# Hibernate\_day03笔记

# 上次课内容回顾

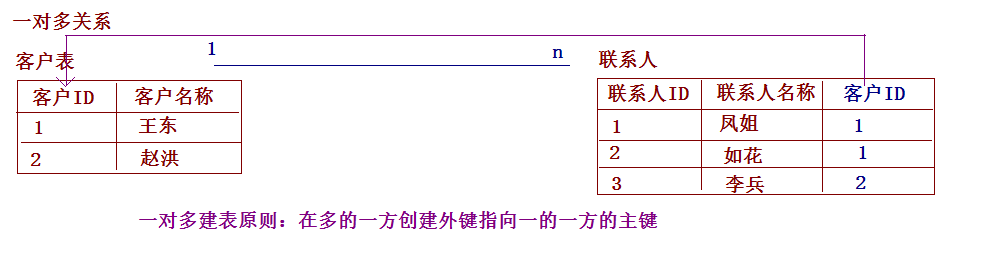
* Hibernate的持久化类的编写规则
  + 无参数构造
  + 属性私有
  + 属性尽量使用包装类
  + 提供一个唯一OID与主键对应
  + 不要使用final修饰
* Hibernate的主键生成策略
  + 主键分类
    - 自然主键
    - 代理主键
  + 主键生成策略
    - increment
    - identity
    - sequence
    - uuid
    - native
    - assigned
    - foreign
* Hibernate的持久化类的三种状态
  + 瞬时态：没有唯一标识OID，没有被session管理
  + 持久态：有唯一标识OID，已经被session管理
  + 脱管态：有唯一标识OID，没有被session管理
  + 状态转换：（了解）
* Hibernate的一级缓存
  + 一级缓存：Hibernate优化手段，称为是session级别缓存。
  + 一级缓存：快照区
* Hibernate的事务管理
  + 事务的回顾
    - 事务的概念
    - 事务的特性
    - 引发安全性问题
    - 安全性问题解决
  + Hibernate解决读问题
    - 配置设置隔离级别
  + Hibernate解决Service事务
    - 采用的是线程绑定的方式：
* Hibernate的其他的API
  + Query ：HQL 面向对象方式的查询。
  + Criteria ：QBC 完成面向对象化。
  + SQLQuery ：SQL查询

## Hibernate的一对多关联映射

### 数据库表与表之间的关系

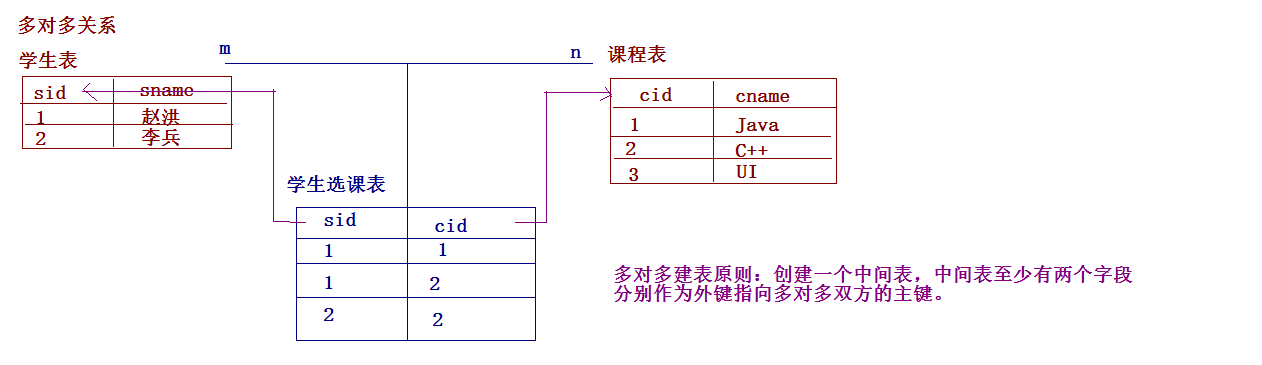
#### 一对多关系

* 什么样关系属于一对多？
  + 一个部门对应多个员工，一个员工只能属于某一个部门。
  + 一个客户对应多个联系人，一个联系人只能属于某一个客户。
* 一对多的建表原则：



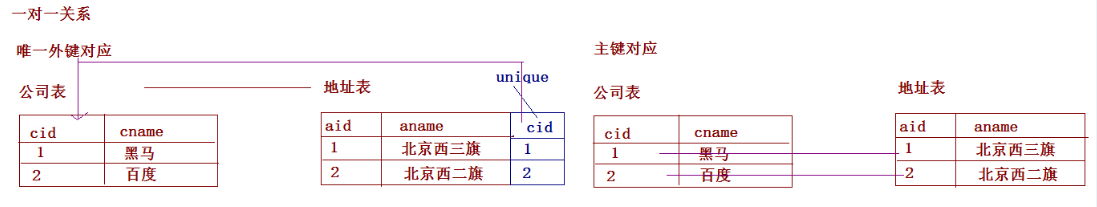
#### 多对多关系

* 什么样关系属于多对多？
  + 一个学生可以选择多门课程，一门课程也可以被多个学生选择。
  + 一个用户可以选择多个角色，一个角色也可以被多个用户选择。
* 多对多的建表原则：



#### 一对一关系（了解）

* 什么样关系属于一对一？
  + 一个公司只能有一个注册地址，一个注册地址只能被一个公司注册。
* 一对一的建表原则：



### Hibernate一对多的关系配置

#### 创建一个项目，引入相应jar包

#### 创建数据库和表

CREATE TABLE `cst\_customer` (

`cust\_id` bigint(32) NOT NULL AUTO\_INCREMENT COMMENT '客户编号(主键)',

`cust\_name` varchar(32) NOT NULL COMMENT '客户名称(公司名称)',

`cust\_source` varchar(32) DEFAULT NULL COMMENT '客户信息来源',

`cust\_industry` varchar(32) DEFAULT NULL COMMENT '客户所属行业',

`cust\_level` varchar(32) DEFAULT NULL COMMENT '客户级别',

`cust\_phone` varchar(64) DEFAULT NULL COMMENT '固定电话',

`cust\_mobile` varchar(16) DEFAULT NULL COMMENT '移动电话',

PRIMARY KEY (`cust\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

CREATE TABLE `cst\_linkman` (

`lkm\_id` bigint(32) NOT NULL AUTO\_INCREMENT COMMENT '联系人编号(主键)',

`lkm\_name` varchar(16) DEFAULT NULL COMMENT '联系人姓名',

`lkm\_cust\_id` bigint(32) DEFAULT NULL COMMENT '客户id',

`lkm\_gender` char(1) DEFAULT NULL COMMENT '联系人性别',

`lkm\_phone` varchar(16) DEFAULT NULL COMMENT '联系人办公电话',

`lkm\_mobile` varchar(16) DEFAULT NULL COMMENT '联系人手机',

`lkm\_email` varchar(64) DEFAULT NULL COMMENT '联系人邮箱',

`lkm\_qq` varchar(16) DEFAULT NULL COMMENT '联系人qq',

`lkm\_position` varchar(16) DEFAULT NULL COMMENT '联系人职位',

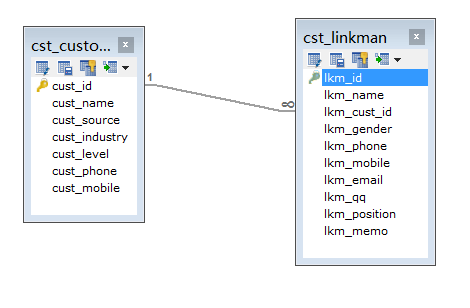
`lkm\_memo` varchar(512) DEFAULT NULL COMMENT '联系人备注',

PRIMARY KEY (`lkm\_id`),

KEY `FK\_cst\_linkman\_lkm\_cust\_id` (`lkm\_cust\_id`),

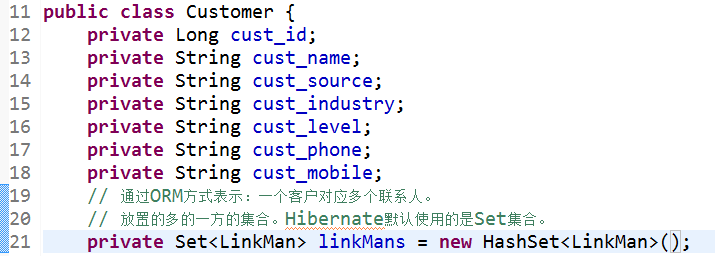
CONSTRAINT `FK\_cst\_linkman\_lkm\_cust\_id` FOREIGN KEY (`lkm\_cust\_id`) REFERENCES `cst\_customer` (`cust\_id`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

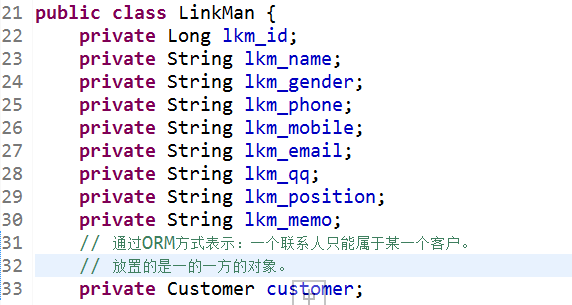


#### 创建实体

* 一的一方的实体



* 多的一方的实体



#### 创建映射文件

* 多的一方的映射的创建



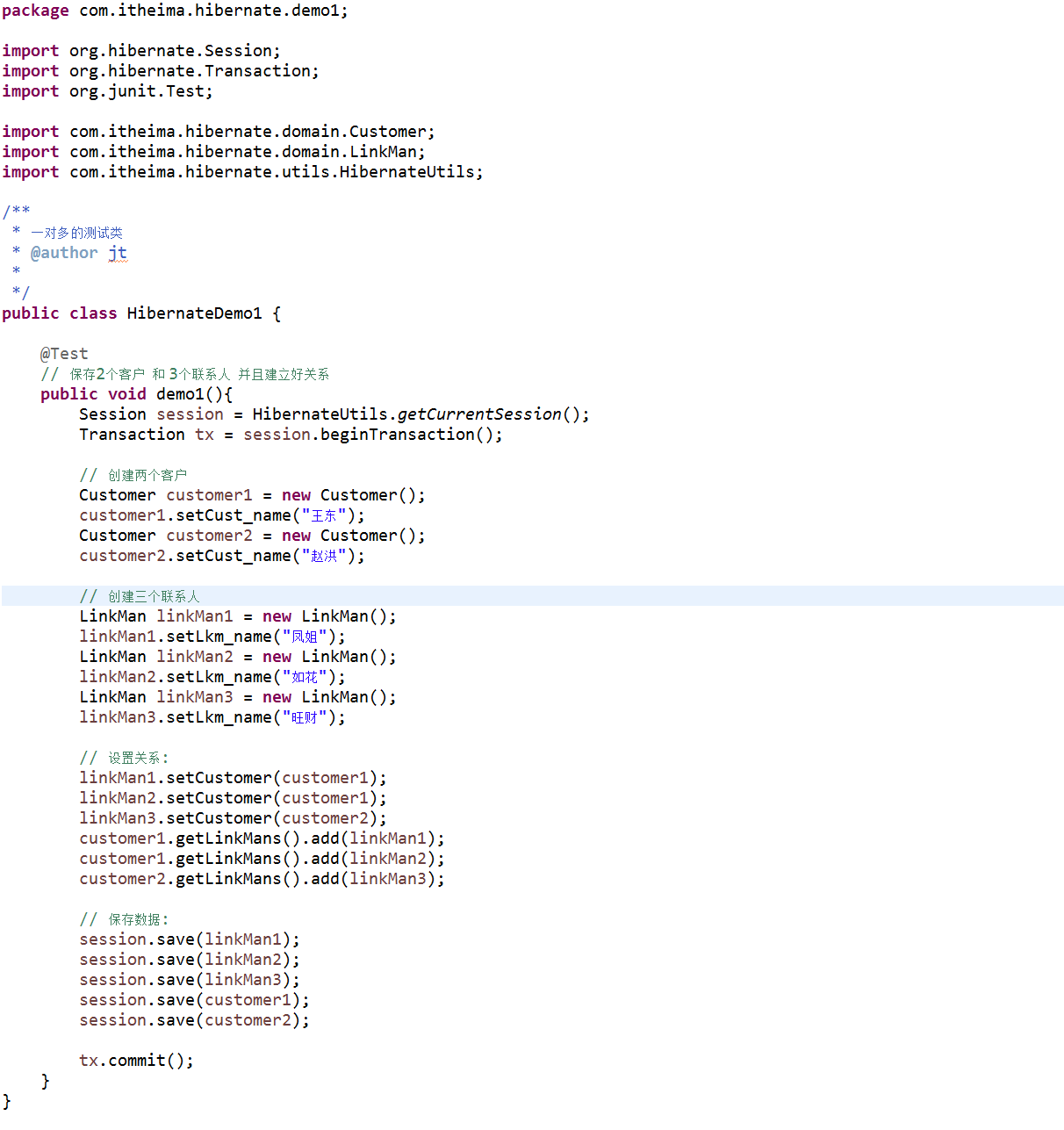
* 一的一方的映射的创建



#### 创建核心配置文件

#### 引入工具类

#### 编写测试类



### Hibernate的一对多相关操作

#### 一对多关系只保存一边是否可以：

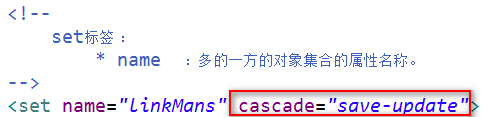


#### 一对多的级联操作

* 什么叫做级联
  + 级联指的是，操作一个对象的时候，是否会同时操作其关联的对象。
* 级联是有方向性
  + 操作一的一方的时候，是否操作到多的一方
  + 操作多的一方的时候，是否操作到一的一方

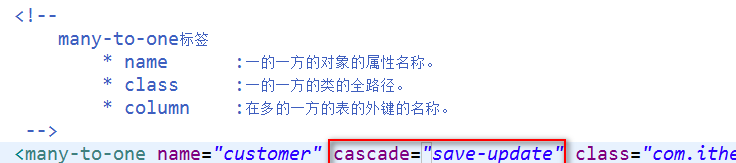
#### 级联保存或更新

* 保存客户级联联系人



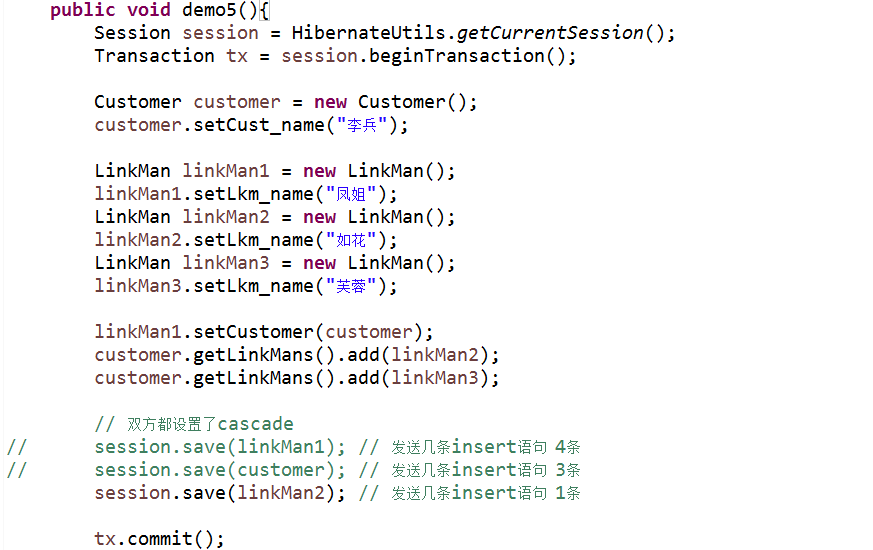


* 保存联系人级联客户





#### 测试对象的导航

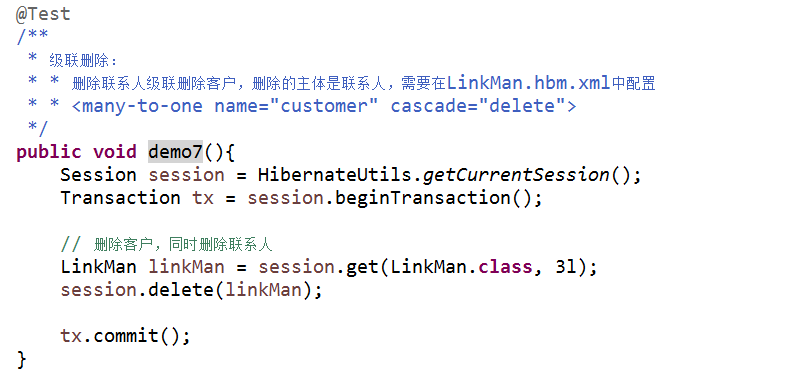


#### 级联删除

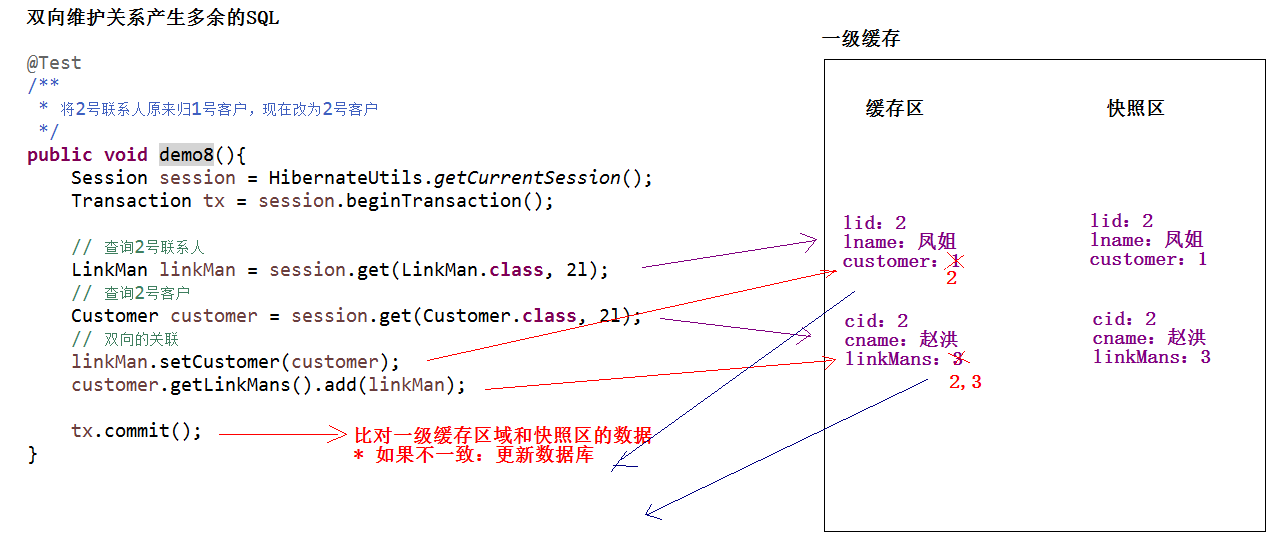
* 级联删除：
  + 删除一边的时候，同时将另一方的数据也一并删除。
* 删除客户级联删除联系人



* 删除联系人级联删除客户（基本不用）



#### 一对多设置了双向关联产生多余的SQL语句



* 解决多余的SQL语句
  + 单向维护：
  + 使一方放弃外键维护权：
    - 一的一方放弃。**在set上配置inverse=”true”**
  + **一对多的关联查询的修改的时候。（CRM练习--）**

#### 区分cascade和inverse



## Hibernate的多对多关联映射

### Hibernate多对多关系的配置

#### 创建表

* 用户表

CREATE TABLE `sys\_user` (

`user\_id` bigint(32) NOT NULL AUTO\_INCREMENT COMMENT '用户id',

`user\_code` varchar(32) NOT NULL COMMENT '用户账号',

`user\_name` varchar(64) NOT NULL COMMENT '用户名称',

`user\_password` varchar(32) NOT NULL COMMENT '用户密码',

`user\_state` char(1) NOT NULL COMMENT '1:正常,0:暂停',

PRIMARY KEY (`user\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

* 角色表

CREATE TABLE `sys\_role` (

`role\_id` bigint(32) NOT NULL AUTO\_INCREMENT,

`role\_name` varchar(32) NOT NULL COMMENT '角色名称',

`role\_memo` varchar(128) DEFAULT NULL COMMENT '备注',

PRIMARY KEY (`role\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

* 中间表

CREATE TABLE `sys\_user\_role` (

`role\_id` bigint(32) NOT NULL COMMENT '角色id',

`user\_id` bigint(32) NOT NULL COMMENT '用户id',

PRIMARY KEY (`role\_id`,`user\_id`),

KEY `FK\_user\_role\_user\_id` (`user\_id`),

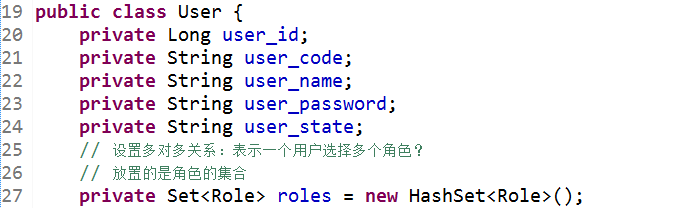
CONSTRAINT `FK\_user\_role\_role\_id` FOREIGN KEY (`role\_id`) REFERENCES `sys\_role` (`role\_id`) ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT `FK\_user\_role\_user\_id` FOREIGN KEY (`user\_id`) REFERENCES `sys\_user` (`user\_id`) ON DELETE NO ACTION ON UPDATE NO ACTION

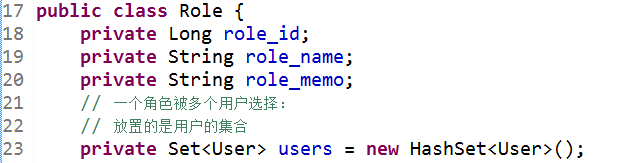
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#### 创建实体

* 用户的实体



* 角色的实体

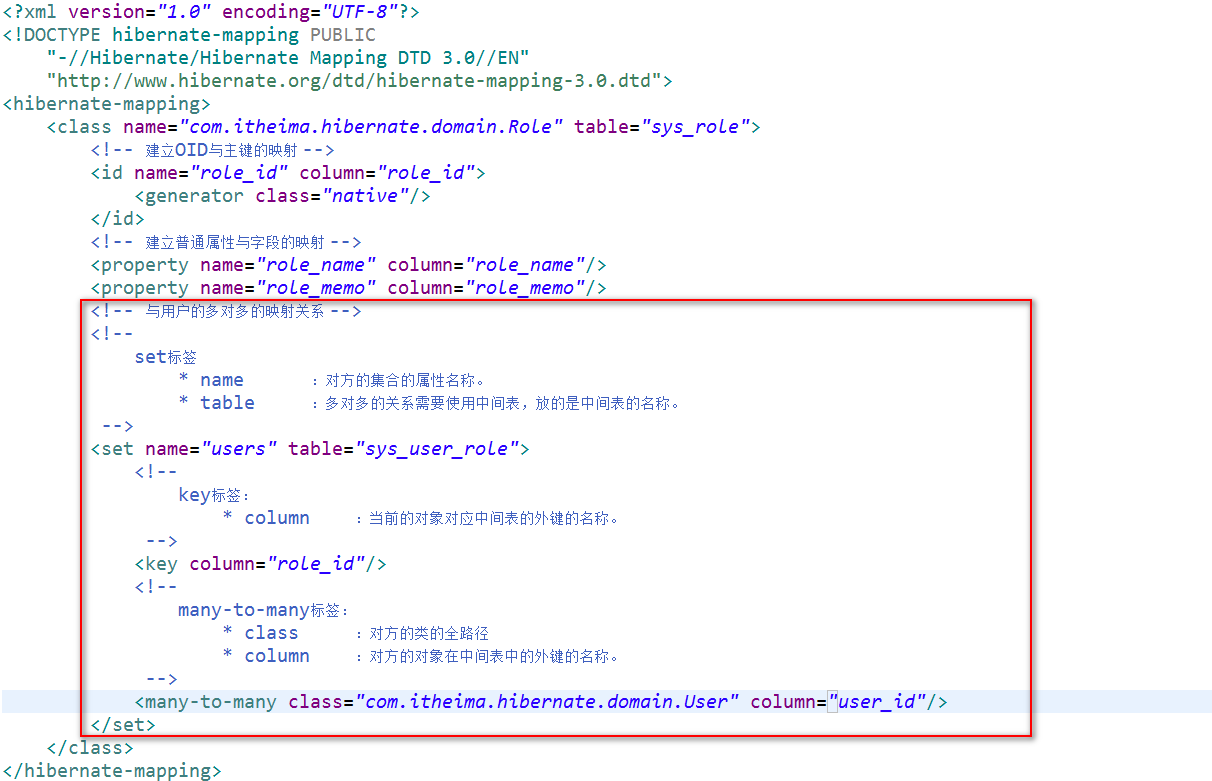


#### 创建映射

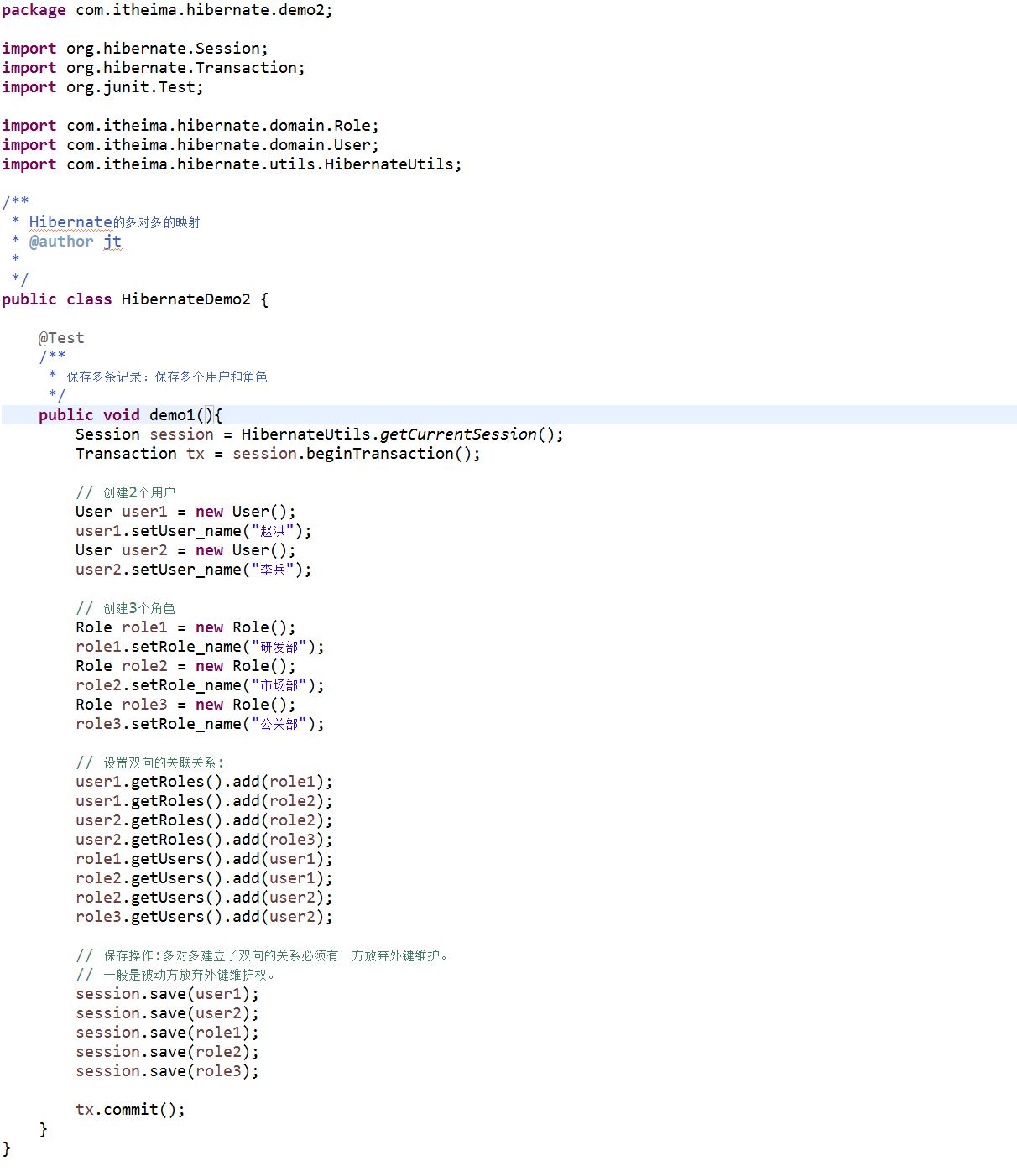
* 用户的映射



* 角色的映射



#### 编写测试类



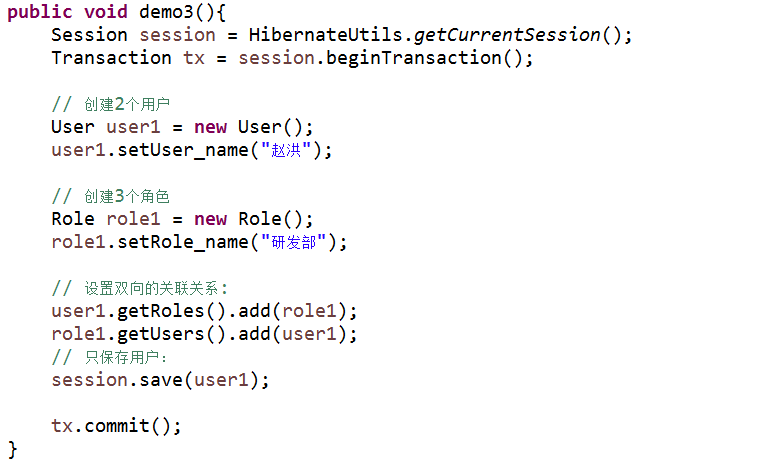
### Hibernate的多对多的操作

#### 只保存一边是否可以

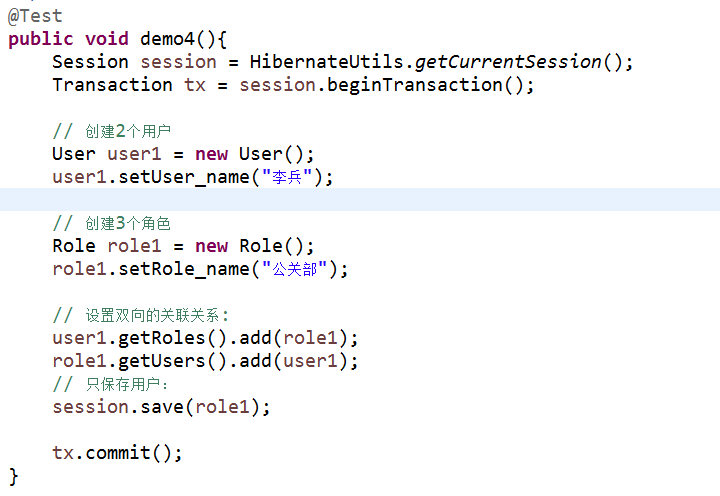


#### 多对多的级联保存或更新

* 保存用户级联保存角色

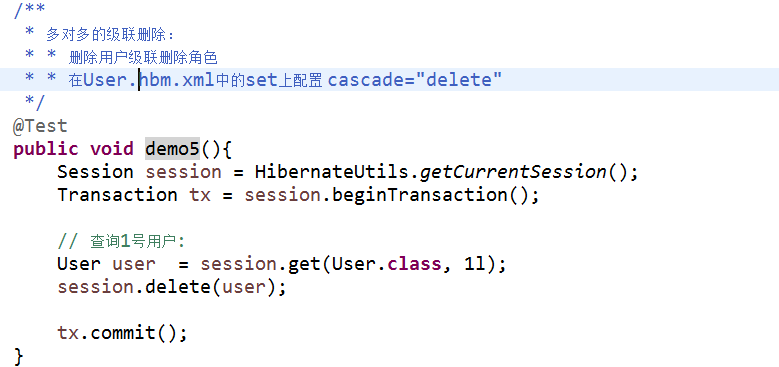


* 保存角色级联保存用户

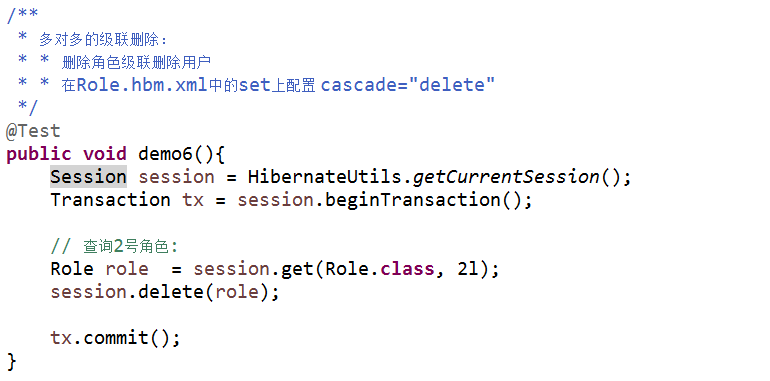


#### 多对多的级联删除（基本用不上）

* 删除用户级联删除角色

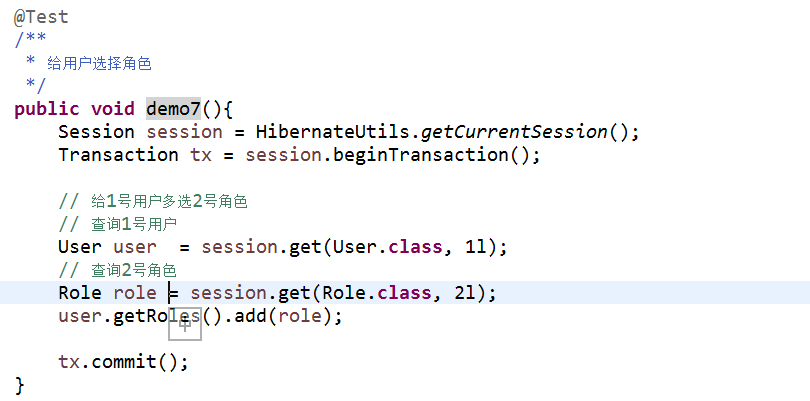


* 删除角色级联删除用户

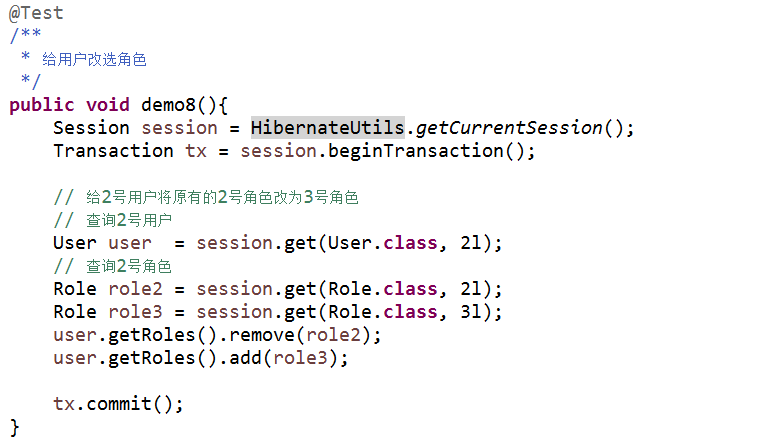


#### 多对多的其他的操作

* 给用户选择角色



* 给用户改选角色



* 给用户删除角色

