TheadLocal

C-2 创建: 张林伟, 最后修改: 张林伟 2018-12-21 17:05

ThreadLocal原理

每个线程 Thread 类有个实例变量 threadLocals，用于存放当前线程的变量副本，threadLocals是 ThreadLocal.ThreadLocalMap 类型，其中 key是 ThreadLocal对象，value 是变量副本。

set 方法

```
public void set(T value) {
    Thread t = Thread.currentThread();
    ThreadLocalMap map = getMap(t);
    if (map != null)
        map.set(this, value);
    else
        createMap(t, value);
}
```

```
ThreadLocalMap getMap(Thread t) {
    return t.threadLocals;
}
```

```
void createMap(Thread t, T firstValue) {
    t.threadLocals = new ThreadLocalMap( firstKey: this, firstValue);
}
```

根据当前线程调用 getMap 方法获取 ThreadLocalMap 对象（即当前线程的 threadLocals 变量），如果不为 null，则以 ThreadLocal 对象为key，需要存储的值为 value 进行存储。如果为 null，则调用 createMap 方法初始化 threadLocals变量并设置key-value。

get 方法

```
public T get() {
    Thread t = Thread.currentThread();
    ThreadLocalMap map = getMap(t);
    if (map != null) {
        ThreadLocalMap.Entry e = map.getEntry( key: this);
        if (e != null) {
            /unchecked/
            T result = (T)e.value;
            return result;
        }
    }
    return setInitialValue();
}
```

```
private T setInitialValue() {
    T value = initialValue();
    Thread t = Thread.currentThread();
    ThreadLocalMap map = getMap(t);
    if (map != null)
        map.set(this, value);
    else
        createMap(t, value);
    return value;
}
```

```
protected T initialValue() {
    return null;
}
```

根据当前线程调用 getMap 方法获取 ThreadLocalMap 对象（即当前线程的 threadLocals 变量），如果不为 null，则返回 ThreadLocalMap 中 Entry 的值，如果为 null，调用 setInitialValue 方法初始化，初始值为 null。

SuppliedThreadLocal 内部类

```
static final class SuppliedThreadLocal<T> extends ThreadLocal<T> {

    private final Supplier<? extends T> supplier;

    SuppliedThreadLocal(Supplier<? extends T> supplier) {
        this.supplier = Objects.requireNonNull(supplier);
    }

    @Override
    protected T initialValue() { return supplier.get(); }
}
```

```
public static <S> ThreadLocal<S> withInitial(Supplier<? extends S> supplier) {
    return new SuppliedThreadLocal<>(supplier);
}
```

提供初始值。使用 withInitial 静态方法创建有初始值的 ThreadLocal 对象。

ThreadLocalMap.Entry 内部类

```
static class Entry extends WeakReference<ThreadLocal<?>> {
    /** The value associated with this ThreadLocal. */
    Object value;

    Entry(ThreadLocal<?> k, Object v) {
        super(k);
        value = v;
    }
}
```

ThreadLocalMap 中的 Entry 是对 ThreadLocal 弱引用，这样当没有强引用指向 ThreadLocal 对象时，它可以被回收。

待确认：

> 但是，这里又可能出现另外一种内存泄漏的问题。ThreadLocalMap 维护 ThreadLocal 变量与具体实例的映射，当 ThreadLocal 变量被回收后，该映射的键变为 null，该 Entry 无法被移除。从而使得实例被该 Entry 引用而无法被回收造成内存泄漏。
>
> *注：* Entry虽然是弱引用，但它是 ThreadLocal 类型的弱引用（也即上文所述它是对 *键* 的弱引用），而非具体实例的的弱引用，所以无法避免具体实例相关的内存泄漏。
>
> http://www.jasongj.com/java/threadlocal/