使用ThreadLocal管理Session（重点）,spring默认也是使用ThreadLocal去管理session

疑问：为什么要使用ThreadLocal来管理Session对象？
在业务层是无法使用事务进行管理方式！！！

ThreadLocal底层可以当做一个map集合，<线程ID，对象>

1，配置hibernate的ThreadLocal

①修改hibernate.cfg.xml配置文件：

```
<!-- 让session被ThreadLocal管理 -->
		<property name="hibernate.current_session_context_class">
		thread
		</property>
```

②修改hibernateUtile工具类

```
private static Configuration cfgConfiguration = null;
	private static SessionFactory factory = null;
	//只需要执行一次
	static {
		cfgConfiguration = new Configuration();
		cfgConfiguration.configure();
		factory = cfgConfiguration.buildSessionFactory();
	}
	/**
	 * 让外部提取session对象
	 */
	public static Session getSession(){
		//从本地线程拿回session对象
		return factory.getCurrentSession();
	}
```

③对相应的hibernate代码进行一个修改

比如在代码中不需要Transaction ts = session.beginTransaction();以及ts.commit();类似的代码。并且最后不能去关闭session。

即在dao层，不需要开启事务，也不需要关闭session对象。

在service层，需要重新拿到一个session对象，Session session = hibernateUtils.getSession();

Transaction ts = session.beginTransaction();将需要执行的代码放在一个try{}catch（）{}块中，当所有的操作都执行完毕之后，执行ts.commit（），如果发生异常，则在catch块中执行ts.rollback（）方法。

事务的四个特征：

原子性：事务要么一起成功，要么一起失败。

一致性：事务操作应该保证数据库的数据操作前后是一致的。

隔离性：并发的事务需要相互隔离。

持久性：事务一旦提交，数据应该永久保存。

并发事务会存在的问题：

1）脏读：一个事务读到另一个并发事务的未提交的数据。

2）不可重复读：一个事务读到了另一个并发事务的update数据。

3）幻读：一个事务读到了另一个并发事务的insert数据。

数据库层面可以设置相应的隔离级别：

数据库的隔离级别防止以上山中现象：

①read uncommited：不能防止脏读，不可重复读以及幻读。

②read commited：防止脏读，但是不能防止不可重复读和幻读。（Oracle的默认隔离级别）

③repeatable read：防止脏读，不可重复读，但是不能防止幻读。（mysql的默认隔离级别）

④serializable：防止脏读，不可重复读以及幻读。

hibernate通过配置修复数据库的隔离级别

```
<!-- 修改数据库的隔离级别 -->
<property name="hibernate.connection.isolation">
1/2/4/8
</property>
```

其中1表示read uncommited；2表示read commited；4表示repeatable read；8表示serializable。