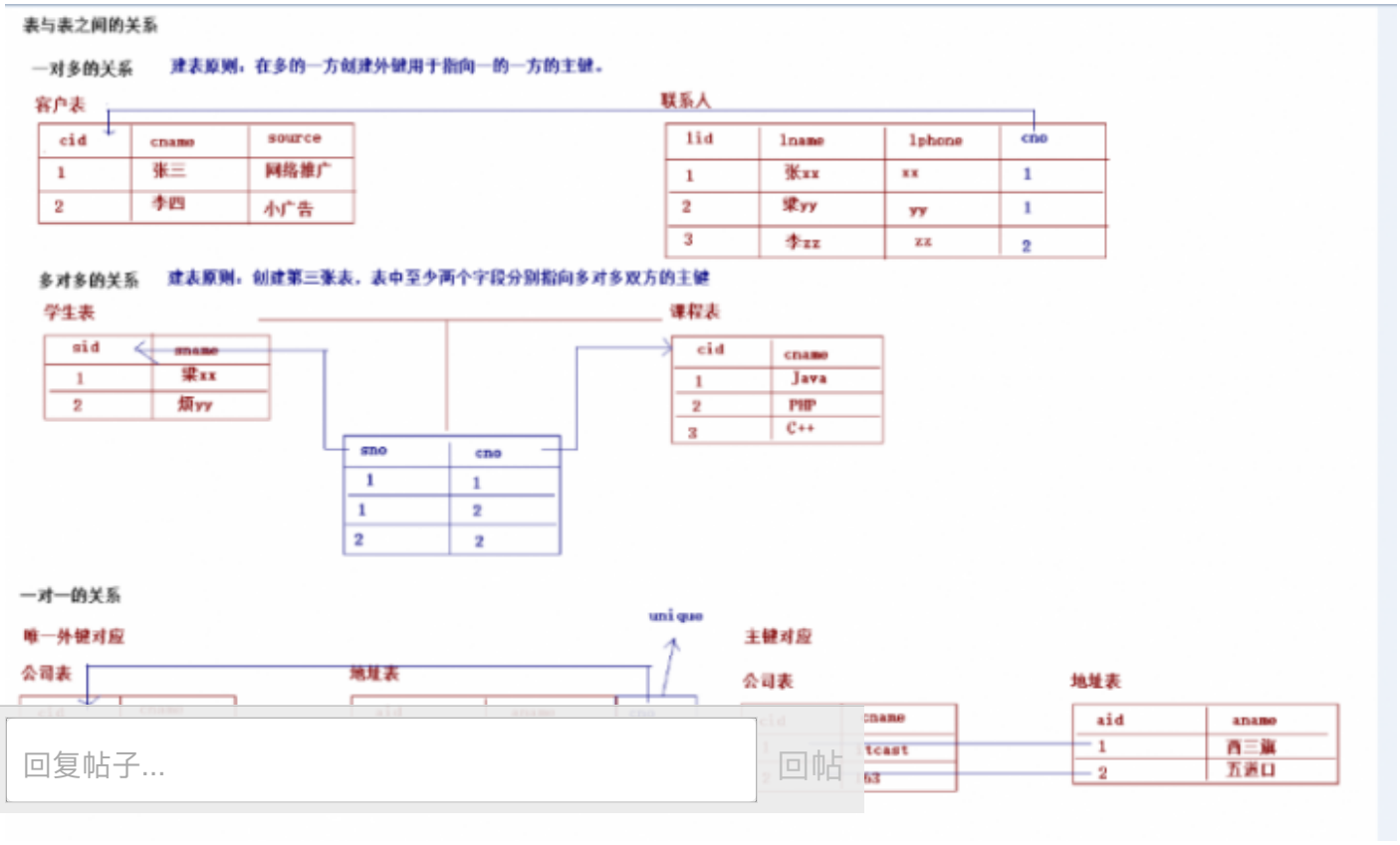
【济南中心】JavaEE就业班同步笔记第三阶段： Hibernate-part03

小鲁哥哥 • 2017-6-30 22:09:27

【济南中心】JavaEE就业班同步笔记第三阶段：Hibernate-part03

1.1 表关系的分析



回复帖子...

回帖

1.2 Hibernate关联关系：一对多的映射

1.2.1 搭建Hibernate基本开发环境

1.2.2 搭建Hibernate的一对多的关联关系映射

1.2.2.1 创建客户表和联系人表

1.2.2.2 创建实体类

【客户的实体】

```
[mw_shl_code=java,true]public class Customer {
	private Long cust_id;
	private String cust_name;
	private String cust_source;
	private String cust_industry;
	private String cust_level;
	private String cust_phone;
	private String cust_mobile;

	// 联系人的集合:
	private Set<LinkMan> linkMans = new HashSet<LinkMan>();
}[/mw_shl_code]
```

【联系人的实体】

```
[mw_shl_code=java,true]public class LinkMan {
	private Long lkm_id;
	private String lkm_name;
	private String lkm_gender;
	private String lkm_phone;
	private String lkm_mobile;
	private String lkm_email;
	private String lkm_qq;
	private String lkm_position;
	private String lkm_memo;

	// private Long lkm_cust_id;
	private Customer customer;// 联系人对应的客户的对象
}[/mw_shl_code]
```

1.2.2.3 创建映射

【联系人的映射】

```
[mw_shl_code=xml,true]<?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>
        <class name="com.itheima.hibernate.domain.LinkMan" table="cst_linkman">
                <id name="lkm_id">
                        <generator class="native"></generator>
                </id>
                <property name="lkm_name"/>
                <property name="lkm_gender"/>
                <property name="lkm_phone"/>
                <property name="lkm_mobile"/>
                <property name="lkm_email"/>
                <property name="lkm_qq"/>
                <property name="lkm_position"/>
                <property name="lkm_memo"/>

                <!-- 在多的一方配置many-to-one -->
                <!--
                        many-to-one标签：用来描述多对一的关系配置。
                                * name                :一的一方的对象的属性名称。
                                * class               :一的一方的类的全路径
                                * column        :外键的名称
                 -->
                <many-to-one name="customer" class="com.itheima.hibernate.domain.Customer" column="lkm
_cust_id"/>
        </class>
</hibernate-mapping>[/mw_shl_code]
```

【客户的映射】

```
[mw_shl_code=xml,true]<?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>
        <class name="com.itheima.hibernate.domain.Customer" table="cst_customer">
                <id name="cust_id">
                        <generator class="native"/>
                </id>
```

```
        <property name="cust_name"/>
        <property name="cust_source"/>
        <property name="cust_industry"/>
        <property name="cust_level"/>
        <property name="cust_phone"/>
        <property name="cust_mobile"/>

        <!-- 配置一对多的关联映射 -->
        <!--
            set标签：用来描述一的一方存放的多的一方的集合
                * name:多的一方的集合属性名称
        -->
        <set name="linkMans">
            <!--
                key标签：
                    * column多的一方的外键的名称
            -->
            <key column="lkm_cust_id"/>
            <!--
                one-to-many标签：描述一对多的关系
                    * class:多的一方的类的全路径
             -->
            <one-to-many class="com.itheima.hibernate.domain.LinkMan"/>
        </set>
    </class>
</hibernate-mapping>[/mw_shl_code]
```

1.2.2.4 在核心配置中加载映射

```
<!-- 加载映射文件 -->
<mapping resource="com/itheima/hibernate/domain/Customer.hbm.xml"/>
<mapping resource="com/itheima/hibernate/domain/LinkMan.hbm.xml"/>
```

1.2.2.5 编写测试方法

[mw_shl_code=java,true]@Test
/**
 * 保存数据
 */

```
public void demo1(){
    Session session = HibernateUtils.getCurrentSession();
    Transaction tx = session.beginTransaction();

    // 创建两个客户
    Customer customer1 = new Customer();
    customer1.setCust_name("柳岩");
    Customer customer2 = new Customer();
    customer2.setCust_name("马蓉");

    // 创建三个联系人
    LinkMan linkMan1 = new LinkMan();
    linkMan1.setLkm_name("梁大鹏");
    LinkMan linkMan2 = new LinkMan();
    linkMan2.setLkm_name("梁宝强");
    LinkMan linkMan3 = new LinkMan();
    linkMan3.setLkm_name("梁喆");

    // 建立关系:双向关系.
    customer1.getLinkMans().add(linkMan1);
    customer2.getLinkMans().add(linkMan2);
    customer2.getLinkMans().add(linkMan3);
    linkMan1.setCustomer(customer1);
    linkMan2.setCustomer(customer2);
    linkMan3.setCustomer(customer2);

    session.save(customer1);
    session.save(customer2);
    session.save(linkMan1);
    session.save(linkMan2);
    session.save(linkMan3);

    tx.commit();
}[/mw_shl_code]1.2.3 Hibernate的一对多的级联操作
```

1.2.3.1 级联保存或更新

级联操作：操作一个对象的时候，同时操作其关联的对象。

级联是有方向性的，保存客户级联联系人，还是要保存联系人级联客户。

【保存客户的同时级联保存联系人】

[mw_shl_code=java,true]@Test

```
/**
 * 级联保存：
 *        * 保存客户同时级联保存联系人。
 *        * 保存的主体对象是客户对象，那么就需要在Customer.hbm.xml文件中配置。
 *                * 在<set>标签上配置cascade="save-update"
 */
public void demo3(){
        Session session = HibernateUtils.getCurrentSession();
        Transaction tx = session.beginTransaction();

        // 创建客户
        Customer customer1 = new Customer();
        customer1.setCust_name("柳岩");

        // 创建联系人
        LinkMan linkMan1 = new LinkMan();
        linkMan1.setLkm_name("梁大鹏");


        // 建立关系:双向关系.
        customer1.getLinkMans().add(linkMan1);
        linkMan1.setCustomer(customer1);



        session.save(customer1);

        tx.commit();
}
```

[/mw_shl_code]【保存联系人同时级联保存客户】

[mw_shl_code=java,true]@Test

```
/**
 * 级联保存：
 *        * 保存联系人同时级联保存客户。
 *        * 保存的主体对象是联系人对象，那么就需要在LinkMan.hbm.xml文件中配置。
```

```
 *                * 在<many-to-one>标签上配置cascade="save-update"
 */
public void demo4(){
        Session session = HibernateUtils.getCurrentSession();
        Transaction tx = session.beginTransaction();

        // 创建客户
        Customer customer1 = new Customer();
        customer1.setCust_name("马蓉");

        // 创建联系人
        LinkMan linkMan1 = new LinkMan();
        linkMan1.setLkm_name("烦司机");

        // 建立关系:双向关系.
        customer1.getLinkMans().add(linkMan1);
        linkMan1.setCustomer(customer1);

        session.save(linkMan1);

        tx.commit();
}
```
[/mw_shl_code]1.2.3.2 测试对象导航

[mw_shl_code=java,true]
```
@Test
/**
 * 测试对象导航
 *         * 一对多的双方都配置了cascade="save-update"
 */
public void demo5(){
        Session session = HibernateUtils.getCurrentSession();
        Transaction tx = session.beginTransaction();

        // 创建1个客户和3个联系人
        Customer customer1 = new Customer();
        customer1.setCust_name("梁某");

        // 创建3个联系人
        LinkMan linkMan1 = new LinkMan();
```

```
        linkMan1.setLkm_name("如花");
        LinkMan linkMan2 = new LinkMan();
        linkMan2.setLkm_name("凤姐");
        LinkMan linkMan3 = new LinkMan();
        linkMan3.setLkm_name("芙蓉");

        // 建立了关系
        linkMan1.setCustomer(customer1);
        customer1.getLinkMans().add(linkMan2);
        customer1.getLinkMans().add(linkMan3);

        // session.save(linkMan1); // 发送几条insert语句？ 4条
        // session.save(customer1); // 发送几条insert语句？ 3条
        session.save(linkMan2); // 发送几条insert语句？1条

        tx.commit();
    }
```

[/mw_shl_code]1.2.3.3 级联删除

【删除客户同时级联删除联系人】

[mw_shl_code=java,true]@Test

```
    /**
     * 级联删除：
     *        * 删除客户同时级联删除联系人。
     *        * 在Customer.hbm.xml中的<set>上配置cascade="delete"
     */
    public void demo7(){
        Session session = HibernateUtils.getCurrentSession();
        Transaction tx = session.beginTransaction();

        // 删除2号客户
        Customer customer = session.get(Customer.class, 2l);
        session.delete(customer);

        tx.commit();
    }
```

[/mw_shl_code] 【删除联系人同时级联删除客户】

[mw_shl_code=java,true]@Test

```
    /**
```

```
 * 级联删除：
 *        * 删除联系人同时级联删除客户。
 *        * 在LinkMan.hbm.xml中在<many-to-one>上配置cascade="delete"
 */
public void demo8(){
        Session session = HibernateUtils.getCurrentSession();
        Transaction tx = session.beginTransaction();

        // 删除1号联系人
        LinkMan linkMan = session.get(LinkMan.class, 1l);
        session.delete(linkMan);

        tx.commit();
}
```

[/mw_shl_code]1.2.4 双向维护关系产生多余的SQL

1.2.4.1 编写代码测试:

```
[mw_shl_code=java,true]@Test
/**
 * 双向维护关系 导致多余的SQL
 */
public void demo9(){
        Session session = HibernateUtils.getCurrentSession();
        Transaction tx = session.beginTransaction();

        Customer customer = session.get(Customer.class, 1l);
        LinkMan linkMan = session.get(LinkMan.class, 2l);

        customer.getLinkMans().add(linkMan);
        linkMan.setCustomer(customer);

        tx.commit();
}
```

[/mw_shl_code]1.2.4.2 区分cascade和inverse

```
[mw_shl_code=java,true]@Test
/**
 * 区分cascade和inverse
 *        * 在客户端配置cascade="save-update" inverse="true"
 */
```

```
    public void demo10(){
        Session session = HibernateUtils.getCurrentSession();
        Transaction tx = session.beginTransaction();

        Customer customer = new Customer();
        customer.setCust_name("高松");

        LinkMan linkMan = new LinkMan();
        linkMan.setLkm_name("高凤儿");

        customer.getLinkMans().add(linkMan);

        session.save(customer);

        tx.commit();
    }
```

1.3 Hibernate关联关系：多对多

1.3.1 搭建Hibernate多对多开发环境

1.3.1.1 创建实体

【用户的实体】

```
public class User {
    private Long user_id;
    private String user_code;
    private String user_name;
    private String user_password;
    private String user_state;

    // 一个用户选择多个角色
    private Set<Role> roles = new HashSet<Role>();
}
```

【角色实体】

```
public class Role {
    private Long role_id;
    private String role_name;
    private String role_memo;
```

```
    // 一个角色被多个用户选择:
    private Set<User> users = new HashSet<User>();
}[/mw_shl_code]
```

1.3.1.2 创建映射

【用户的映射】

```
[mw_shl_code=java,true]<?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>
    <class name="com.itheima.hibernate.domain.User" table="sys_user">
        <id name="user_id" >
            <generator class="native"/>
        </id>
        <property name="user_code"/>
        <property name="user_name"/>
        <property name="user_password"/>
        <property name="user_state"/>

        <!-- 配置多对多的映射 -->
        <!--
            set标签
                * name       :多对多另一方的集合的属性名称
                * table      :多对多建立的中间表的名称。
         -->
        <set name="roles" table="sys_user_role">
            <!--
                key标签:描述外键
                    * column：当前对象在中间表的外键的名称
             -->
            <key column="user_no"/>
            <!--
                many-to-many标签：多对多配置
                    * class:对方的类的全路径
                    * column:对方在中间表的外键名称
             -->
```

```
            <many-to-many class="com.itheima.hibernate.domain.Role" column="role_no"/>
        </set>
    </class>
</hibernate-mapping>
```

【角色的映射】

```
<?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>
    <class name="com.itheima.hibernate.domain.Role" table="sys_role">
        <id name="role_id">
            <generator class="native"/>
        </id>

        <property name="role_name"/>
        <property name="role_memo"/>

        <set name="users" table="sys_user_role">
            <key column="role_no"/>
            <many-to-many class="com.itheima.hibernate.domain.User" column="user_no"/>
        </set>
    </class>
</hibernate-mapping>
```

1.3.1.3 在核心配置中加载映射

```
<mapping resource="com/itheima/hibernate/domain/User.hbm.xml"/>
<mapping resource="com/itheima/hibernate/domain/Role.hbm.xml"/>
```

1.3.1.4 编写测试类：

```
@Test
/**
 * 保存数据:
 *       * 多对多保存的时候，需要单向关联。如果是双向关联需要有一方放弃外键维护权。
 */
public void demo1(){
    Session session = HibernateUtils.getCurrentSession();
```

```
        Transaction tx = session.beginTransaction();

        // 创建2个用户
        User user1 = new User();
        user1.setUser_name("梁司机");
        User user2 = new User();
        user2.setUser_name("梁蓉");

        // 创建3个角色
        Role role1 = new Role();
        role1.setRole_name("司机");
        Role role2 = new Role();
        role2.setRole_name("经济人");
        Role role3 = new Role();
        role3.setRole_name("讲师");

        // 建立关系:
        user1.getRoles().add(role1);
        user1.getRoles().add(role3);
        user2.getRoles().add(role1);
        user2.getRoles().add(role2);
        role1.getUsers().add(user1);
        role1.getUsers().add(user2);
        role2.getUsers().add(user2);
        role3.getUsers().add(user1);

        session.save(user1);
        session.save(user2);
        session.save(role1);
        session.save(role2);
        session.save(role3);
        tx.commit();
    }[/mw_shl_code]1.3.2 Hibernate的多对多的级联操作
```

1.3.2.1 级联保存或更新

【保存用户同时级联角色】

[mw_shl_code=java,true]@Test

```
/**
 * 保存用户同时级联角色
 * 在User.hbm.xml中的<set>上配置cascade="save-update"
 */
public void demo2(){
	Session session = HibernateUtils.getCurrentSession();
	Transaction tx = session.beginTransaction();

	User user = new User();
	user.setUser_name("梁某");

	Role role = new Role();
	role.setRole_name("司机");

	user.getRoles().add(role);
	role.getUsers().add(user);

	session.save(user);

	tx.commit();
}
```
[/mw_shl_code]【保存角色同时级联用】

```
[mw_shl_code=java,true]@Test
/**
 * 保存角色同时级联用户
 * 在Role.hbm.xml中的<set>上配置cascade="save-update"
 */
public void demo3(){
	Session session = HibernateUtils.getCurrentSession();
	Transaction tx = session.beginTransaction();

	User user = new User();
	user.setUser_name("梁某");

	Role role = new Role();
	role.setRole_name("司机");

	user.getRoles().add(role);
```

```
        role.getUsers().add(user);

        session.save(role);

        tx.commit();
    }
```
[/mw_shl_code]1.3.2.2 级联删除（了解）

【删除用户级联删除角色】

[mw_shl_code=java,true]@Test
```
    /**
     * 删除用户同时级联删除角色
     * 在User.hbm.xml中配置cascade="delete"
     */
    public void demo4(){
        Session session = HibernateUtils.getCurrentSession();
        Transaction tx = session.beginTransaction();

        User user = session.get(User.class, 2l);
        session.delete(user);

        tx.commit();
    }
```
[/mw_shl_code]【删除角色级联删除用户】

[mw_shl_code=java,true]@Test
```
    /**
     * 删除角色同时级联删除用户
     * 在Role.hbm.xml中配置cascade="delete"
     */
    public void demo5(){
        Session session = HibernateUtils.getCurrentSession();
        Transaction tx = session.beginTransaction();

        Role role = session.get(Role.class, 3l);
        session.delete(role);

        tx.commit();
    }
```
[/mw_shl_code]1.3.3 Hibernate多对多的其他的操作：

1.3.3.1 为某个用户添加角色

[mw_shl_code=java,true]@Test
/**
 * 给1号用户添加2号角色
 */
public void demo6(){
	Session session = HibernateUtils.getCurrentSession();
	Transaction tx = session.beginTransaction();
	// 查询一号用户
	User user = session.get(User.class, 1l);
	// 查询二号角色
	Role role = session.get(Role.class, 2l);
	user.getRoles().add(role);

	tx.commit();
}[/mw_shl_code]1.3.3.2 为用户删除某个角色

[mw_shl_code=java,true]@Test
/**
 * 给1号用户添加1号角色
 */
public void demo7(){
	Session session = HibernateUtils.getCurrentSession();
	Transaction tx = session.beginTransaction();
	// 查询一号用户
	User user = session.get(User.class, 1l);
	// 查询一号角色
	Role role = session.get(Role.class, 1l);
	user.getRoles().remove(role);

	tx.commit();
}[/mw_shl_code]1.3.3.3 为用户改选角色

[mw_shl_code=java,true]@Test
/**
 * 给2号用户将2号角色改为3号角色
 */
public void demo8(){
	Session session = HibernateUtils.getCurrentSession();

```
        Transaction tx = session.beginTransaction();

        User user = session.get(User.class, 2l);

        Role role2 = session.get(Role.class, 2l);
        Role role3 = session.get(Role.class, 3l);
        user.getRoles().remove(role2);
        user.getRoles().add(role3);

        tx.commit();
    }
```

扫码关注“黑马程序员视频库”公众号
获取更多学习资料

回帖

2条回帖

→_→←_←

不错，收藏学习ing

沙发 • 2017-7-6 08:33:26

回帖

str.arr

刚上就业班或许用得着

藤椅 • 2017-7-9 09:29:27

回帖