Java ThreadLocal的使用

Java中的ThreadLocal类允许我们创建只能被同一个线程读写的变量。因此，如果一段代码含有一个ThreadLocal变量的引用，即使两个线程同时执行这段代码，它们也无法访问到对方的ThreadLocal变量。

如何创建ThreadLocal变量

以下代码展示了如何创建一个ThreadLocal变量：

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

我们可以看到，通过这段代码实例化了一个ThreadLocal对象。我们只需要实例化对象一次，并且也不需要知道它是被哪个线程实例化。虽然所有的线程都能访问到这个ThreadLocal实例，但是每个线程却只能访问到自己通过调用ThreadLocal的set()方法设置的值。即使是两个不同的线程在同一个ThreadLocal对象上设置了不同的值，他们仍然无法访问到对方的值。

如何访问ThreadLocal变量

一旦创建了一个ThreadLocal变量，你可以通过如下代码设置某个需要保存的值：

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

可以通过下面方法读取保存在ThreadLocal变量中的值：

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

get()方法返回一个Object对象，set()对象需要传入一个Object类型的参数。

为ThreadLocal指定泛型类型

我们可以创建一个指定泛型类型的ThreadLocal对象，这样我们就不需要每次对使用get()方法返回的值作强制类型转换了。下面展示了指定泛型类型的ThreadLocal例子：

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

现在我们只能往ThreadLocal对象中存入String类型的值了。

并且我们从ThreadLocal中获取值的时候也不需要强制类型转换了。

如何初始化ThreadLocal变量的值

由于在ThreadLocal对象中设置的值只能被设置这个值的线程访问到，线程无法在ThreadLocal对象上使用set()方法保存一个初始值，并且这个初始值能被所有线程访问到。

但是我们可以通过创建一个ThreadLocal的子类并且重写initialValue()方法，来为一个ThreadLocal对象指定一个初始值。就像下面代码展示的那样：

```
1 private ThreadLocal myThreadLocal = new ThreadLocal<String>() {
2
3     @Override
4     protected String initialValue() {
5         return "This is the initial value";
6     }
7
8 };
```

一个完整的ThreadLocal例子

下面是一个完整的可执行的ThreadLocal例子：

```
01 public class ThreadLocalExample {
02
03     public static class MyRunnable implements Runnable {
04
05         private ThreadLocal threadLocal = new ThreadLocal();
06
07         @Override
08         public void run() {
09             threadLocal.set((int) (Math.random() * 100D));
10             try {
11             Thread.sleep(2000);
12             } catch (InterruptedException e) {
13
14             }
15             System.out.println(threadLocal.get());
16         }
17     }
18
19     public static void main(String[] args) {
20          MyRunnable sharedRunnableInstance = new MyRunnable();
```

```
        Thread thread1 = new Thread(sharedRunnableInstance);
        Thread thread2 = new Thread(sharedRunnableInstance);
        thread1.start();
        thread2.start();
    }

}
```

上面的例子创建了一个MyRunnable实例，并将该实例作为参数传递给两个线程。两个线程分别执行run()方法，并且都在ThreadLocal实例上保存了不同的值。如果它们访问的不是ThreadLocal对象并且调用的set()方法被同步了，则第二个线程会覆盖掉第一个线程设置的值。但是，由于它们访问的是一个ThreadLocal对象，因此这两个线程都无法看到对方保存的值。也就是说，它们存取的是两个不同的值。

关于InheritableThreadLocal

InheritableThreadLocal类是ThreadLocal类的子类。ThreadLocal中每个线程拥有它自己的值，与ThreadLocal不同的是，InheritableThreadLocal允许一个线程以及该线程创建的所有子线程都可以访问它保存的值。

【注：所有子线程都会继承父线程保存的ThreadLocal值】