

目录

第1章 基础

01 开篇词：为什么学习本专栏

02 String、Long 源码解析和面试题

03 Java 常用关键字理解

04 Arrays、Collections、Objects 常用方法源码解析

第2章 集合

05 ArrayList 源码解析和设计思路

06 LinkedList 源码解析

07 List 源码会问哪些面试题

08 HashMap 源码解析

09 TreeMap 和 LinkedHashMap 核心源码解析

10 Map源码会问哪些面试题

11 HashSet、TreeSet 源码解析

12 彰显细节：看集合源码对我们实际工作的帮助和应用

13 差异对比：集合在 Java 7 和 8 有何不同和改进

14 简化工作：Guava Lists Maps 实际工作运用和源码

第3章 并发集合类

15 CopyOnWriteArrayList 源码解析和设计思路

16 ConcurrentHashMap 源码解析和设计思路

17 并发 List、Map源码面试题

18 场景集合：并发 List、Map的应用

# 44 场景实战：ThreadLocal 在上下文传值场景下的实践

更新时间：2019-11-28 09:55:13



“

生活的理想，就是为了理想的生活。

”

——张闻天

## 开篇语

我们在《打动面试官：线程池流程编排中的运用实战》一文中将流程引擎简单地完善了一下，本文在其基础上继续进行改造，建议同学可以先看看 GitHub 上的代码，或者看看之前的文章。

## 1 回顾

流程引擎编排的对象，我们称为组件（就是 SpringBean），之前我们给组件定义了通用的接口，组件实现时就实现这个接口，代码如下：



<div>← 慕课专栏</div> <div>三 面试官系统精讲Java源码及大厂真题 / 44 场景实战：ThreadLocal 在上下文传值场景下的实践</div>	
目录	<div>}</div> <div>}</div>
第1章 基础	
01 开篇词：为什么学习本专栏	如果你想往 ThreadLocal 放数据，调用 ContextCache.putAttribute 方法，如果想从 ThreadLocal 拿数据，调用 ContextCache.getAttribute 方法即可。
02 String、Long 源码解析和面试题	
03 Java 常用关键字理解	我们写了两个组件，一个组件放数据，一个组件拿数据，如下：
04 Arrays、Collections、Objects 常用方法源码解析	
第2章 集合	
05 ArrayList 源码解析和设计思路	
06 LinkedList 源码解析	
07 List 源码会问哪些面试题	
08 HashMap 源码解析	
09 TreeMap 和 LinkedHashMap 核心源码解析	我们把两个 SpringBean 注册到流程注册中心中，让其按照先执行 BeanThree 再执行 BeanFive 的顺序进行执行，运行 DemoApplication 类的 main 方法进行执行，执行结果如下：
10 Map源码会问哪些面试题	<pre>ServletContainer : Tomcat started on port(s): 8080 (http)                   : Started DemoApplication in 6.705 seconds (JVM running for 8.433) ..BeanThree      : put key1,value1 to contextCache success ..BeanFive       : get key1,value is value1</pre>
11 HashSet、TreeSet 源码解析	从打印的日志可以看到，在 Spring 容器管理的 SpringBean 中，ThreadLocal 也是可以储存中间缓存值的。
12 彰显细节：看集合源码对我们实际工作的帮助和应用	3 开启子线程
13 差异对比：集合在 Java 7 和 8 有何不同和改进	我们做一个实验，我们在 BeanFive 中开启子线程，然后再从 ThreadLocal 中拿值，看看能否拿到值，BeanFive 的代码修改成如下：
14 简化工作：Guava Lists Maps 实际工作运用和源码	<pre>@Slf4j @Component public class BeanFive implements DomainAbilityBean {      @Override     public void invoke() {         new Thread(() -&gt; {             log.info("子线程开启成功，子线程名为:{}", Thread.currentThread().getName());             String value = ContextCache.getAttribute( sourceKey: "key1");             log.info("在子线程中 get key1,value is {}", value);         }).start();     } }</pre> <div>修改成在子线程中，从 ThreadLocal 中获取值</div>
第3章 并发集合类	
15 CopyOnWriteArrayList 源码解析和设计思路	
16 ConcurrentHashMap 源码解析和设计思路	
17 并发 List、Map源码面试题	我们再来运行一下，打印的日志如下：
18 场景集合：并发 List、Map的应用	

<div>← 慕课专栏</div> <div>面试官系统精讲Java源码及大厂真题 / 44 场景实战：ThreadLocal 在上下文传值场景下的实践</div>	
目录	
第1章 基础	<div><div>BeanThree.java : registering beans for scan exposure on startup</div><div>atEmbeddedServletContainer : Tomcat started on port(s): 8080 (http)</div><div>cation : Started DemoApplication in 6.476 seconds (JVM running for 7.313)</div><div>rw.threadlocal.BeanThree : put key1,value1 to contextCache success</div><div>rw.threadlocal.BeanFive : 子线程开启成功, 子线程名为:Thread-4</div><div>rw.threadlocal.BeanFive : 在子线程中 get key1 value is null</div></div>
01 开篇词：为什么学习本专栏	<p>从打印的日志中，我们发现子线程中从 ThreadLocal 取值时，并没有取得值，这个原因主要是我们之前说的，线程在创建的时候，并不会把父线程的 ThreadLocal 中的值拷贝给子线程的 ThreadLocal，解决方案就是把 ThreadLocal 修改成 InheritableThreadLocal，代码修改如下：</p>
02 String、Long 源码解析和面试题	<div><pre>public class ContextCache implements Serializable {     private static final long serialVersionUID = 2136539028591849277L;     // 使用 ThreadLocal 缓存上下文信息     public static final ThreadLocal&lt;Map&lt;String,String&gt;&gt; CACHE = new InheritableThreadLocal&lt;&gt;(); }</pre><div>从 ThreadLocal 修改成这个类</div></div>
03 Java 常用关键字理解	<p>我们再次运行，结果如下：</p>
04 Arrays、Collections、Objects 常用方法源码解析	<div><div>ree : Started DemoApplication in 8.058 seconds (JVM running for 9.46)</div><div>hree : put key1,value1 to contextCache success</div><div>ive : 子线程开启成功, 子线程名为:Thread-4</div><div>ive : 在子线程中 get key1,value is value1</div></div>
第2章 集合	<p>从运行结果看，我们成功的在子线程中拿到值。</p>
05 ArrayList 源码解析和设计思路	<h3>4 线程池 + ThreadLocal</h3>
06 LinkedList 源码解析	<p>如果是拿数据的 springBean 是丢给线程池执行的，我们能够成功的从 ThreadLocal 中拿到数据么？</p>
07 List 源码会问哪些面试题	<p>首先我们在放数据的 springBean 中，把放的值修改成随机的，接着拿数据的 SpringBean 修改成异步执行，代码修改如下：</p>
08 HashMap 源码解析	<div><div>BeanThree.java</div><div>11 @Component</div><div>12 public class BeanThree implements DomainAbilityBean {</div><div>13</div><div>14 @Override</div><div>15 public void invoke() {</div><div>16 int i = new Random().nextInt(1000);</div><div>17 ContextCache.putAttribute( sourceKey: "key1", value: "value"+i);</div><div>18 log.info("put key1,value{}",i); 往 ThreadLocal 中放数据</div><div>BeanFive.java</div><div>9 @Slf4j</div><div>10 @AsyncComponent 加上异步执行的注解, 任务会被丢到线程池中去执行</div><div>11 @Component</div><div>12 public class BeanFive implements DomainAbilityBean {</div><div>13</div><div>14 @Override</div><div>15 public void invoke() {</div><div>16 String value = ContextCache.getAttribute( sourceKey: "key1");</div><div>17 try {</div><div>18 Thread.sleep( millis: 5000); 沉睡一段时间, 让线程消费更慢, 可以让大多数任务可以在阻塞队列中去排队</div><div>19 } catch (InterruptedException e) {</div><div>20 e.printStackTrace();</div><div>21 }</div><div>22 log.info("get 线程名称为{}, value is {}", Thread.currentThread().getName(),value);</div><div>23 log.info("-----");</div><div>24 }</div></div>

<div>← 慕课专栏</div>	<div>面试官系统精讲Java源码及大厂真题 / 44 场景实战：ThreadLocal 在上下文传值场景下的实践</div>
<div>目录</div>	<div><pre>public static void main(String[] args) throws InterruptedException {     SpringApplication.run(DemoApplication.class);     for (int i = 0; i &lt; 10; i++) {         ApplicationContextHolder.getBean(FlowStart.class)             .start( flowName: "flow2", new FlowContent());     } }</pre></div>
<div>第1章 基础</div>	
<div>01 开篇词：为什么学习本专栏</div>	
<div>02 String、Long 源码解析和面试题</div>	
<div>03 Java 常用关键字理解</div>	<div>我们期望的结果：</div> <div><div></div><div><div>1. 线程池中执行的 BeanFive 可以成功从 ThreadLocal 中拿到数据；</div><div>2. 能够从 ThreadLocal 拿到正确的数据，比如 BeanThree 刚放进 key1，value5，那么期望在 BeanFive 中根据 key1 能拿出 value5，而不是其它值。</div></div></div>
<div>04 Arrays、Collections、Objects 常用方法源码解析</div>	
<div>第2章 集合</div>	<div>我们运行一下，结果如下：</div> <div><div></div></div>
<div>05 ArrayList 源码解析和设计思路</div>	
<div>06 LinkedList 源码解析</div>	
<div>07 List 源码会问哪些面试题</div>	
<div>08 HashMap 源码解析</div>	
<div>09 TreeMap 和 LinkedHashMap 核心源码解析</div>	<div>从结果中可以看到，并没有符合我们的预期，我们往 ThreadLocal 中 put 进很多值，但最后拿出来的值却很多都是 value379，都为最后 put 到 ThreadLocal 中的值。</div>
<div>10 Map源码会问哪些面试题</div>	
<div>11 HashSet、TreeSet 源码解析</div>	<div>这个原因主要是 ThreadLocal 存储的 HashMap 的引用都是同一个，main 主线程可以修改 HashMap 中的值，子线程从 ThreadLocal 中拿值时，也是从 HashMap 中拿值，从而导致不能把 put 的值通过 ThreadLocal 正确的传递给子线程。</div>
<div>12 彰显细节：看集合源码对我们实际工作的帮助和应用</div>	<div>为了证明是这个原因，我们在从 ThreadLocal 放、拿值的地方，把 HashMap 的内存地址都打印出来，改动代码如下：</div> <div><div></div></div>
<div>13 差异对比：集合在 Java 7 和 8 有何不同和改进</div>	
<div>14 简化工作：Guava Lists Maps 实际工作运用和源码</div>	
<div>第3章 并发集合类</div>	
<div>15 CopyOnWriteArrayList 源码解析和设计思路</div>	
<div>16 ConcurrentHashMap 源码解析和设计思路</div>	
<div>17 并发 List、Map源码面试题</div>	<div>我们再次运行测试代码，运行的结果如下：</div>
<div>18 场景集合：并发 List、Map的应用</div>	

目录	
第1章 基础	
01 开篇词：为什么学习本专栏	
02 String、Long 源码解析和面试题	
03 Java 常用关键字理解	
04 Arrays、Collections、Objects 常用方法源码解析	
第2章 集合	
05 ArrayList 源码解析和设计思路	
06 LinkedList 源码解析	
07 List 源码会问哪些面试题	
08 HashMap 源码解析	
09 TreeMap 和 LinkedHashMap 核心源码解析	
10 Map源码会问哪些面试题	
11 HashSet、TreeSet 源码解析	
12 彰显细节：看集合源码对我们实际工作的帮助和应用	
13 差异对比：集合在 Java 7 和 8 有何不同和改进	
14 简化工作：Guava Lists Maps 实际工作运用和源码	
第3章 并发集合类	
15 CopyOnWriteArrayList 源码解析和设计思路	
16 ConcurrentHashMap 源码解析和设计思路	
17 并发 List、Map源码面试题	
18 场景集合：并发 List、Map的应用	



从测试结果中可以看到，不管是主线程还是子线程和 ThreadLocal 进行交互时，HashMap 都是同一个，也就是说 ThreadLocal 中保存的 HashMap 是共享的，这就导致了线程安全的问题，子线程读取到的值就会混乱掉。

## 5 解决方案

针对这个问题，我们提出了一种解决方案，在把任务提交到线程池时，我们进行 HashMap 的拷贝，这样子线程的 HashMap 和 main 线程的 HashMap 就不同了，可以解决上面的问题。

我们提交任务时，使用的是 Runnable，要实现 HashMap 的拷贝的话，我们需要把 Runnable 进行一层包装，包装的代码如下：



运行结果如下：



从运行结果中可以看出，线程池拿出来的 value 已经是正确的了。

## 6 总结

本文通过 ThreadLocal 来改造流程引擎中的上下文传递，希望能够加深大家对 ThreadLocal 的认识和使用技巧，有兴趣的同学可以把我们的代码下载下来，跑跑看。

目录	欢迎在这里发表留言，作者筛选后可公开显示		
第1章 基础			
01 开篇词：为什么学习本专栏			
02 String、Long 源码解析和面试题			
03 Java 常用关键字理解			
04 Arrays、Collections、Objects 常用方法源码解析			
第2章 集合			
05 ArrayList 源码解析和设计思路			
06 LinkedList 源码解析			
07 List 源码会问哪些面试题			
08 HashMap 源码解析			
09 TreeMap 和 LinkedHashMap 核心源码解析			
10 Map源码会问哪些面试题			
11 HashSet、TreeSet 源码解析			
12 彰显细节：看集合源码对我们实际工作的帮助和应用			
13 差异对比：集合在 Java 7 和 8 有何不同和改进			
14 简化工作：Guava Lists Maps 实际工作运用和源码			
第3章 并发集合类			
15 CopyOnWriteArrayList 源码解析和设计思路			
16 ConcurrentHashMap 源码解析和设计思路			
17 并发 List、Map源码面试题			
18 场景集合：并发 List、Map的应用			