构，当我需要刷卡的时候再向这个机构要公交卡（当然每次拿的都是同一张公交卡）。这样就能达到只要是我(同一个线程)需要公交卡，何时何地都能向这个机构要的目的。**

有人要说了：你可以将公交卡设置为全局变量啊，这样不是也能何时何地都能取公交卡吗？但是如果有很多个人（很多个线程）呢？大家可不能都使用同一张公交卡吧(我们假设公交卡是实名认证的)，这样不就乱套了嘛。现在明白了吧？这就是ThreadLocal设计的初衷：提供线程内部的局部变量，在本线程内随时随地可取，隔离其他线程。

ThreadLocal基本操作

构造函数

ThreadLocal的构造函数签名是这样的：

```
/**  
 * Creates a thread local variable.
```

http://blog.csdn.net/mccand1234/article/details/54173084

1/13

LEFT JOIN 和JOIN 多表连接 (6585)

联系我们



请扫描二维码联系客服
✉ webmaster@csdn.net
☎ 400-660-0108
💬 QQ客服 💬 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

dominghao : 说的很详细,思路也很清晰...t hanks

java 异常处理 Throwab...
DD_Davina : 如果可以整理一下排版会更好。

```
* @see #withInitial(java.util.function.Supplier)
*/
public ThreadLocal() {
}
内部啥也没做。
```

initialValue函数

initialValue函数用来设置ThreadLocal的初始值，函数签名如下：

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

该函数在调用get函数的时候会第一次调用，但是如果一开始就调用了set函数，则该函数不会被调用。通常该函数只会被调用一次，除非手动调用了remove函数之后又调用get函数，这种情况下，get函数中还是会调用initialValue函数。该函数是protected类型的，很显然是建议在子类重载该函数的，所以通常该函数都会以匿名内部类的形式被重载，以指定初始值，比如：

```
1 package com.winwill.test;
2 /**
3  * @author qifuguang
4  * @date 15/9/2 00:05
5  */
6 public class TestThreadLocal {
7     private static final ThreadLocal<Integer> value = new ThreadLocal<Integer>() {
8         @Override
9         protected Integer initialValue() {
10             return Integer.valueOf(1);
11         }
12     };
13 }
```

get函数

该函数用来获取与当前线程关联的ThreadLocal的值，函数签名如下：

```
public T get()
```

如果当前线程没有该ThreadLocal的值，则调用initialValue函数获取初始值返回。

set函数

set函数用来设置当前线程的该ThreadLocal的值，函数签名如下：

```
public void set(T value)
```

设置当前线程的ThreadLocal的值为value。

remove函数

remove函数用来将当前线程的ThreadLocal绑定的值删除，函数签名如下：

```
public void remove()
```

在某些情况下需要手动调用该函数，防止内存泄露。

代码演示

学习了最基本的操作之后，我们用一段代码来演示ThreadLocal的用法，该例子实现下面这个场景：

有5个线程，这5个线程都有一个值value，初始值为0，线程运行时用一个循环往value值相加数字。


1





联系我们



请扫描二维码联系客服
webmaster@csdn.net
400-660-0108
QQ客服 客服论坛

关于 招聘 广告服务 百度
©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

代码实现：

```
1 package com.winwill.test;
2 /**
3  * @author qifuguang
4  * @date 15/9/2 00:05
5  */
6 public class TestThreadLocal {
7     private static final ThreadLocal<Integer> value = new ThreadLocal<Integer>() {
8         @Override
9         protected Integer initialValue() {
10             return 0;
11         }
12     };
13     public static void main(String[] args) {
14         for (int i = 0; i < 5; i++) {
15             new Thread(new MyThread(i)).start();
16         }
17     }
18     static class MyThread implements Runnable {
19         private int index;
20         public MyThread(int index) {
21             this.index = index;
22         }
23         public void run() {
24             System.out.println("线程" + index + "的初始value:" + value.get());
25             for (int i = 0; i < 10; i++) {
26                 value.set(value.get() + i);
27             }
28             System.out.println("线程" + index + "的累加value:" + value.get());
29         }
30     }
31 }
```



执行结果为：

线程0的初始value:0
线程3的初始value:0
线程2的初始value:0
线程2的累加value:45
线程1的初始value:0
线程3的累加value:45
线程0的累加value:45
线程1的累加value:45
线程4的初始value:0
线程4的累加value:45

可以看到，各个线程的value值是相互独立的，本线程的累加操作不会影响到其他线程的值，真正达到了线程内部隔离的效果。

如何实现的

看了基本介绍，也看了最简单的效果演示之后，我们更应该好好研究下ThreadLocal内部的实现原理。如果给你设计，你会怎么设计？相信大部分人会有这样的想法：

每个ThreadLocal类创建一个Map，然后用线程的ID作为Map的key，实例对象作为Map的value，这样就能达到各个线程的值隔离的效果。

没错，这是最简单的设计方案，JDK最早期的ThreadLocal就是这样设计的。JDK1.3（不确定是否是1.3）之后ThreadLocal的设计换了一种方式。

我们先看看JDK8的ThreadLocal的get方法的源码：

联系我们



请扫描二维码联系客服
webmaster@csdn.net
400-660-0108
QQ客服 客服论坛

关于 招聘 广告服务 百度
©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

```
1 public T get() {
2     Thread t = Thread.currentThread();
3     ThreadLocalMap map = getMap(t);
4     if (map != null) {
5         ThreadLocalMap.Entry e = map.getEntry(this);
6         if (e != null) {
7             @SuppressWarnings("unchecked")
8             T result = (T)e.value;
9             return result;
10        }
11    }
12    return setInitialValue();
13 }
```



其中getMap的源码：

```
1 ThreadLocalMap getMap(Thread t) {
2     return t.threadLocals;
3 }
4 setInitialValue函数的源码：
5
6 private T setInitialValue() {
7     T value = initialValue();
8     Thread t = Thread.currentThread();
9     ThreadLocalMap map = getMap(t);
10    if (map != null)
11        map.set(this, value);
12    else
13        createMap(t, value);
14    return value;
15 }
```

createMap函数的源码：

```
1 void createMap(Thread t, T firstValue) {
2     t.threadLocals = new ThreadLocalMap(this, firstValue);
3 }
```

简单解析一下，get方法的流程是这样的：

- 1. 首先获取当前线程
 - 2. 根据当前线程获取一个Map
 - 3. 如果获取的Map不为空，则在Map中以ThreadLocal的引用作为key来在Map中获取对应的value e，否则转到5
 - 4. 如果e不为null，则返回e.value，否则转到5
 - 5. Map为空或者e为空，则通过initialValue函数获取初始值value，然后用ThreadLocal的引用和value作为firstKey和firstValue创建一个新的Map
- 然后需要注意的是Thread类中包含一个成员变量：

ThreadLocal.ThreadLocalMap threadLocals = null;

所以，可以总结一下ThreadLocal的设计思路：

每个Thread维护一个ThreadLocalMap映射表，这个映射表的key是ThreadLocal实例本身，value是真正需要存储的Object。

这个方案刚好与我们开始说的简单的设计方案相反。查阅了一下资料，这样设计的主要有以下几点优势：

- 1. 这样设计之后每个Map的Entry数量变小了：之前是Thread的数量，现在是ThreadLocal的数量，能提高性能，据说性能的提升不是一点两点(没有亲测)
 - 2. 当Thread销毁之后对应的ThreadLocalMap也就随之销毁了，能减少内存使用量。
- 再深入一点

联系我们



请扫描二维码联系客服
webmaster@csdn.net
400-660-0108
QQ客服 客服论坛

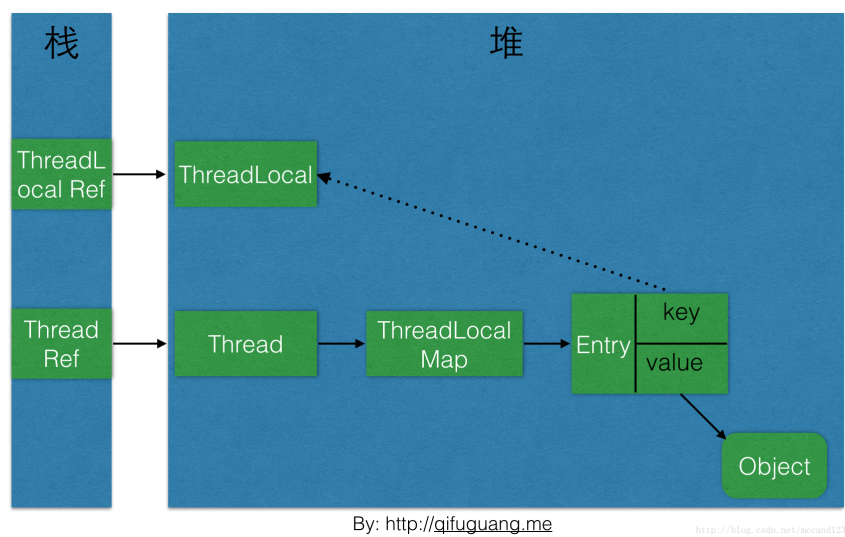
关于 招聘 广告服务 百度
©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

先交代一个事实：ThreadLocalMap是使用ThreadLocal的弱引用作为Key的：

```
1 static class ThreadLocalMap {
2     /**
3      * The entries in this hash map extend WeakReference, using
4      * its main ref field as the key (which is always a
5      * ThreadLocal object). Note that null keys (i.e. entry.get()
6      * == null) mean that the key is no longer referenced, so it
7      * entry can be expunged from table. Such entries are referred
8      * as "stale entries" in the code that follows.
9      */
10    static class Entry extends WeakReference<ThreadLocal<?>> {
11        /** The value associated with this ThreadLocal. */
12        Object value;
13        Entry(ThreadLocal<?> k, Object v) {
14            super(k);
15            value = v;
16        }
17    }
18    ...
19    ...
20 }
```

下图是本文介绍到的一些对象之间的引用关系图，实线表示强引用，虚线表示弱引用：



然后网上就传言，ThreadLocal会引发内存泄露，他们的理由是这样的：
如上图，ThreadLocalMap使用ThreadLocal的弱引用作为key，如果一个ThreadLocal没有外部强引用引用他，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 -> ThreadLocalMap -> Entry -> value
永远无法回收，造成内存泄露。

我们来看看到底会不会出现这种情况。
其实，在JDK的ThreadLocalMap的设计中已经考虑到这种情况，也加上了一些防护措施，下面是ThreadLocalMap的getEntry方法的源码：

```
1 private Entry getEntry(ThreadLocal<?> key) {
2     int i = key.threadLocalHashCode & (table.length - 1);
3     Entry e = table[i];
4     if (e != null && e.get() == key)
5         return e;
6     else
7         return getEntryAfterMiss(key, i, e);
8 }
```

联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

💬 QQ客服 🗨 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

```

8  }
9  getEntryAfterMiss函数的源码:
10
11 private Entry getEntryAfterMiss(ThreadLocal<?> key, int i, Entry e) {
12     Entry[] tab = table;
13     int len = tab.length;
14     while (e != null) {
15         ThreadLocal<?> k = e.get();
16         if (k == key)
17             return e;
18         if (k == null)
19             expungeStaleEntry(i);
20         else
21             i = nextIndex(i, len);
22         e = tab[i];
23     }
24     return null;
25 }
26 expungeStaleEntry函数的源码:
27
28 private int expungeStaleEntry(int staleSlot) {
29     Entry[] tab = table;
30     int len = tab.length;
31     // expunge entry at staleSlot
32     tab[staleSlot].value = null;
33     tab[staleSlot] = null;
34     size--;
35     // Rehash until we encounter null
36     Entry e;
37     int i;
38     for (i = nextIndex(staleSlot, len);
39          (e = tab[i]) != null;
40          i = nextIndex(i, len)) {
41         ThreadLocal<?> k = e.get();
42         if (k == null) {
43             e.value = null;
44             tab[i] = null;
45             size--;
46         } else {
47             int h = k.threadLocalHashCode & (len - 1);
48             if (h != i) {
49                 tab[i] = null;
50                 // Unlike Knuth 6.4 Algorithm R, we must scan until
51                 // null because multiple entries could have been stale
52                 while (tab[h] != null)
53                     h = nextIndex(h, len);
54                 tab[h] = e;
55             }
56         }
57     }
58     return i;
59 }

```



1



整理一下ThreadLocalMap的getEntry函数的流程：

首先从ThreadLocal的直接索引位置(通过ThreadLocal.threadLocalHashCode & (len-1)运算得到)获取Entry e，如果e不为null并且key相同则返回e；

如果e为null或者key不一致则向下一个位置查询，如果下一个位置的key和当前需要查询的key相等，则返回对应的Entry，否则，如果key值为null，则擦除该位置的Entry，否则继续向下一个位置查询

在这个过程中遇到的key为null的Entry都会被擦除，那么Entry内的value也就没有强引用链，自然会被回收。仔细研究代码可以发现，set操作也有类似的思想，将key为null的这些Entry都删除，防止内存泄露。

但是光这样还是不够的，上面的设计思路依赖一个前提条件：要调用ThreadLocalMap的

联系我们



请扫描二维码联系客服
webmaster@csdn.net
400-660-0108
QQ客服 客服论坛

关于 招聘 广告服务 百度
©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

getEntry函数或者set函数。这当然是不可能任何情况都成立的，所以很多情况下需要使用者手动调用ThreadLocal的remove函数，手动删除不再需要的ThreadLocal，防止内存泄露。所以JDK建议将ThreadLocal变量定义成private static的，这样的话ThreadLocal的生命周期就更长，由于一直存在ThreadLocal的强引用，所以ThreadLocal也就不会被回收，也就能保证任何时候都能根据ThreadLocal的弱引用访问到Entry的value值，然后remove它，防止内存泄露。

Web服务器中线程池的状态问题



Web服务在创建线程的过程中，频繁的创建线程对系能的影响巨大，故很多服务采用了线程池的方式来解决线程不断创建锁导致的问题，所以在使用线程池的过程中，由于线程是不断的回收和利用故ThreadLocal在服务器中也是被反复利用的，在使用中如果不做清理操作，很容易导致变量污染。尽管对于很多服务器来说，ThreadLocal的确是相对于每个线程，每个线程会有自己的ThreadLocal。但考虑到服务器都会维护一套线程池。因此，不同用户访问，可能会接受到同样的线程。因此，在做基于ThreadLocal时，需要谨慎，避免出现ThreadLocal变量的缓存，导致其他线程访问到本线程变量，如果运用不当，会导致系统效率低下，举个例子，假设我们得系统在访问的时候在ThreadLocal中加入变量不予以更新和删除，则这个保存的对象就变成一个增量的容器对象，如果访问量巨大，将导致jvm内存不足而频繁触发gc，gc在工作的时候会进行数据复制，频繁的触发gc对系统的性能会带来不利影响，同时还有可能导致内存溢出。

遇到的问题

```
1 public class ContextHolder {
2
3     private static ThreadLocal<UserContext> userContext = new InheritableThreadLocal<>();
4
5     public static UserContext getUserContext(){
6         if(userContext.get() == null){
7             userContext.set(new UserContext());
8         }
9         return userContext.get();
10    }
11
12    public static void setContext(UserContext context) {
13        userContext.set(context);
14    }
15
16    public static void clear(){
17        userContext.remove();
18    }
19 }
```

如果每个线程（访问请求）在结束的时候没有调用clear方法的时候，其他线程再访问，就会造成线程污染，即拿到了其他线程的变量。

参考：

[http://qifuguang.me/2015/09/02/\[Java%E5%B9%B6%E5%8F%91%E5%8C%85%E5%A4%A6%E4%B9%A0%E4%B8%83\]%E8%A7%A3%E5%AF%86ThreadLocal/](http://qifuguang.me/2015/09/02/[Java%E5%B9%B6%E5%8F%91%E5%8C%85%E5%A4%A6%E4%B9%A0%E4%B8%83]%E8%A7%A3%E5%AF%86ThreadLocal/)

<http://blog.csdn.net/chichengit/article/details/7994712>

<http://blog.csdn.net/lufeng20/article/details/24314381>

<http://www.importnew.com/22039.html>

<http://www.importnew.com/22046.html>

- 上一篇 Java 强引用、弱引用、软引用、虚引用
- 下一篇 Spring log4j配置详解

联系我们



请扫描二维码联系客服
webmaster@csdn.net
400-660-0108
QQ客服 客服论坛

关于 招聘 广告服务 百度
©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心



ThreadLocal理解与使用



u012481172 2015年09月01日 13:12 636

在看FrameWork源码时，在ActivityThread类中有一个ThreadLocal变量，是这么定义的：static final ThreadLocal sThreadLocal = new...



1

ThreadLocal使用场景



indexchen 2007年07月11日 14:18 1064

在Java的多线程编程中，为保证多个线程对共享变量的安全访问，通常会使用synchronized，同一时刻只有一个线程对共享变量进行操作。但在有些情况下，synchronized不能保证多线程对...



深入理解ThreadLocal



zdp072 2014年09月09日 14:22 2547

一. ThreadLocal是什么？ ThreadLocal，顾名思义，它不是一个线程，而是线程的一个本地化对象。当工作于多线程中的对象使用ThreadLocal维护变量时，ThreadLocal为每...

Java中 ThreadLocal用法 - 个人实用总结



qsc0624 2015年06月25日 09:39 651

在多线程中同时被使用的类中使用ThreadLocal，能保证每个线程中有一个单独的对象，互不影响。用法如下：假如A类在多个线程中同时出现了（不管使用的是它的new对象还是用的静态方法），假如需要...

Java中的ThreadLocal对象



danchu 2017年04月04日 23:32 2463

1.什么是ThreadLocal 根据JDK文档中的解释：ThreadLocal的作用是提供线程内的局部变量，这种变量在多线程环境下访问时能够保证各个线程里变量的独立性。 从这里可以看出，...

彻底理解ThreadLocal



lufeng20 2014年04月22日 16:59 346665

ThreadLocal是什么 早在JDK 1.2的版本中就提供java.lang.ThreadLocal，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...

Java线程(篇外篇)：线程本地变量ThreadLocal



ghsau 2013年11月13日 21:37 44882

首先说明ThreadLocal存放的值是线程内共享的，线程间互斥的，主要用于线程内共享一些数据，避免通过参数来传递，这样处理后，能够优雅的解决一些实际问题，比如Hibernate中的OpenSessi...

Java并发编程（四）未处理异常、线程池和ThreadLocal类



huaxun66 2017年10月25日 10:34 330

未处理异常以我们通常的经验，如果线程执行过程中抛出了未处理异常（没有用try-catch），那么我们的APP就会崩溃，并且我们可以从Error Log中看到出错的异常堆栈信息。那么我们有没有方法，在异...

跟我学Java多线程——ThreadLocal

本篇文章讲解了ThreadLocal是什么，通过一个简单的demo来说明了ThreadLocal在同一线程中实现了线程内的数据共享，不同线程间我们实现了数据的隔离性，接下来通过一步步的去读Thread...

联系我们



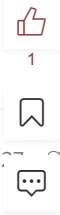
请扫描二维码联系客服
✉ webmaster@csdn.net
☎ 400-660-0108
💬 QQ客服 💬 客服论坛

关于 招聘 广告服务 百度
©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

java多线程并发控制之ThreadLocal

下面是ThreadLocal的测试代码，更多信息 maguanghai_2012 2017年02月11日 11:41 799
请参考注释 package com.jadyer.thread.l
ocal; import java.util.Rand...



Java8 ThreadLocal类源码 详解

u013803262 2016年11月22日 18:~ 754
Java8 ThreadLocal类源码 详解

利用ThreadLocal实现全局上下文工具类

/** * 全局上下文工具类,用于储存一些东西 */ publi flysun3344 2017年02月07日 15:36 591
c class MyContext { private static final ThreadL
ocal> mycontext...

java多线程--深入理解threadlocal以及适用场景

如何使用： 简介： JDK 1.2的版本中就提供java.l sean417 2017年04月10日 15:51 837
ang.ThreadLocal，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...

Java ThreadLocal

HackerSaillen 2015年06月09日 11:11 1897
背景： 最近项目中需要调用其他业务系统的服务，使用的是Java的RMI机制，但是在调用过程中中间
件发生了Token校验问题。而这个问题的根源是每次用户操作，没有去set Token导致...

详解Java中的ThreadLocal、ThreadLocalMap和Thread之间的关系

每个ThreadLocal实例都有一个唯一的threadLocalHashCode（这个值将会用于在ThreadLocalMap中找到ThreadLocal对应的value值），它是通过静态变量nex...
 woshiliufeng 2015年11月18日 02:49 3718

ThreadLocal父子线程传递实现方案

前言 介绍InheritableThreadLocal之前，假设对 a837199685 2016年09月30日 17:06 5906
ThreadLocal 已经有了一定的理解，比如基本概
念，原理，如果没有，可以参考：ThreadLocal源码分析解密.在讲解...

联系我们



请扫描二维码联系客服
✉ webmaster@csdn.net
☎ 400-660-0108
💬 QQ客服 🗨 客服论坛

关于 招聘 广告服务 百度
©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

深入剖析ThreadLocal实现原理以及内存泄漏问题

ThreadLocal ; 2017京东校园招聘笔试 LHQJ1992 2016年09月06日 17:19 21506

ThreadLocal 内部实现、应用场景和内存泄漏

一、什么是ThreadLocal 首先明确一个概念，那 u012834750 2017年05月11日 17:30 2261
就是ThreadLocal并不是用来并发控制访问一个
共同对象，而是为了给每个线程分配一个只属于该线程的变量，顾名思义它是local varia...



理解ThreadLocal

qjyong 2008年03月08日 10:59 3017
ThreadLocal是什么早在JDK 1.2的版本中就提供java.lang.ThreadLocal，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...

ThreadLocal实现方式&使用介绍---无锁化线程封闭

虽然现在可以说很多程序员会用ThreadLocal，但 xieyuooo 2013年02月21日 16:51 8245
是我相信大多数程序员还不知道ThreadLocal，而
使用ThreadLocal的程序员大多只是知道其然而不知其所以然，因此，使用ThreadLo...

Android的消息机制之ThreadLocal的工作原理

提到消息机制大家应该都不陌生，在日常 singwhatiwanna 2015年09月10日 21:54 24858
开发中不可避免地要涉及到这方面的内
容。从开发的角度来说，Handler是Android消息机制的上层接口，这使得开发过程中只需要和Handler交互即可。Handl...

ThreadLocal的正确用法

vking_wang 2013年11月06日 12:36 38719
ThreaLocal的JDK文档中说明：ThreadLocal instances are typically private static fields in classes that wish t...

线程的私家领地:ThreadLocal

GoGLETech 2018年02月04日 09:39 89
原创 2018-01-31 老刘 码农翻身 张大胖上午遇到了一个棘手的问题，他在一个AccountService中写了一段类似这样的代码： 然后这个AccountService 调用了其他Ja...

二叉排序树、红黑树、AVL树最简单的理解

前言[为什么写这篇]之前在知乎上看过一个提问： linshijun33 2016年12月04日 17:56 3252
为什么红黑树比AVL树用的场景更为广泛，红黑树
在 STL 和 Linux 都有一定的运用。而AVL树也在 Windows进程地址空间管理 中得到了使用。既...

JAVA多线程实现的三种方式及内部原理

JAVA多线程实现方式主要有三种：继承Thr honghailiang888 2016年06月16日 14:42 2437
ead类、实现Runnable接口、使用Executo
rService、Callable、Future实现有返回结果的多线程。其中前两种方式线程执行完后都没...

联系我们



请扫描二维码联系客服
webmaster@csdn.net
400-660-0108
QQ客服 客服论坛

关于 招聘 广告服务 百度
©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

ThreadLocal

fychung 2012年09月13日 00:11 674
原文地址：http://javapapers.com/core-java/threadlocal/ ThreadLocal 每个线程通过thr... al的se
t，get方法来访问它独... 1

elasticsearch中文分词

huwei2003 2014年12月07日 21:46 15958
由于elasticsearch基于lucene，所以天然地就多了许多lucene上的中文分词的支持，比如 IK, Paoding, M
MSEG4J等lucene中文分词原理上都能在elasticsea...

ES中的分词器

xiaomin1991222 2016年03月10日 16:05 1102
一、概念介绍 全文搜索引擎会用某种算法对要建索引的文档进行分析，从文档中提取出若干Token(词
元)，这些算法称为Tokenizer(分词器)，这些Token会被进一步处理，比如转成...

索引基础——B-Tree、B+Tree、红黑树、B*Tree数据结构

zhangliangzi 2016年05月10日 23:10 6590
B树 (B-Tree，并不是B“减”树，横杠为连接符，容易被误导) 是一种多路搜索树 (并
不是二叉的)： 1.定义任意非叶子结点最多只有M个儿子；且M>2； ...

Mysql的BTree索引的原则和限制

xiao2shiqi 2017年01月06日 18:06 776
这是自己在阅读《高性能MySQL》所做的笔记，和大家分享下 当人们谈论索引时，没有特别指明类型，那
么多半就是B-Tree索引，它使用树形结构来存储数据，大多数MySQL引擎都支持这种索引类型，比如...

MySQL优化之BTree索引使用规则

lipc_ 2016年10月18日 23:36 978
MySQL优化之BTree索引使用规则 从一道题开始分析：假设某个表有一个联合索引 (c1,c2,c3,c4) 一下
——只能使用该联合索引的c1,c2,c3部分 A where c1=x and c...

threadlocal原理及常用应用场景

sonny543 2016年05月07日 10:09 14280
想必很多朋友对ThreadLocal并不陌生，今天我们就来一起探讨下ThreadLocal的使用方法和实现原理。首
先，本文先谈一下对ThreadLocal的理解，然后根据ThreadLocal类的源码...

Mysql-索引-BTree类型【精简】

ty_hf 2016年12月08日 23:37 4058
网络上看了很多关于B-TREE的总结，b树,B-树,B+树,B*树(艾玛怎么还4个呢？都快蒙圈了呢)，有的真的很
精彩令人佩服，但是都是篇幅太长啊，一大长段的文字就让人望而生畏啊。干脆做一个简化版的...

一起写RPC框架（五）RPC网络模块的搭建三 序列化

linuu 2016年10月19日 14:30 1895
说到序列化，这在RPC的层面上也是很重要的一个环
节，因为在我们的业务层面，你传输的一个对象，是一
个Object，不过在网络上，却不管你传输的是Obj1，还是Obj2，网络只认byte，所以在代码层面上...

联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

💬 QQ客服 🗨 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

ThreadLocal工作原理



androidzhaoxiaogang

2011年09月19日 01:54 14690

1.概述 ThreadLocal为我们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。 ThreadLocal很容易让人望文生义，想当...

ThreadLocal遇到线程池时, 各线程间的数据会互相干扰, 串来串去



最近遇到一个比较隐蔽而又简单的问题,在使用Threa



cxh5060

2015年10月20日 12:16 2397

dLocal时发现出现多个线程中值串来串去,排查一番,

确定问题为线程池的问题,线程池中的线程是会重复利用的,而ThreadLocal是用线程来做Ke...



ThreadLocal工作原理



imzoer

2012年12月05日 21:03 20973

在这篇文章中,总结了一下面试过程中遇到的关于ThreadLocal的内容。总体上说,这样回答,面试算是过得去了。但是,这样的回答,明显仅仅是背会了答案,而没有去研究ThreadLocal的最根本的实现...

ThreadLocal的简单例子



yemaozi2009

2014年09月11日 21:56 3278

```
package com.bbwl.threadlocal; public class SequenceNumber { private static ThreadLocal seqNum ...
```

python学习——ThreadLocal



youzhouliu

2016年07月17日 11:17 598

在多线程环境下,每个线程都有自己的数据。一个线程使用自己的局部变量比使用全局变量好,因为局部变量只有线程自己能看见,不会影响其他线程,而全局变量的修改必须加锁。但是局部变量也有问题,就是在函数...

Python学习之ThreadLocal



pp634077956

2016年03月13日 13:59 4104

[0]:首先引出问题:我们在使用线程的时候,每个线程都使用自己的局部变量,但是,我们如果在线程内部调用某个函数的时候且需要让这个函数处理我们的变量时,就会产生一个问题,必须将该局部变量给传递进该函数....

ThreadLocal设计模式



hua286306956

2013年03月11日 17:59 4008

本文主要整理自 线程安全问题的由来 在传统的Web开发中,我们处理Http请求最常用的方式是通过实现Servlet对象来进行Http请求的响应.Servlet是J2EE的重要标准之一,规定了J...

ThreadLocal浅析



zldeng19840111

2011年08月20日 11:46 3666

1.目的 ThreadLocal目的是保存一些线程级别的全局变量,比如connection,或者事务上下文,避免这些值需要一直通过函数参数的方式一路传递。 2. 常见用法 public cl...

ThreadLocal



Kurry4ever_

2018年01月02日 10:05 86

参考: <http://www.iteye.com/topic/1123824> java.lang.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...

SSM框架——详细整合教程 (Spring+SpringMVC+ MyBatis)

使用SSM (Spring、SpringMVC和Mybatis) 已经有三个多月了,项目在技术上已经没有什么难点了,基于现有的技术就可以实现想要的功能,当然肯定有很多可以改进的地方。之前没有记录SSM整合...



gebitan505

2015年03月19日 11:44 1240668

muduo源码分析：线程特定/私有数据类ThreadLocal

联系我们



请扫描二维码联系客服
✉ webmaster@csdn.net
☎ 400-660-0108
💬 QQ客服 🗣 客服论坛

关于 招聘 广告服务 百度
©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

线程私有数据 1.__thread : gcc内置的线程局部存储 🤖
设施 __thread只能修饰POD类型 POD类型 (plain old data) , 与C兼容的原始数据 , 例如 , 结构和整型等C语言中的...

java 并发编程之 ThreadLocal 🌟 MakeContral 2018年01月11日 16:27 📖 74
ThreadLocal在Spring中发挥着重要的作用 , 在管理request作用域的Bean、事务管理、任务 AOP等模块都出现了它们的身影 , 起着举足轻重的作用。要想了解Spring事务管理的底层...

通过一堂化学课来彻底理解 ThreadLocal
关于 ThreadLocal 相信很多读者都在网上看 🌟 niuzhucdenglu 2018年01月09日 16:46 💬 269
到了这样的介绍 : ThreadLocal 为解决多线程程序的并发问题提供了一种新的思路 ; ThreadLocal的目的是为了解决多线程访问资源时的共享问...

ThreadLocal的用法理解 🌟 u013696062 2018年01月08日 16:46 📖 34
其实很简单 , 就是创建一个对象 , 然后每个线程去访问时 , 访问的是这个对象的副本。即该对象会为每个线程拷贝出一个副本。 其实效果和local variable是一个效果。即在线程内初始化一个本地变量。 Th...

ThreadLocal详解 🌟 u013410771 2018年01月03日 21:23 📖 180
1.这个类是干嘛用的 Implements a thread-local storage, that is, a variable for which each thread has its ...

彻底理解ThreadLocal 🌟 u013521220 2017年06月22日 15:42 📖 253
ThreadLocal是什么 早在JDK 1.2的版本中就提供Java.lang.ThreadLocal , 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...