ThreadLocal

> **参考视频或文章**：
>
> - https://blog.csdn.net/u010445301/article/details/111322569?fromshare=blogdetail&sharetype=blogdetail&sharerId=111322569&sharerefer=PC&sharesource=2401_83600210&sharefrom=from_link

一、技术介绍

1.简介

- `ThreadLocal<T>`是当前线程独有的变量：在ThreadLocal中填充的变量属于当前线程，对于其他线程而言是隔离的。
- ThreadLocal为每个线程提供了变量的实例副本，每个使用该变量的线程都会初始化一个完全独立的实例副本。
- ThreadLocal变量通常被`private static`修饰，当一个线程结束时，它所使用的所有ThreadLocal变量的实例副本都可以被回收。

2.ThreadLocal与Synchronized的区别

区别	ThreadLocal	Synchronized
用途	用于线程间的数据隔离	用于线程间的数据共享
原理	为每个线程都提供了变量的副本，使得每个线程在某一时间访问到的并非同一对象而是该对象的不同副本	利用锁机制，使得变量或代码块在某一时刻只能被一个线程访问

二、项目应用

1.需求

- 管理端每次登入的用户不同，我们需要根据用户id生成对应的jwt令牌，然后保存这个用户id（即保存到ThreadLocal中），因为后续填写`create_user`和`update_user`等场景需要使用到当前登入的用户id。
- 用户端每次登入的用户也不同，我们也需要根据用户id生成对应的jwt令牌，然后保存这个用户id，因为后续订单相关接口等场景需要使用到当前登入的用户id。

2.使用ThreadLocal实现需求

```java
/**
 * 封装了ThreadLocal的工具类
 */
public class BaseContext {

    public static ThreadLocal<Long> threadLocal = new ThreadLocal<>();

    public static void setCurrentId(Long id) {
        threadLocal.set(id);
    }

    public static Long getCurrentId() {
        return threadLocal.get();
    }

    public static void removeCurrentId() {
        threadLocal.remove();
    }

}
```