假设有如下的类 User UserGroup

然后在UserGroup显然会有一堆的User

一般可以用Set<User> 或 List<User> 来保存

也可以用Map来保存,但问题是它的Key是什么呢?

在UserGroup里

@OneToMany

@MapKey(name="name")

**public** Map<String,User> **getUsers**() {

**return** users;

}

这样就会以User的name为key建立一个map

当然你的key肯定要具有唯一性

否则不会报错,而是会覆盖如果name="abc"的user有2个,那么只会出现其中一个

不会报错也不会自动生成唯一约束

session的refresh方法可以可以更新一个对象的状态

s.beginTransaction();

Person p=(Person)s.get( Person.**class**, 3 );

System.out.println(p.getName());

s.getTransaction().commit();

Thread.sleep( 3000 );

s.beginTransaction();

s.refresh(p);

System.out.println(p.getName());

s.getTransaction().commit();

在睡觉的3秒期间去更改数据库的name,则两次输出结果是不一样的

但是要在两个不同的事务内,否则输出还是一样的,

因为隔离级别的关系?

Session s = sessionFactory.openSession();

s.beginTransaction();

Person p=(Person)s.get( Person.**class**, 3 );

p.setName( "333" );

s.getTransaction().commit();

s.close();

这样会将name设置为333

而不用显示调用update

可以用s.evict( p );将p驱逐出去

从而在事务提交的时候不会将 对p的修改 写入到数据库

session.contains(o)

检测o是不是在session上下文中(被session缓存)

lazy模式的时候可以使用Hibernate这个类的静态方法

强制某个属性初始化或一些其他的设置if ( Hibernate.isInitialized( customer.getAddress() ) {

//display address if loaded

}

if ( Hibernate.isInitialized( customer.getOrders()) ) ) {

//display orders if loaded

}

if (Hibernate.isPropertyInitialized( customer, "detailedBio" ) ) {

//display property detailedBio if loaded

}