《Java并发性和多线程介绍》-Java TheadLocal

原文链接 作者：Jakob Jenkov 查看全部文章

Java中的ThreadLocal类可以让你创建的变量只被同一个线程进行读和写操作。因此，尽管有两个线程同时执行一段相同的代码，而且这段代码又有一个指向同一个ThreadLocal变量的引用，但是这两个线程依然不能看到彼此的ThreadLocal变量域。

1、创建一个ThreadLocal对象

如下所示，创建一个ThreadLocal变量：

```
1 private ThreadLocal myThreadLocal = new ThreadLocal();
```

你实例化了一个ThreadLocal对象。每个线程仅需要实例化一次即可。虽然不同的线程执行同一段代码时，访问同一个ThreadLocal变量，但是每个线程只能看到私有的ThreadLocal实例。所以不同的线程在给ThreadLocal对象设置不同的值时，他们也不能看到彼此的修改。

2、访问ThreadLocal对象

一旦创建了一个ThreadLocal对象，你就可以通过以下方式来存储此对象的值：

```
1 myThreadLocal.set("A thread local value");
```

也可以直接读取一个ThreadLocal对象的值：

```
1 String threadLocalValue = (String) myThreadLocal.get();
```

get()方法会返回一个Object对象，而set()方法则依赖一个Object对象参数。

3、ThreadLocal泛型

为了使get()方法返回值不用做强制类型转换，通常可以创建一个泛型化的ThreadLocal对象。以下就是一个泛型化的ThreadLocal示例：

```
1 private ThreadLocal myThreadLocal1 = new ThreadLocal<String>();
```

现在你可以存储一个字符串到ThreadLocal实例里，此外，当你从此ThreadLocal实例中获取值的时候，就不必要做强制类型转换。

```
1 myThreadLocal1.set("Hello ThreadLocal");
2 String threadLocalValues = myThreadLocal.get();
```

4、初始化ThreadLocal

由于ThreadLocal对象的set()方法设置的值只对当前线程可见，那有什么方法可以为ThreadLocal对象设置的值对所有线程都可见。

为此，我们可以通过ThreadLocal子类的实现，并覆写initialValue()方法，就可以为ThreadLocal对象指定一个初始化值。如下所示:

```
1 private ThreadLocal myThreadLocal = new ThreadLocal<String>() {
2   @Override protected String initialValue() {
3       return "This is the initial value";
4   }
5 };
```

此时，在set()方法调用前，当调用get()方法的时候，所有线程都可以看到同一个初始化值。

5、Full ThreadLocal Example

以下是一个完整的ThreadLocal示例：

```
01 public class ThreadLocalExample {
02
03     public static class MyRunnable implements Runnable {
04
05         private ThreadLocal<Integer> threadLocal =
06                 new ThreadLocal<Integer>();
07
08         @Override
09         public void run() {
10             threadLocal.set( (int) (Math.random() * 100D) );
11
12             try {
13                 Thread.sleep(2000);
14             } catch (InterruptedException e) {
15             }
16
17             System.out.println(threadLocal.get());
```

```
        }
    }

    public static void main(String[] args) {
        MyRunnable sharedRunnableInstance = new MyRunnable();

        Thread thread1 = new Thread(sharedRunnableInstance);
        Thread thread2 = new Thread(sharedRunnableInstance);

        thread1.start();
        thread2.start();

        thread1.join(); //wait for thread 1 to terminate
        thread2.join(); //wait for thread 2 to terminate
    }

}
```

上面创建了两个线程共享一个MyRunnable实例。每个线程执行run()方法的时候，会给同一个ThreadLocal实例设置不同的值。如果调用set()方法的时候用synchronized关键字同步，而且不是一个ThreadLocal对象实例，那么第二个线程将会覆盖第一个线程所设置的值。

然而，由于是ThreadLocal对象，所以两个线程无法看到彼此的值。因此，可以设置或者获取不同的值。

6、InheritableThreadLocal

InheritableThreadLocal类是ThreadLocal的子类。为了解决ThreadLocal实例内部每个线程都只能看到自己的私有值，所以InheritableThreadLocal允许一个线程创建的所有子线程访问其父线程的值。

闪小达
2015/10/30 4:07下午

支持楼

whsshuai
2016/03/25 4:21下午

“如果调用set()方法的时候用synchronized关键字同步，而且不是一个ThreadLocal对象实例”

这里的“一个”把我整蒙b了，看了原链接才明白指的是“不是ThreadLocal对象的实例”

是我语文水平太低了么

elong
2016/09/20 11:27上午

我也被“一个”整懵逼了！！！

cenyol
2017/03/16 9:54上午

看了下源码，ThreadLocal类的这个原理是：

每个线程自身都有一个ThreadLocalMap类型的成员变量，用来维护它自己所属的ThreadLocal数据。其中ThreadLocalMap是ThreadLocal的内部static类。

ThreadLocal中的set和get方法在操作之前都会先获取当前的线程t，然后在取得t的成员变量ThreadLocalMap，之后再对这个map进行操作。

通过将数据存放在每个线程的私有成员中，以此来实现ThreadLocal的效果