user实体：

```
package hibernate3;
import java.io.Serializable;
import java.util.HashSet;
import java.util.Set;
/**
 * 用户多方
 * @author Anly
 *
 */
public class user implements Serializable{
public Set<role> getRoles() {
                return roles;
        }
        public void setRoles(Set<role> roles) {
                this.roles = roles;
        }
private Integer id;
private String name;
private Set<role> roles = new HashSet<role>();
@Override
public String toString() {
        return "user [id=" + id + ", name=" + name + "]";
}
public Integer getId() {
        return id;
}
public void setId(Integer id) {
        this.id = id;
}
public String getName() {
        return name;
}
public void setName(String name) {
```

```
        this.name = name;
}
}
```

role实体：

```
package hibernate3;
import java.io.Serializable;
import java.util.HashSet;
import java.util.Set;
/**
 * 角色
 * @author Anly
 *
 */
public class role implements Serializable{
private Integer id;
private String name;
private Set<user>users = new HashSet<user>();
public Set<user> getUsers() {
        return users;
}
public void setUsers(Set<user> users) {
        this.users = users;
}
@Override
public String toString() {
        return "role [id=" + id + ", name=" + name + "]";
}
public Integer getId() {
        return id;
}
public void setId(Integer id) {
        this.id = id;
}
public String getName() {
```

```
        return name;
}
public void setName(String name) {
        this.name = name;
}
}
```

user.hbm.xml配置文件：

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping package="hibernate3">
        <class name="user" table="t_user">
                <id name="id" column="id">
                        <generator class="native"></generator>
                </id>
                <property name="name" column="name"></property>
                <!-- 多对多映射配置-->
                <!-- table中间表的表名 -->
                <set name="roles" table="t_user_role" cascade="all">
                        <!-- 当前方在中间表的外键 -->
                        <key column="user_id"></key>
                        <!-- column对方在中间表的外键 -->
                        <many-to-many class="role" column="role_id"/>
                </set>
        </class>
</hibernate-mapping>
```

role.hbm.xml配置文件：

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping package="hibernate3">
        <class name="role" table="t_role">
```

```
            <id name="id" column="id">
                <generator class="native"></generator>
            </id>
            <property name="name" column="name"></property>
            <!-- 多对多映射配置-->
            <!-- table中间表的表名 -->
            <set name="users" table="t_user_role" inverse="true">
                <!-- 当前方在中间表的外键 -->
                <key column="role_id"></key>
                <!-- column对方在中间表的外键 -->
                <many-to-many class="user" column="user_id"/>
            </set>
        </class>
</hibernate-mapping>
```

configuration.cfg.xml配置文件：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
 <hibernate-configuration>
<session-factory>
        <property name="hibernate.hbm2ddl.auto">update</property>

        <!-- 配置数据库连接的驱动器 -->
        <property name="hibernate.connection.driver_class">
            com.mysql.jdbc.Driver
        </property>
        <!-- 配置数据库连接的url -->
        <property name="hibernate.connection.url">
            jdbc:mysql://localhost:3306/hb2
        </property>
        <!-- 配置登录数据库的用户名 -->
        <property name="hibernate.connection.username">root</property>
        <!-- 配置登录数据库的密码 -->
```

```
        <property name="hibernate.connection.password">AQL271422</property>
        <!-- 配置mysql方言：解决不同数据库连接的不同的操作，从而生成不同的SQL语句-->
        <!--    <property name="hibernate.dialect">
                org.hibernate.dialect.MySQLDialect
                </property> -->
        <!-- show_sql操作数据库时，会向控制台打印SQL语句 -->
        <property name="hibernate.show_sql">true</property>
        <!-- format_sql操作数据库时，会将SQL语句先格式化 -->
        <property name="hibernate.format_sql">true</property>
        <!--  hbm2ddl.auto是否需要hibernate去维护这个表-->
        <!-- 事务自动提交 -->
        <property name="hibernate.connection.autocommit">true</property>

        <!-- <mapping resource="yiduiduo/orderss.hbm.xml"/>
                <mapping resource="yiduiduo/user1.hbm.xml"/> -->
        <!--    <mapping resource="one2many/customer.hbm.xml"/> -->
        <mapping resource="hibernate3/role.hbm.xml" />
        <mapping resource="hibernate3/user.hbm.xml" />

</session-factory>
</hibernate-configuration>
```

测试类：

```
package test;

import java.util.Set;

import org.hibernate.Session;
import org.hibernate.Transaction;
import org.junit.Test;

import hibernate3.role;
import hibernate3.user;
import utils.hibernateUtils;
```

```
/**
 * 演示多对多的操作
 *
 * @author Anly
 *
 */
public class demo {
	// 添加
	@Test
	public void te1() {
		user u1 = new user();
		u1.setName("安启力");

		user u2 = new user();
		u2.setName("刘丽");

		role r1 = new role();
		r1.setName("超级管理员");

		role r2 = new role();
		r2.setName("超级管理员");

		// 建立双向关系
		u1.getRoles().add(r1);
		u1.getRoles().add(r2);
		u2.getRoles().add(r2);
		u2.getRoles().add(r2);

		r1.getUsers().add(u1);
		r2.getUsers().add(u2);
		Session session = hibernateUtils.getSession();
		Transaction ts = session.beginTransaction();
		session.save(u1);
		session.save(u2);
```

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

// 查询
@Test
public void te2() {

        Session session = hibernateUtils.getSession();
        Transaction ts = session.beginTransaction();
        // 查询一个用户：看该用户的角色是什么
        user u1 = session.get(user.class, 21);
        Set<role> roles = u1.getRoles();
        System.out.println("当前用户为：" + u1.getName());
        System.out.println("当前用户的角色是：");
        for (role role : roles) {
                System.out.println(role.getName());
        }
        ts.commit();
        session.close();
}

// 级联添加
@Test
public void te3() {

        Session session = hibernateUtils.getSession();
        Transaction ts = session.beginTransaction();
        user u1 = new user();
        u1.setName("wehjhfd");
        role r1 = new role();
        r1.setName("二级管理员");
        // 级联操作
```

```
            u1.getRoles().add(r1);
            session.save(u1);
            ts.commit();
            session.close();
        }

        // 级联删除
        @Test
        public void te4() {
            //删除用户，同时删除中间表和用户相关的数据，HIA还删除了对应的角色的数据，
            //通常在实际开发中，在多对多的关系中不建议使用级联删除
            Session session = hibernateUtils.getSession();
            Transaction ts = session.beginTransaction();
            user us = session.get(user.class,18);
            session.delete(us);
            ts.commit();
            session.close();
        }
}
```