需求：用户和角色之间的关系是典型的多对多关系。

用户端：

```
// 关联用户
        @ManyToMany(targetEntity = role.class)
        //targetEntity =另一端实体类的类名.class
        @JoinTable(name = "t_tuser_trole",
        joinColumns = @JoinColumn(name = "tuser_id"),
        inverseJoinColumns = @JoinColumn(name = "trole_id"))
        // name为中间表的名字,joinColumns表示当前方在中间表中的字段名tuser_id
        // inverseJoinColumns对方在中间表中的字段名trole_id

        private Set<role> roles = new HashSet<role>();
```

角色端：

```
// 关联角色
        @ManyToMany(targetEntity = user.class, mappedBy = "roles")
        //targetEntity = 另一端实体类的名字.class,
        //mappedBy = "该实体在另一端实体中的属性名称"
        private Set<user> users = new HashSet<user>();
```

hibernate.cfg.xml配置文件：

```
<mapping class="多端其中一方的实体路径" />
<mapping class="多端另一方的实体路径" />
```

## demo操作代码：

```
/**
         * 多对多映射
         */
        @Test
        public void test3(){
                Session session = hibernateUtils.getSession();
                Transaction ts = session.beginTransaction();
                user u1 = new user();
                u1.setId(1);
                u1.setUsername("aaa");
                user u2 = new user();
                u2.setId(2);
                u2.setUsername("bbb");

                role r1 = new role();
                r1.setId(1);
                r1.setRolename("111");
                role r2 = new role();
                r2.setId(2);
                r2.setRolename("222");

                u1.getRoles().add(r2);
                r1.getUsers().add(u1);
                u2.getRoles().add(r1);
                r2.getUsers().add(u2);

                session.save(r2);
                session.save(r1);
                session.save(u1);
                session.save(u2);
                ts.commit();
                session.close();
        }
```