实验七 Hibernate 关联关系映射——登录用户的地址管理

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 7-1 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：

```
<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;
```

```
……
public 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();
        if (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；

20、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。

（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 cust_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 `cust_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 address2 (`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：

```
package 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_addr" 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>
……
<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。

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;
……
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" />
     </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">
        <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；

（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。