线程封闭

线程封闭的概念

1. 多线程访问共享可变数据时，涉及到线程间数据同步的问题。并不是所有时候，都要用到共享数据，所以线程封闭概念就提出来了。
2. 数据都被封闭在各自的线程中，就不需要同步，这种通过将数据封闭在线程中而避免使用到同步的技术称为线程封闭。
3. 线程封闭的具体实现：ThreadLocal，局部变量

ThreadLocal

1. ThreadLocal是Java里的一种特殊变量。它是一个线程级别变量，每个线程都有一个ThreadLocal就是每个线程都拥有了自己独立的一个变量，竞争条件被彻底消除了，在并发模式下绝对安全。
2. ThreadLocal value = new ThreadLocal<>();会自动在每一个线程上创建一个T的副本，副本之间彼此独立，互不影响。可以用ThreadLocal存储一些参数，以便在线程中多个方法中使用，用来代替方法传参的做法。

```
public class Demo7 {
    /** threadLocal变量，每个线程都有一个副本，互不干扰 */
    public static ThreadLocal<String> value = new ThreadLocal<>();

    /**
     * threadlocal测试
     *
     * @throws Exception
     */
    public void threadLocalTest() throws Exception {

        // threadlocal线程封闭示例
        value.set("这是主线程设置的123"); // 主线程设置值
        String v = value.get();
        System.out.println("线程1执行之前，主线程取到的值：" + v);

        new Thread(new Runnable() {
            @Override
            public void run() {
                String v = value.get();
                System.out.println("线程1取到的值：" + v);
                // 设置 threadLocal
                value.set("这是线程1设置的456");

                v = value.get();
                System.out.println("重新设置之后，线程1取到的值：" + v);
                System.out.println("线程1执行结束");
            }
        }).start();

        Thread.sleep(5000L); // 等待所有线程执行结束
```

```
        v = value.get();
        System.out.println("线程1执行之后，主线程取到的值：" + v);

    }

    public static void main(String[] args) throws Exception {
        new Demo7().threadLocalTest();
    }
}
```

局部变量

1. 局部变量的固有属性之一就是封闭在线程中。它们位于执行线程的栈中，其他线程无法访问这个栈。