浙江工业大学

JavaEE技术实验报告

实验名称：Hibernate 关联关系映射——登录用户的地址管理

学 院： 计算机学院

班 级： 软件工程1801

姓 名： 陈南

学 号： 201806062101

组 号： 无

时 间： 2020年11月13日

**一、基础实验——多对一/一对多关联**

（一）实验目的

1、掌握 Hibernate 关联关系映射的基本概念，理解关联的方向和数量，重点理解

双向一对多/多对一的关联关系，及其在实际应用中的体现；

2、学习 Hibernate 框架处理一对多/多对一关联关系的方法，掌握关联关系中持

久化类的实现方法、以及相应 Hibernate 映射文件的配置方案；

3、能在实际应用中通过 Hibernate 建立正确的一对多/多对一关联关系映射，并

以面向对象的方式进行数据库访问。

（二）基本知识与原理

1、客观世界中的对象往往不孤立存在，例如老师与被授课的学生存在关联关系，

如果已经得到某老师的实例，那么应该可以获取该老师对应的全部学生，反

之如果已经得到一个学生的实例，也应该可以访问该学生对应的老师——这

种实例之间的相互访问就是关联关系；Hibernate 框架可以处理各种不同的关

联关系；

2、关联的方向可分为单向关联和双向关联：

（1）单向关联：只需单向访问关联端，例如只能通过老师访问学生，或者只能

通过学生访问老师；

（2）双向关联：关联的两端可以相互访问，例如老师和学生之间可以相互访问；

3、除考虑关联的方向问题之外，还要考虑关联双方的数量问题，即一对一、一

对多、多对一、多对多的关联关系；

4、双向的一对多/多对一关系是现实中最为常见的关联关系，假设实体类 A 到实

体类 B 是一对多（一个 A 的实例关联多个 B 的实例），则 B 到 A 就是多对

一（多个 B 的实例可能关联同一个 A 的实例）；要表示这种关系，则 B 类（多

的一端）中关联一个 A 的实例，而 A 类（一的一端）中关联一个集合对象，

集合元素为 B 的实例；

5、在 Hibernate 映射文件中，作为一的一端，需要使用<set.../>或<bag.../>元素来

映射关联属性；作为多的一端，则需要使用<many-to-one.../>元素来映射关联

9394

属性。

（三）实验内容及步骤

1、在 MySQL 中创建一个名称为 hibernatedb 的数据库，并在该数据库中创建一

个名称为 customer 的数据表，创建表的 DDL 如下：

CREATE TABLE `customer` (

`customerID` INTEGER(11) NOT NULL,

`account` VARCHAR(20) DEFAULT NULL,

`password` VARCHAR(20) DEFAULT NULL,

`name` VARCHAR(20) DEFAULT NULL,

`sex` TINYINT(1) DEFAULT NULL,

`birthday` DATE DEFAULT NULL,

`email` VARCHAR(100) DEFAULT NULL,

PRIMARY KEY (`customerID`)

)

2、在表 customer 中添加 3 条记录，具体如表 1-2 所示：

表 7-1 customer 中的记录

**customerID**

**account**

**password**

1

zjut

zjut

2

admin

admin

3

temp

temp

3、在 hibernatedb 数据库中创建一个名称为 address 的数据表，用于记录用户的

联系地址，customer 与 address 是一对多的关系，其中 address 表中的 cust\_id

是外键，参考 customer 表的主键，创建表的 DDL 如下：

CREATE TABLE `address` (

`addressID` INTEGER(11) NOT NULL,

`detail` VARCHAR(200) DEFAULT NULL,

`zipcode` VARCHAR(10) DEFAULT NULL,

`phone` VARCHAR(20) DEFAULT NULL,

`type` VARCHAR(20) DEFAULT NULL,

`cust\_id` INTEGER(11) DEFAULT NULL,

PRIMARY KEY (`addressID`),

FOREIGN KEY (`cust\_id`) REFERENCES customer (`customerID`)

)

4、在 Eclipse 中新建 Web 工程 hibernate-prj3，并添加 MySQL 驱动程序库文件、

commons-logging-1.2.jar、Struts2 核心包和 Hibernate 核心包到工程中；5、在 hibernate-prj3/src 中新建配置文件 hibernate.cfg.xml，具体代码可参照“实

验五 Hibernate 基础应用”中基础实验里的 hibernate.cfg.xml；

6、在 hibernate-prj3 中 新 建 cn.edu.zjut.po 包 ， 并 在 其 中 创 建 持 久 化 类

Customer.java 和 Address.java，Customer 与 Address 是一对多的关系，因此需

要在 Customer 中增加一个 Set 集合属性，用于记录它关联的一系列 Address

实体，而在 Address 中只需增加一个 Customer 类型的属性，代码片段如下：

/\*\* Customer \*\*/

package cn.edu.zjut.po;

……

public class Customer {

private int customerId;

private String account;

private String password;

private String repassword;

private String name;

private Boolean sex;

private String sexStr;

private Date birthday;

private String email;

**private Set addresses = new HashSet(0);**

//省略构造函数

//省略 getters/setters 方法

}

/\*\* Address \*\*/

package cn.edu.zjut.po;

……

public class Address {

private int addressId;

private String detail;

private String zipcode;

private String phone;

private String type;

**private Customer customer;**

//省略构造函数

//省略 getters/setters 方法

}

7、在 hibernate-prj3 的 cn.edu.zjut.po 包中新建映射文件 Customer.hbm.xml，作为

一的一端，需要使用<set.../>或<bag.../>元素来映射关联属性，在<set.../>或

<bag.../>元素中需要增加<key.../>子元素映射外键列，并使用<one-to-many.../>

子元素映射关联属性，代码片段如下：

95<hibernate-mapping>

<class name="cn.edu.zjut.po.Customer" table="customer"

catalog="hibernatedb">

<id name="customerId" type="int">

<column name="customerID" />

<generator class="increment" />

</id>

……

**<set name="addresses" inverse="true" cascade="all" lazy="false">**

**<key column="cust\_id"/>**

**<one-to-many class="cn.edu.zjut.po.Address"/>**

**</set>**

</class>

</hibernate-mapping>

8、在 hibernate-prj3 的 cn.edu.zjut.po 包中新建映射文件 Address.hbm.xml，作为

多的一端，需要使用<many-to-one.../>元素来映射关联属性，代码片段如下：

<hibernate-mapping>

<class name="cn.edu.zjut.po.Address" table="address"

catalog="hibernatedb">

<id name="addressId" type="int">

<column name="addressID" />

<generator class="increment" />

</id>

……

**<many-to-one name="customer" class="cn.edu.zjut.po.Customer"**

**fetch="select" not-null="true">**

**<column name="cust\_id"/>**

**</many-to-one>**

</class>

</hibernate-mapping>

9、修改配置文件 hibernate.cfg.xml，增加 Customer.hbm.xml 与 Address.hbm.xml

映射文件的声明；

10、在 hibernate-prj3 中新建 cn.edu.zjut.dao 包，并在其中创建 DAO 操作辅助类

HibernateUtil.java 和数据库操作基础类 BaseHibernateDAO.java（可参考“实

验六 Hibernate 的体系结构”中基础实验里的代码）；

11、在 cn.edu.zjut.dao 包中创建数据库操作类 CustomerDAO.java 和 AddressDAO.

java（代码略）；

12、在 hibernate-prj3 中新建 cn.edu.zjut.service 包，在其中创建 UserService.java，

并实现用户登录和增加地址的逻辑，代码片段如下：

package cn.edu.zjut.service;

……

96public class UserService {

public boolean login(Customer loginUser) {

//代码略

}

public boolean addAddr(Customer loginUser, Address address) {

ActionContext ctx= ActionContext.getContext();

request=(Map) ctx.get("request");

**CustomerDAO c\_dao = new CustomerDAO();**

**loginUser = (Customer)c\_dao**

**.findById(loginUser.getCustomerId());**

**address.setCustomer(loginUser); //注释 1**

**loginUser.getAddresses().add(address);**

**Transaction tran = null;**

try {

**tran = c\_dao.getSession().beginTransaction();**

**c\_dao.update(loginUser);**

**tran.commit();**

request.put("loginUser", loginUser);

request.put("tip", "添加地址成功！");

return true;

} catch (Exception e) {

**if(tran != null) tran.rollback();**

return false;

} finally {

c\_dao.getSession().close();

}

}

}

13、在 hibernate-prj3 中新建 cn.edu.zjut.action 包，并在其中创建 UserAction.java，

代码片段如下：

package cn.edu.zjut.action;

……

public class UserAction {

private Customer loginUser;

private Address address;

public String login() {

UserService userServ = new UserService();

if (userServ.login(loginUser))

return "success";

else

return "fail";

}

public String addAddr() {

UserService userServ = new UserService();

97if (userServ.addAddr(loginUser, address))

return "success";

else

return "fail";

}

//省略 getters/setters 方法

}

14、在 hibernate-prj3 中新建 login.jsp 页面，作为用户登录的视图（代码略）；

15、在 hibernate-prj3 中新建 loginSuccess.jsp 页面，作为登录成功的视图，在该

视图中显示登录用户的所有个人信息（包括地址信息），并在该视图中增加

“添加新地址”的表单，代码片段如下：

<table>

<tr><td>个人信息：<p></td></tr>

<tr>

<td>用户名：</td>

<td><s:property value="#request.loginUser.account" /></td>

</tr>

……

<s:iterator value="#request.loginUser.addresses" status="st">

<tr><td>地址<s:property value="#st.count"/>：</td><tr>

<tr><td>详细地址：</td><td><s:property value="detail" /></td><tr>

……

</s:iterator>

</table>

<hr>

添加新地址：<p>

<s:form action="UseraddAddr" method="post">

**<s:hidden name="loginUser.customerId"**

**value="%{#request.loginUser.customerId}"/>**

<s:textfield name="address.detail" label="详细地址" />

<s:textfield name="address.zipcode" label="邮政编码" />

<s:textfield name="address.phone" label="联系电话" />

<s:textfield name="address.type"

label="地址类型（office,home,etc.）" />

<s:submit value="添加"/>

</s:form>

16、在工程 hibernate-pr3 的 src 目录中创建 struts.xml 文件，用于配置 Action 并

设置页面导航，“登录成功”或“添加地址成功”都转向 loginSuccess.jsp 页

面（代码略）；

17、编辑 Web 应用的 web.xml 文件，增加 Struts2 核心 Filter 的配置（代码略）；

18、将 hibernate-prj3 部署在 Tomcat 服务器上；

19、通过浏览器访问 login.jsp 页面，并记录运行结果；

9820、修改 hibernate.cfg.xml 配置文件，增加属性使得能在控制台输出 SQL 语句，

将 hibernate-prj3 重新部署在 Tomcat 服务器上并运行，当 Customer.hbm.xml

中 set 元素的 inverse 属性为 true 时，观察并记录控制台输出的 SQL 语句，再

将 inverse 属性值设置成 false，观察并记录控制台输出的 SQL 语句；

21、修改 Customer.hbm.xml 中 set 元素的 lazy 属性值为 true，将 hibernate-prj3 重

新部署在 Tomcat 服务器上并运行，观察并记录 loginSuccess.jsp 页面的输出；

22、修改 UserService.java，将“注释 1”处的代码删除，将 hibernate-prj3 重新部

署在 Tomcat 服务器上并运行，观察并记录运行结果。

（四）实验要求

1、填写并上交实验报告，报告中应包括：

（1）运行结果截图；

（2）根据实验步骤 6-8，总结双向一对多/多对一关联关系中持久化类的实现方

法以及相应 Hibernate 映射文件的配置方案，并记录下来；

（3）查找相关资料，总结单向一对多/多对一关联关系中持久化类的实现方法以

及相应 Hibernate 映射文件的配置方案，并记录下来；

（4）根据实验步骤 20-22，查找相关资料，总结 Hibernate 映射文件的 set 元素

中 inverse、lazy、cascade 等属性的取值及作用，以及 many-to-one 元素中

的 not-null、fetch 等属性的作用，并记录下来；

（5）碰到的问题及解决方案或思考；

（6）实验收获及总结。

2、上交程序源代码，代码中应有相关注释。

**二、提高实验——多对多关联**

（一）实验目的

1、进一步掌握 Hibernate 关联关系映射的基本概念，理解关联的方向和数量，重

点理解双向多对多的关联关系，及其在实际应用中的体现；

2、学习 Hibernate 框架处理多对多关联关系的方法，掌握关联关系中持久化类的

实现方法、以及相应 Hibernate 映射文件的配置方案；

3、能在实际应用中通过 Hibernate 建立正确的多对多关联关系映射，并以面向对

象的方式进行数据库访问。

99（二）基本知识与原理

1、多对多关系在现实中也很常见，例如学生与选修课之间的关系，一个学生可

以选择多门选修课，而每个选修课又可以被多名学生选择；

2、数据库中的多对多关联关系一般需采用连接表的方式处理，将多对多关系转

化为两个一对多关系；若在实体类 A 和 B 之间表示双向多对多关系，则需要

A 类和 B 类中各关联一个集合对象，集合元素为对方的实例；

3、在 Hibernate 映射文件中，两个多端都需要使用<set.../>或<bag.../>元素来映射

关联属性，并在其 table 属性中指定连接表的名字。

（三）实验内容及步骤

1、在 hibernatedb 数据库中创建一个名称为 address2 的数据表，用于记录用户的

联系地址，customer 与 address2 是多对多的关系，创建表的 DDL 如下：

CREATE TABLE `address2` (

`addressID` INTEGER(11) NOT NULL,

`detail` VARCHAR(200) DEFAULT NULL,

`zipcode` VARCHAR(10) DEFAULT NULL,

`phone` VARCHAR(20) DEFAULT NULL,

`type` VARCHAR(20) DEFAULT NULL,

PRIMARY KEY (`addressID`)

)

2、在 hibernatedb 数据库中创建一个名称为 joincust\_addr 的连接表，将 customer

与 address2 的多对多关系转化为两个一对多关系，其中 cust\_id 是外键，参考

customer 表的主键，addr\_id 也是外键，参考 address2 表的主键，创建表的

DDL 如下：

CREATE TABLE `joincust\_addr` (

`cust\_id` INTEGER(11) NOT NULL,

`addr\_id` INTEGER(11) NOT NULL,

PRIMARY KEY (`cust\_id`, addr\_id),

FOREIGN KEY (`cust\_id`) REFERENCES customer (`customerID`),

FOREIGN KEY (`addr\_id`) REFERENCES customeraddress2 (`addressID`)

)

3、修改 cn.edu.zjut.po 包中的 Address.java，由于 Customer 与 Address 是多对多

的关系，因此需要在 Address 中增加一个 Set 集合属性，用于记录它关联的一

系列 Customer 实体，代码片段如下：

100package cn.edu.zjut.po;

……

public class Address {

private int addressId;

private String detail;

private String zipcode;

private String phone;

private String type;

**private Set customers = new HashSet(0);**

//省略构造函数

//省略 getters/setters 方法

}

4、修改cn.edu.zjut.po包中的映射文件Customer.hbm.xml，使用<set.../>或<bag.../>

元素来映射关联属性，并在其 table 属性中指定连接表的名字，代码片段如下：

<hibernate-mapping>

<class name="cn.edu.zjut.po.Customer" table="customer"

catalog="hibernatedb">

<id name="customerId" type="int">

<column name="customerID" />

<generator class="increment" />

</id>

……

**<set name="addresses" table="cust\_addrjoin" cascade="all">**

**<key column="cust\_id"/>**

**<many-to-many column="addr\_id"**

**class="cn.edu.zjut.po.Address"/>**

**</set>**

</class>

</hibernate-mapping>

5、修改 cn.edu.zjut.po 包中的 Address.hbm.xml，使得持久化类 Address 与数据库

表 address2 相映射，并使用<set.../>或<bag.../>元素来映射关联属性，同时在

其 table 属性中指定连接表的名字（代码略）；

6、将 hibernate-prj3 重新部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页

面，并记录运行结果；

7、修改 loginSuccess.jsp 页面，在该视图中增加“删除用户地址”的表单，代码

片段如下（字体加粗部分）：

<table>

<tr><td>个人信息：<p></td></tr>

<tr>

<td>用户名：</td>

<td><s:property value="#request.loginUser.account" /></td>

</tr>

101……

<s:iterator value="#request.loginUser.addresses" status="st">

**<s:form action="UserdelAddr" method="post">**

**<s:hidden name="loginUser.customerId"**

**value="%{#request.loginUser.customerId}"/>**

**<s:hidden name="address.addressId" value="%{addressId}"/>**

<tr><td>地址<s:property value="#st.count"/>：</td><tr>

<tr><td>详细地址：</td><td><s:property value="detail" /></td><tr>

……

**<tr><td><s:submit value="删除"/></td><tr>**

**</s:form>**

</s:iterator>

</table>

8、修改 cn.edu.zjut.service 包中的 UserService.java，在其中添加用户删除地址的

逻辑，**注意是删除用户与地址的关联，不是删除地址本身**（代码略）；

9、修改 cn.edu.zjut.action 包中的 UserAction.java，在其中添加地址删除的方法（代

码略）；

10、修改 struts.xml 文件，添加删除用户地址的页面导航（代码略）；

11、将 hibernate-prj3 重新部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页

面，并记录运行结果；

12、修改映射文件 Customer.hbm.xml 和 Address.hbm.xml 中<set>元素的 inverse

属性、lazy 属性和 cascade 属性的值，观察并记录运行结果。

（四）实验要求

1、填写并上交实验报告，报告中应包括：

（1）运行结果截图；

（2）根据实验步骤 3-5，总结双向多对多关联关系中持久化类的实现方法以及相

应 Hibernate 映射文件的配置方案，并记录下来；

（3）查找相关资料，总结单向多对多关联关系中持久化类的实现方法以及相应

Hibernate 映射文件的配置方案，并记录下来；

（4）根据实验过程，查找相关资料，总结 Hibernate 映射文件的<set>元素中

inverse、lazy、cascade 等属性的取值及作用，以及这些属性在多对多关联

关系和一对多/多对一关联关系中取值方法的异同，并记录下来；

（5）碰到的问题及解决方案或思考；

（6）实验收获及总结。

2、上交程序源代码，代码中应有相关注释。

102**三、扩展实验——一对一关联**

（一）实验目的

1、进一步掌握 Hibernate 关联关系映射的基本概念，理解关联的方向和数量，重

点理解双向一对一的关联关系，及其在实际应用中的体现；

2、学习 Hibernate 框架处理一对一关联关系的方法，掌握关联关系中持久化类的

实现方法、以及相应 Hibernate 映射文件的配置方案；

3、能在实际应用中通过 Hibernate 建立正确的一对一关联关系映射，并以面向对

象的方式进行数据库访问。

（二）基本知识与原理

1、在现实中的一对一关系，例如学生与身份证之间的关系，一个学生只有一份

身份证信息，而一个身份证也只能对应一名学生；

2、Hibernate 中的一对一关联分为主键关联和外键关联两类：主键关联是通过主

表和辅表的主键相关联，即两个表的主键值是一样的，不必增加额外的字段；

而外键关联方式里，主表和辅表通过外键相关联，可以看成是一对多/多对一

关联关系的特殊形式，多方退化成一方；

3、若采用外键关联，则在 Hibernate 映射文件中，一方使用<one-to-one>元素来

映射关联属性，并在其中用 property-ref 指定关联类的属性名，而多方退化成

一方只需要在<many-to-one>元素中设置"unique"="true"；若采用主键关联，

则在 Hibernate 映射文件中，两方都使用<one-to-one>元素来映射关联属性，

并将辅表的<id>元素（主键）的<generator>（生成策略）取值改成“foreign”，

同时加上属性参数。

（三）实验内容及步骤

1、修改 hibernatedb 数据库里的 customer 表与 address 表，使它们形成是一对一

外键关联的关系，即一个用户只能与一个地址形成唯一的对应关系；

2、修改 cn.edu.zjut.po 包中的 Customer.java 和 Address.java，由于本例中 Customer

与 Address 是一对一的关系，因此 Customer 中包括一个 Address 类型的属性，

Address 中包括一个 Customer 类型的属性，代码片段如下：

/\*\* Customer \*\*/

package cn.edu.zjut.po;

103……

public class Customer {

private int customerId;

private String account;

private String password;

private String repassword;

private String name;

private Boolean sex;

private String sexStr;

private Date birthday;

private String email;

**private Address address;**

//省略构造函数

//省略 getters/setters 方法

}

/\*\* Address \*\*/

package cn.edu.zjut.po;

……

public class Address {

private int addressId;

private String detail;

private String zipcode;

private String phone;

private String type;

**private Customer customer;**

//省略构造函数

//省略 getters/setters 方法

}

3、修改 cn.edu.zjut.po 包中的映射文件 Customer.hbm.xml 和 Address.hbm.xml，

使 它 们 形 成 一 对 一 外 键 关 联 的 关 系 ： 在 Customer.hbm.xml 中 使 用

<one-to-one.../>元素映射关联属性，并在其中用 property-ref="customer"表示

对应类的保存本类的属性名；而作为一对多/多对一关联关系的特例，在

Addres.hbm.xml 中仍使用<many-to-one.../>元素来映射关联属性，然后用属性

unique="true"做限制，代码片段如下：

/\*\* Customer.hbm.xml \*\*/

<hibernate-mapping>

<class name="cn.edu.zjut.po.Customer" table="customer"

catalog="hibernatedb">

<id name="customerId" type="int">

<column name="customerID" />

<generator class="increment" />

104105

</id>

……

**<one-to-one name="address" class="cn.edu.zjut.po.Address"**

**cascade="all" property-ref="customer"/>**

</class>

</hibernate-mapping>

/\*\* Address.hbm.xml \*\*/

<hibernate-mapping>

<class name="cn.edu.zjut.po.Address" table="address"

catalog="hibernatedb">

<id name="addressId" type="int">

<column name="addressID" />

<generator class="increment" />

</id>

……

**<many-to-one name="customer" class="cn.edu.zjut.po.Customer"**

**fetch="select" unique="true">**

**<column name="cust\_id"/>**

**</many-to-one>**

</class>

</hibernate-mapping>

4、修改 loginSuccess.jsp 页面，使得在该视图中，若用户已有一个地址信息，则

将其显示出来，并能选择删除；若用户没有地址信息，则显示“添加新地址”

的表单，代码片段如下：

<table>

<tr><td>个人信息：<p></td></tr>

<tr>

<td>用户名：</td>

<td><s:property value="#request.loginUser.account" /></td>

</tr>

……

**<s:if test="#request.loginUser.address">**

<s:form action="UserdelAddr" method="post">

<s:hidden name="loginUser.customerId"

value="%{#request.loginUser.customerId}"/>

<tr>

<td>详细地址：</td>

<td><s:property

value="#request.loginUser.address.detail" /></td>

<tr>

……

<tr><td><s:submit value="删除

"/></td><tr></s:form>

**</s:if>**

**<s:else>**

添加新地址：<p>

<s:form action="UseraddAddr" method="post">

<s:hidden name="loginUser.customerId"

value="%{#request.loginUser.customerId}"/>

<s:textfield name="address.detail" label="详细地址" />

……

<s:submit value="添加"/>

</s:form>

**</s:else>**

</table>

5、对 cn.edu.zjut.service 包 中 的 UserService.java 和 cn.edu.zjut.action 包 中 的

UserAction.java 做相应的修改，使得用户能添加或删除一个地址（代码略）；

6、将 hibernate-prj3 重新部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页

面，并记录运行结果；

7、尝试修改映射文件中<one-to-one.../>或<many-to-one.../>元素的 cascade 等属

性的值，观察并记录运行结果；

8、在 hibernatedb 数据库中创建一个名称为 address3 的数据表，用于记录用户的

联系地址，customer 与 address3 是一对一主键关联的关系，创建表的 DDL 如

下：

CREATE TABLE `address3` (

`addressID` INTEGER(11) NOT NULL,

`detail` VARCHAR(200) DEFAULT NULL,

`zipcode` VARCHAR(10) DEFAULT NULL,

`phone` VARCHAR(20) DEFAULT NULL,

`type` VARCHAR(20) DEFAULT NULL,

PRIMARY KEY (`addressID`),

)

9、维持 cn.edu.zjut.po 包中的 Customer.java 和 Address.java 代码不变，修改映射

文件 Customer.hbm.xml 和 Address.hbm.xml，使它们形成一对一主键关联的关

系：双方都使用<one-to-one.../>元素映射关联属性，在辅表 Addres.hbm.xml

的<one-to-one.../>元素中添加属性 constrained="true"表示受到外键约束，并将

其主键的<generator>设置成 foreign，同时增加属性参数，代码片段如下：

/\*\* Customer.hbm.xml \*\*/

<hibernate-mapping>

<class name="cn.edu.zjut.po.Customer" table="customer"

catalog="hibernatedb">

<id name="customerId" type="int">

106<column name="customerID" />

<generator class="increment" />

</id>

……

**<one-to-one name="address" class="cn.edu.zjut.po.Address"**

**cascade="all"/>**

</class>

</hibernate-mapping>

/\*\* Address.hbm.xml \*\*/

<hibernate-mapping>

<class name="cn.edu.zjut.po.Address" table="address3"

catalog="hibernatedb">

<id name="addressId" type="int">

<column name="addressID" />

**<generator class="foreign">**

**<param name="property">customer</param>**

**</generator>**

</id>

……

**<one-to-one name="customer" class="cn.edu.zjut.po.Customer"**

**cascade="all" constrained="true"/>**

</class>

</hibernate-mapping>

10、将 hibernate-prj3 重新部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页

面，并记录运行结果；

11、尝试修改映射文件中<one-to-one.../>元素的 cascade 等属性的值，观察并记录

运行结果。

（四）实验要求

1、填写并上交实验报告，报告中应包括：

（1）运行结果截图；

（2）根据实验过程，总结双向一对一关联关系中持久化类的实现方法，以及主

键关联和外键关联情况下 Hibernate 映射文件的配置方案，并记录下来；

（3）查找相关资料，总结单向一对一关联关系中持久化类的实现方法以及相应

Hibernate 映射文件的配置方案，并记录下来；

（4）根据实验过程，查找相关资料，总结一对一关联关系下，Hibernate 映射文

件的<one-to-one.../>元素 cascade 等属性的取值及作用，并记录下来；

（5）碰到的问题及解决方案或思考；

107（6）实验收获及总结。

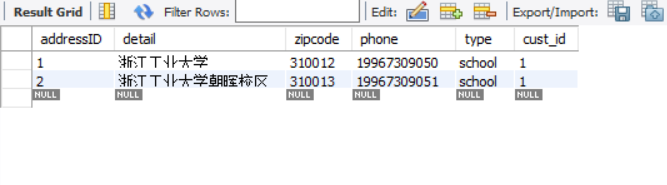
2、上交程序源代码，代码中应有相关注释。

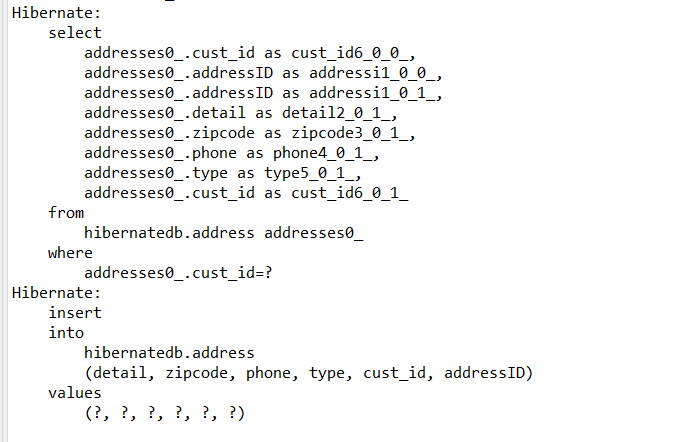
**基础实验——多对一/一对多关联**

1. **运行结果截图；**

**1.第19小题**

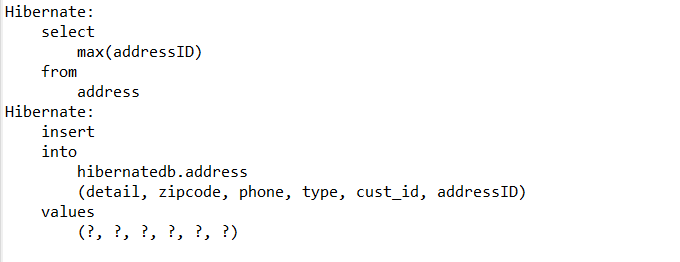




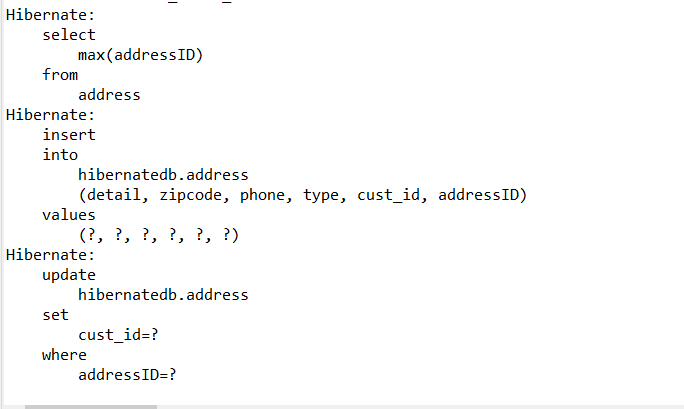


第20小题

（1）inverse值设为true



（2）inverse值设为false



第21小题

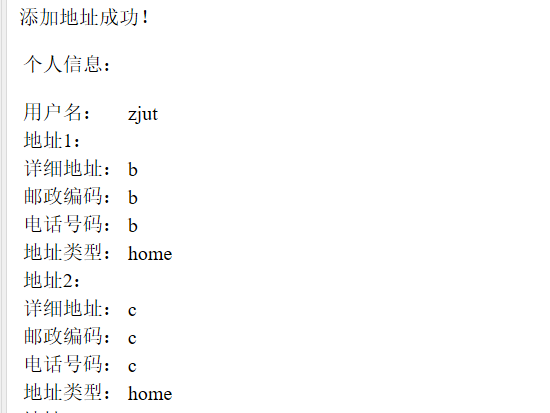
1. 在访问loginSuccess时报错

查询信息，发现lazy设置为true时即使用了延迟加载，延迟加载的含义指的是：当两个及以上表使用hibernate来管理级联关系时，一个表被加载到jvm内存时，其相关表可以暂存在数据库的缓存中，当需要使用相关表数据时在加载到jvm内存中。在访问子表时，若将session关闭，则断开了与数据库的连接，此时如果要访问子表数据，由于子表数据是存放在数据库的缓存中，而连接已经关闭，则访问子表数据会抛出LazyInitializationException异常。而显然此实验中就是这样的错误。

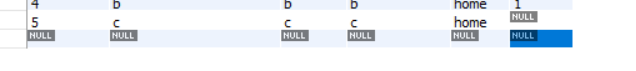


第二十二小题

（1）添加地址信息成功



（2）但数据库中的cust\_id为null



**（2）根据实验步骤 6-8，总结双向一对多/多对一关联关系中持久化类的实现方**

**法以及相应 Hibernate 映射文件的配置方案，并记录下来；**

双向一对多/多对一关联关系

Customer.java中属性的配置：

private Set addresses = new HashSet(0);

Address.java中属性的配置：

private Customer customer;

Customer.hbm.xml中的配置：

<set name="addresses" inverse="true" cascade="all" lazy="false">

<key column="cust\_id"/>

<one-to-many class="cn.edu.zjut.po.Address"/>

</set>

Address.hbm.xml中的配置：

<many-to-one name="customer" class="cn.edu.zjut.po.Customer"

fetch="select" not-null="true">

<column name="cust\_id"/>

</many-to-one>

**（3）查找相关资料，总结单向一对多/多对一关联关系中持久化类的实现方法以**

**及相应 Hibernate 映射文件的配置方案，并记录下来；**

单向一对多/多对一关联关系

只需在多的持久化类里加入如以下的配置

Address.java中属性的配置：

private Customer customer;

在Address.hbm.xml中

映射多对一的关联关系。 使用 many-to-one 来映射多对一的关联关系

name: 多这一端关联的一那一端的属性的名字

class: 一那一端的属性对应的类名

column: 一那一端在多的一端对应的数据表中的外键的名字

<many-to-one name="customer" class="Customer" column="CUSTOMER\_ID"></many-to-one>

**（4）根据实验步骤 20-22，查找相关资料，总结 Hibernate 映射文件的 set 元素中 inverse、lazy、cascade 等属性的取值及作用，以及 many-to-one 元素中**

**的 not-null、fetch 等属性的作用，并记录下来；**

inverse 有两个值 true ,false ;如果设置为true 则表示当前对象不负责讲级联对象的状态变化同步到数据库 ;设置false则相反,其默认值为false;

cascade 有五个选项,分别是: all ,delete ,none,save-update,delete-orphan ;

all : 所有情况下均进行关联操作。

none：所有情况下均不进行关联操作。这是默认值。

save-update:在执行save/update/saveOrUpdate时进行关联操作。

delete：在执行delete时进行关联操作。

delete-orphan: 当save/update/saveOrUpdate时，相当于save-update ；当删除操作时，相当于delete ；

lazy有三个属性：true、false、extratrue:默认取值，它的意思是只有在调用这个集合获取里面的元素对象时，才发出查询语句，加载其集合元素的数据

false:取消懒加载特性，即在加载对象的同时，就发出第二条查询语句加载其关联集合的数据

extra:一种比较聪明的懒加载策略，即调用集合的size/contains等方法的时候，hibernate并不会去加载整个集合的数据，而是发出一条聪明的SQL语句，以便获得需要的值，只有在真正需要用到这些集合元素对象数据的时候，才去发出查询语句加载所有对象的数据。

**（5）碰到的问题及解决方案或思考；**

1.CustomerDAO和AddressDAO的配置遇到困难

解决方案：根据Service中的方法来书写CustomerDAO中的方法，利用CustomerDAO中调用AddressDAO中的方法

2.思考：Customer.hbm.xml中set的lazy设置为true时，虽然懒加载机制可以减少内存中不必要的开销，以提高程序的性能，但懒加载也有他相应的缺点，如果将session关闭，则断开了与数据库的连接，此时如果要访问子表数据，由于子表数据是存放在数据库的缓存中，而连接已经关闭，则访问子表数据会抛出LazyInitializationException异常。

**（6）实验收获及总结。**

1.了解并实践了单向与多向一对多/多对一关联关系中持久化类的实现方法以

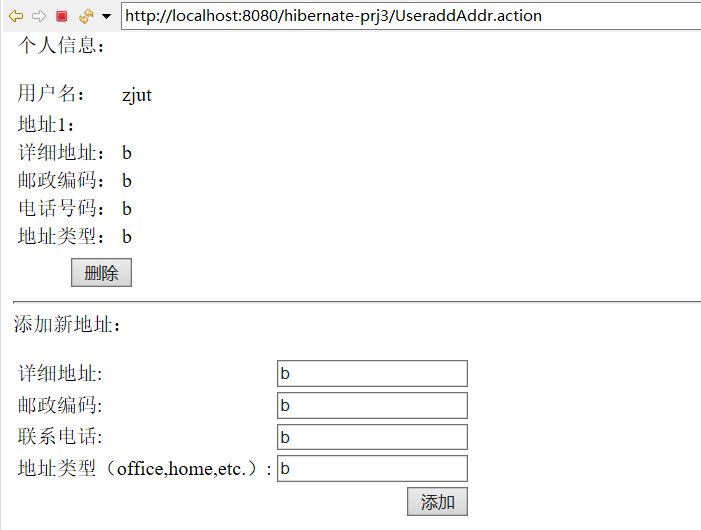
及相应 Hibernate 映射文件的配置方案。

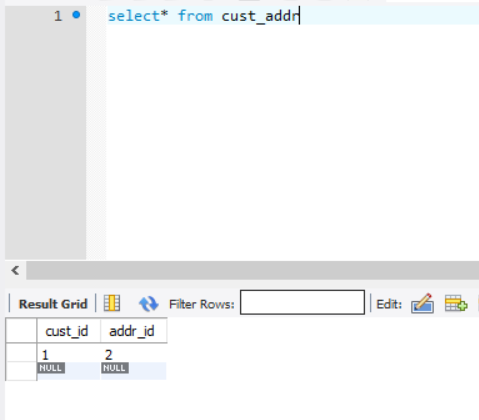
1. 熟悉了hbm.xml文件中相关属性的作用。

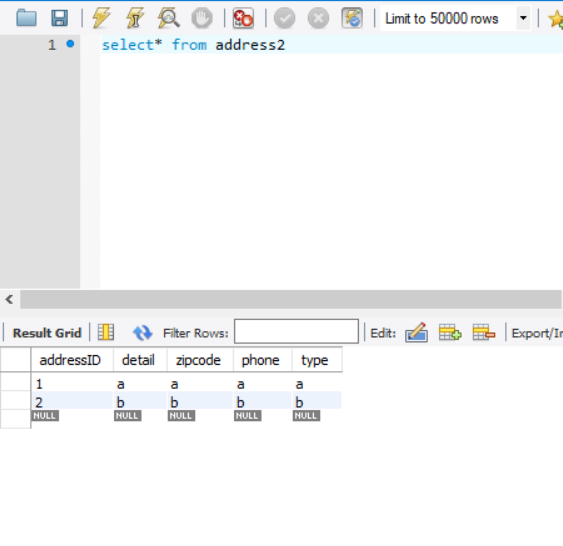
**二、提高实验——多对多关联**

**（1）运行结果截图；**

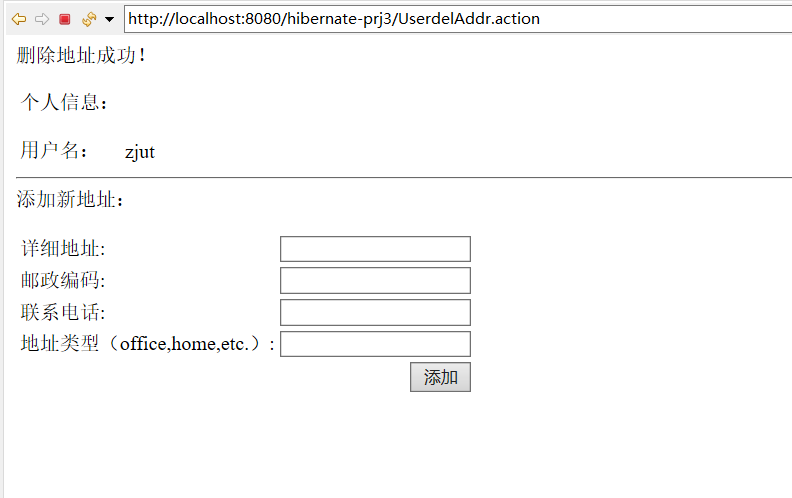
1删除前：

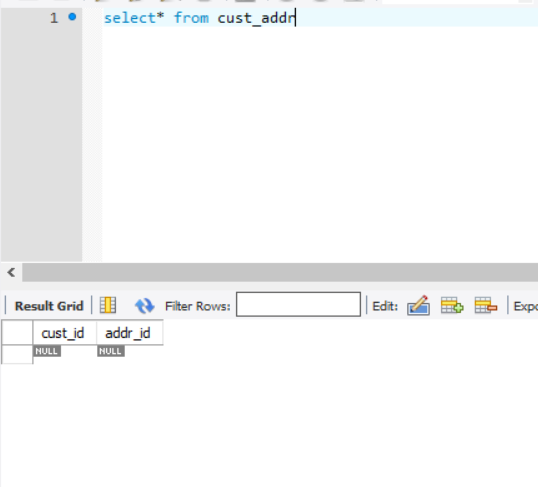


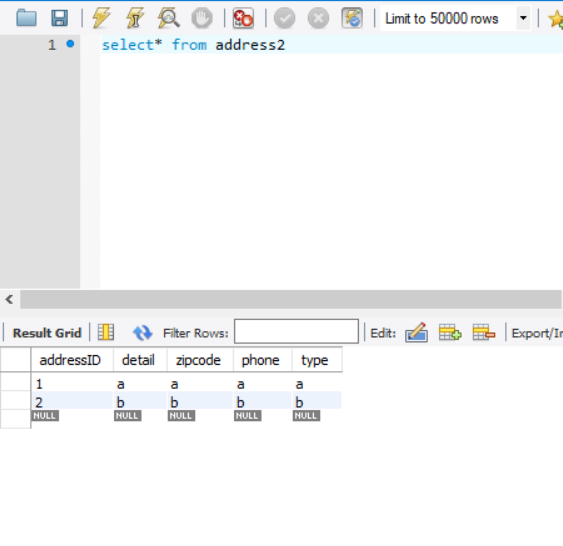




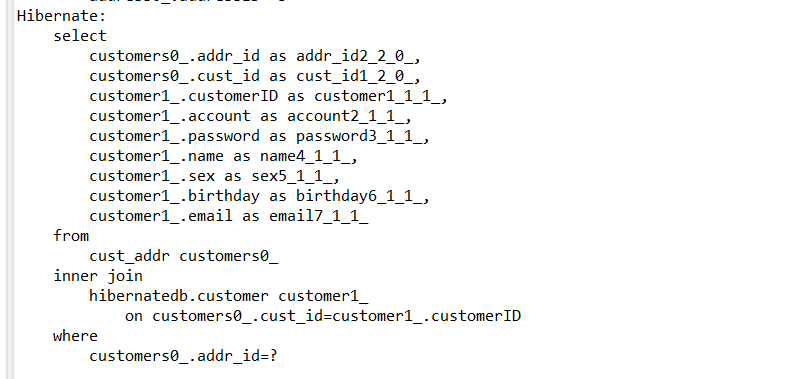
2删除后：







1. inverse设置为true

4.lazy设置为true(报错)



**（2）根据实验步骤 3-5，总结双向多对多关联关系中持久化类的实现方法以及相应 Hibernate 映射文件的配置方案，并记录下来；**

1.Customer.java

private Set addresses = new HashSet(0);

2.Address.java

private Set customers = new HashSet(0);

3.Customer.hbm.xml

<set name="addresses" table="cust\_addr" cascade="all" lazy="false" inverse="true">

<key column="cust\_id"/>

<many-to-many column="addr\_id" class="cn.edu.zjut.po.Address"/>

</set>

4.Address.hbm.xml

<set name="customers" table="cust\_addr" cascade="all" lazy="false">

<key column="addr\_id"/>

<many-to-many column="cust\_id" class="cn.edu.zjut.po.Customer"/>

</set>

**（3）查找相关资料，总结单向多对多关联关系中持久化类的实现方法以及相应**

**Hibernate 映射文件的配置方案，并记录下来；**

在持久化类中形如一下配置：

1.Customer.java

private Set addresses = new HashSet(0);

2.Address.java

private Set customers = new HashSet(0);

若是单向，只需在单向的hbm.xml文件中形如以下配置：

<set name="projects" table="PROEMP">

<!--column 多的一方表的外键列-->

<key column="REMPID"></key>

<!--column 中间表的另外一个外键列-->

<many-to-many column="RPROID" class="Project"></many-to-many>

</set>

**（4）根据实验过程，查找相关资料，总结 Hibernate 映射文件的<set>元素中**

**inverse、lazy、cascade 等属性的取值及作用，以及这些属性在多对多关联**

**关系和一对多/多对一关联关系中取值方法的异同，并记录下来；**

1. inverse：

“set的inverse属性决定是否把对set的改动反映到数据库中去。

inverse=false————反映；inverse=true————不反映”

inverse属性默认为false

对<one-to-many>和<many-to-many>子标记，这两条都适用。

当inverse=false时，hibernate如何将对set的改动反映到数据库中：

对set的操作主要有：（1）新增元素 address.getSchoolSet().add(oneSchool);

（2）删除元素 address.getSchoolSet().remove(oneSchool);

（3）删除set address.setSchoolSet(null);

（4）设新set address.setSchoolSet( newSchoolSet);

（5）转移set otherSchoolSet = otherAddress.getSchoolSet();

otherAddress.setSchoolSet(null);

address.setSchoolSet(otherSchoolSet);

改变set后，hibernate对数据库的操作根据是<one-to-many>关系还是<many-to-many>关系而有不同

1）对one-to-many而言，改变set，会让hibernate执行一系列的update语句， 不会delete/insert数据

（2）对many-to-many而言，改变set,只修改关系表的数据，不会影响many-to-many的另一方。

（3）虽然one-to-many和many-to-many的数据库操作不一样，但目的都是一个：维护数据的一致性。执行的sql都

只涉及到“桥字段”，不会考虑或改变其他的字段，所以对set的操作(6)是没有效果地。

extend:对list,可能还会维护index字段。

2.cascade

总结：级联（cascade）就是操作一个对象时，对它的属性（其cascade=true）也进行这个操作

inverse对集合对象整体起作用，cascade对集合对象中的一个一个元素起作用，如果集合为空，那么cascade不会引发关联操作

对cascade，一般对many-to-one，many-to-many，constrained=true的one-to-one 不设置级联删除

cascase 对set都是一致的，不管one-to-many还是many-to-many。都简单地把操作传递到set中的每个元素。所以它总是更新many

方的数据库表

3.Lazy

lazy可以在<class><one-to-one><many-to-one><set>元素中出现

<class>:lazy取值false/true,默认为true

<one-to-one>取值false/proxy/no-proxy,默认为proxy

<many-to-one>:lazy取值false/proxy/no-proxy,默认为proxy

<set>：lazy取值true/false/extra,默认为true

extra:可以发出智能的SQL语句,当获取集合对象的数量，发出count查询语句，当获取集合对象的属性值，发出select查询语句

**（5）碰到的问题及解决方案或思考；**

1.思考：多对多的联系，若是双向的。对删除方式的处理只需先remove删去的关系，再进行更新操作，即可完成只对关系的删除。若是想要删除其中一个对象（例如本例中的address），则可以在DAO中配置删除操作，再在service层中对需要删除的内容进行删除即可。

**（6）实验收获及总结。**

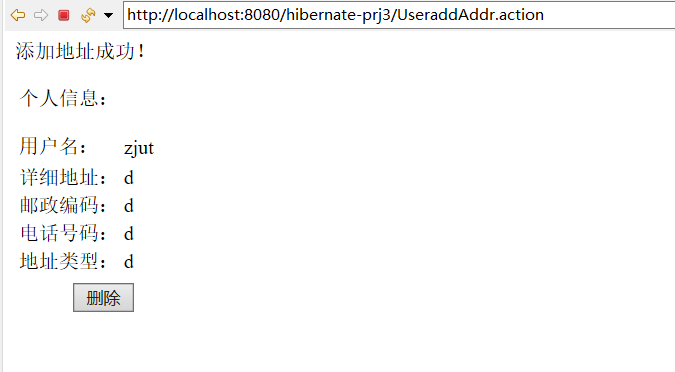
1.了解了多对多关系的处理方式以及相对应的set中属性的作用。

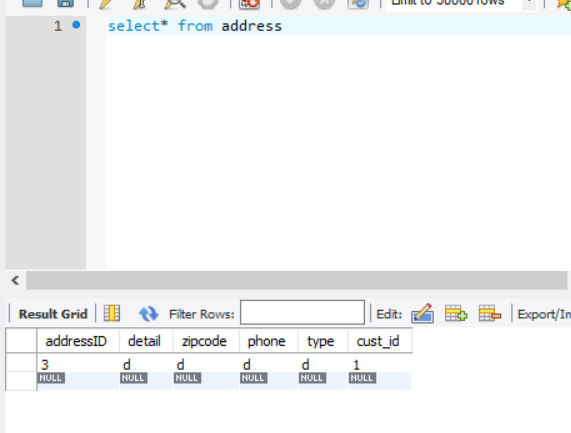
**三、扩展实验——一对一关联**

**（1）运行结果截图；**

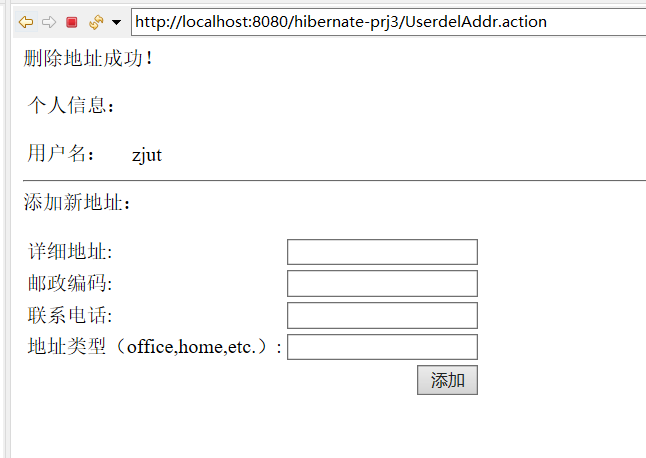
**1.第六小题**

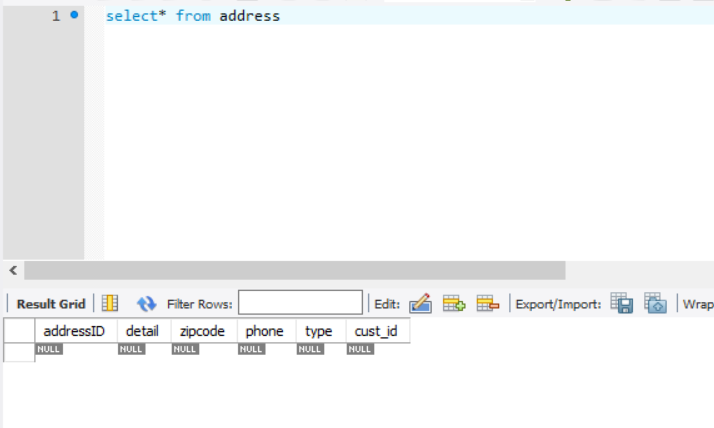
添加地址后：



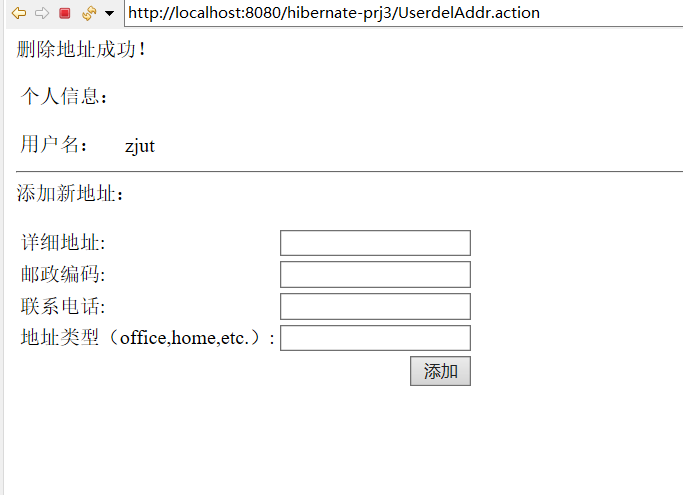


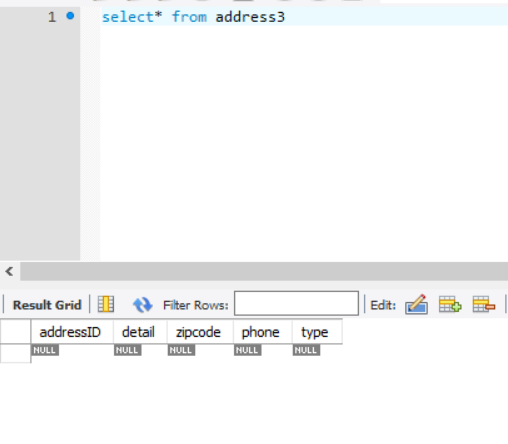
删除后：





**2.第十小题**





**（2）根据实验过程，总结双向一对一关联关系中持久化类的实现方法，以及主**

**键关联和外键关联情况下 Hibernate 映射文件的配置方案，并记录下来；**

配置相关的hbm.xml文件

Customer.hbm.xml

<one-to-one name="address" class="cn.edu.zjut.po.Address" cascade="all" property-ref="customer"/>

Address.hbm.xml

<many-to-one name="customer" class="cn.edu.zjut.po.Customer" fetch="select" unique="true"> <column name="cust\_id"/> </many-to-one>

**（3）查找相关资料，总结单向一对一关联关系中持久化类的实现方法以及相应**

**Hibernate 映射文件的配置方案，并记录下来；**

Customer.hbm.xml

<one-to-one name="address" class="cn.edu.zjut.po.Address" cascade="all"/>

Address.hbm.xml

<id name="addressId" type="int"> <column name="addressID" /> <generator class="foreign">

<param name="property">customer</param>

</generator>

</id>

**<one-to-one name="customer" class="cn.edu.zjut.po.Customer"**

**cascade="all" constrained="true"/>**

**（4）根据实验过程，查找相关资料，总结一对一关联关系下，Hibernate 映射文 件的<one-to-one.../>元素 cascade 等属性的取值及作用，并记录下来；**

name：映射类属性的名字

class：映射的目标类

cascade：设置操作中的级联策略 可选值为 all所有操作情况均进行级联、none所有操作情况均不进行级联、save-update执行更新操作时级联、delete执行删除操作时级联

constrained：表明当前类对应的表与被关联的表之间是否存在着外键约束 默认值为false

fetch：设置抓取数据的策略 可选值为 join外连接抓取、select序列选择抓取

property-ref：设置关联类的属性名，此属性和本类的主键相对应 默认值为关联类的主键

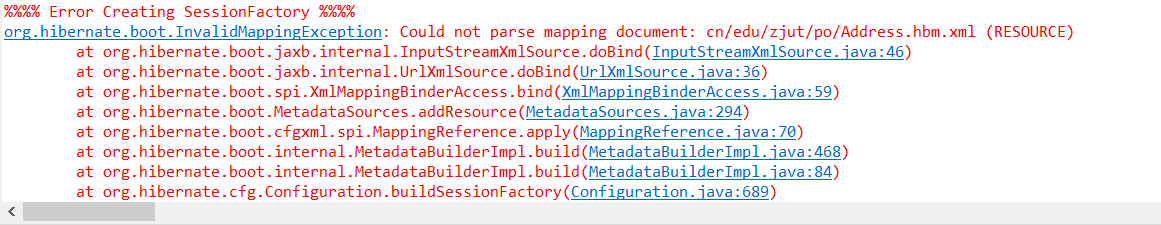
access：Hibernate访问这个属性的策略 默认值为property

formula：绝大多数一对一关联都指向其实体的主键。在某些情况下会指向一个或多个字段或是一个表达式，此时可用一个SQL公式来表示

lazy：指定是否采用延迟加载及加载策略 默认值为proxy通过代理进行关联 可选值为 true此对象采用延迟加载并在变量第一次被访问时抓取、false此关联对象不采用延迟加载

entity-name：被关联类的实体名（5）碰到的问题及解决方案或思考；

1.报了以下的错误。



解决方法：从Address.hbm.xml中找问题，发现多了一个/，删除后解决。

1. 报了很多次空指针问题，发现address取不到。

解决方法：address需要根据loginUser.getAddress()中取出。

**（6）实验收获及总结。**

1.熟悉并实践了一对一关联关系中持久化类的实现方法，以及主

键关联和外键关联情况下 Hibernate 映射文件的配置方案。

2.清楚了<one-to-one>中元素的取值及作用，做了很多次的尝试。