Spring中使用自定义ThreadLocal存储导致的坑及解决

作者：有故事的人 来源：范文网 www.wtabcd.cn/fanwen/

本文原地址：https://www.wtabcd.cn/fanwen/zuowen/59e9f9a35156a652d96855f796078fa5.html

目录

spring自定义threadlocal存储导致的坑

spring 中有时候我们需要存储一些和 request
相关联的变量，例如用户的登陆有关信息等，它的生命周期和 request 相同。

一个容易想到的实现办法是使用threadlocal

```
public class securitycontextholder {  private static final threadlocal<securitycontext> securitycontext =
new threadlocal<securitycontext>();  public static void set(securitycontext context) {  sec
uritycontext.set(context);  }  public static securitycontext get() {  return securityco
ntext.get();  }  public static void clear() {  securitycontext.remove();  }}
```

使用一个自定义的handlerinterceptor将有关信息注入进去

```
@slf4j@componentpublic class requestinterceptor implements handlerinterceptor {  @override  publi
c boolean prehandle(httpservletrequest request, httpservletresponse response, object handler) throws
  exception {  try {  securitycontextholder.set(retrieverequest
context(request));  } catch (exception ex) {  log.warn("读取请求信息失
败", ex);  }  return true;  }  @override  public void posth
andle(httpservletrequest request, httpservletrespo论文的格式范文nse response, object handler, @nullable
  modelandview modelandview) throws exception {  securitycontextholder.clear();
}
```

通过这样，我们就可以在 controller 中直接使用这个 context，很方便的获取到有关用户的信息

```
@slf4j@restcontrollerclass controller {  public result get() {  long userid = securitycontextholder.ge
t().getuserid();  // ...  }}
```

这个方法也是很多博客中使用的。然而这个方法却存在着一个很隐蔽的坑： handlerinterceptor 的 posthandle
并不总是会调用。

当controller中出现exception

```
@slf4j@restcontrollerclass controller {  public result get() {  long userid = securitycontextholder.ge
```

t().getuserid(); // ... throw new runtimeexception(); }}

或者在handlerinterceptor的prehandle中出现exception

@slf4j@componentpublic class requestinterceptor implements handlerinterce核心价值观的作用ptor { @ove
rride public boolean prehandle(httpservletrequest request, httpservletresponse response, object handler)
throws exception { try { securitycontexthol
der.set(retrieverequestcontext(request)); } catch (exception ex) { log.wa
rn("读取请求信息失败", ex); } // ... 东方绿舟好玩吗 throw n
ew runtimeexception(); //... return true; }}

这些情况下， posthandle 并不会调用。这就导致了 threadlocal 变量不能被清理。

在平常的 java 环境中，threadlocal 变量随着 thread 本身的销毁，是可以被销毁掉的。但 spring 由于采用了线程池的设计，响应请求的线程可能会一直常驻，这就导致了变量一直不能被 gc 回收。更糟糕的是，这个没有被正确回收的变量，由于线程池对线程的复用，可能会串到别的 request 当中，进而直接导得用致代码逻辑的错误。

为了解决这个问题，我们可以使用 spring 自带的 requestcontextholder，它背后的原理也是 threadlocal，不过它总会被更底层的 servlet 的 filter 清理掉，因此不存在泄露的问题。

下面是一个使用requestcontextholder重写的例子

public class securitycontextholder { private static final string security_context_attributes = "security_co
ntext"; public static void setcontext(securitycontext context) { requestcontextholder.curre
ntrequestattributes().setattribute(security_context_attributes,
context, requestattributes.scope_request); } public st
atic securitycontext get() { return (securitycontext)requestcontextholder.currentrequestattributes()
.getattribute(security_context_attributes, requestattributes.scope_request); }}

除了使用 requestcontextholder 还可以使用 request scope 的 bean，或者使用 threadlocaltargetsource，原理上是类似的。

需要时刻注意 threadlocal 相当于线程内部的 static 变量，是一个非常容易产生泄露的点，因此使用 threadlocal 应该额外小心。

threadlocal可能会产生内存泄露的问题及原理

刚遇到一个关于threadlocal的内存泄漏问题，刚好总结一下

比较常用的这里先不提，直接提比较重要的部分

为什么会产生内存泄露？

public void set(t value) { thread t = thread.currentthread(); threa
dlocalmap map = getmap(t); if (map != null) map.set(this, value)
; 学写倡议书 else createmap(t, value); }

set方法里面，先调用到当前线程thread，每个线程里都会有一个threadlocals成员变量，指向对应的threadlocalmap，然后以new出来的引用作为key，和给定的value一块保存起来。

当外部引用解除以后，对应的threadlocal对象由于被内部threadlocalmap 引用，不会gc，可能会导致内存泄露。

jvm解决的办法

```
static class entry extends weakreference<threadlocal<?>> {          /** the
 value associated with this threadlocal. */          object value;          e
ntry(threadlocal<?> k, object v) {              super(k);
  value = v;          }      }
```

继承了一个软引用，在系统进行gc的时候就可以回收

但是回收以后，key变成null，value也无法被访问到，还是可能存在内存泄露。 因此一旦不用了，必须对里面的keyvalue对remove掉，否则就会有内存泄露；而且在threadlocal源码里面，在每次get或者set的时候会清楚里面key为value的记录

以上为个人经验，希望能给大家一个参考，也希望大家多多支持www.887551.com。