实验五 **Hibernate** 基础应用——基于 **Hibernate** 框架的用户

登录模块

一、基础实验——**Hibernate** 框架搭建

# （一）实验目的

1. 掌握 Hibernate 开发环境搭建的基本步骤；
2. 观察持久化类与数据库表的映射关系，观察相应的 Hibernate 映射文件

（.hbm.xml）配置，并能够做简单应用；

1. 观察 Hibernate 配置文件（hibernate.cfg.xml）中的主要元素及属性配置，并能够做简单应用。

# （二）基本知识与原理

1. Hibernate 是一个 ORM（Object-Relational Mapping）框架，用于把对象模型表示的对象映射到基于 SQL 的关系模型数据结构中去，采用完全面向对象的方式来操作数据库；
2. Hibernate 的主要作用是简化应用的数据持久层编程，不仅能管理 Java 类到数据库表的映射，还提供数据查询和获取数据的方法，从而大幅减少了开发人员编写 SQL 和 JDBC 代码的时间；
3. Hibernate 框架主要包括持久化对象（Persistent Objects）、Hibernate 配置文件（一般被命名为\*.cfg.xml）、Hibernate 映射文件（一般被命名为\*.hbm.xml）三部分；
4. 编译运行基于 Hibernate 框架的工程，需要导入相应的 Hibernate 类库；
5. 由于 Hibernate 底层是基于 JDBC 的，因此在应用程序中使用 Hibernate 执行持久化操作时也需要导入相关的 JDBC 驱动（例如 MySQL 数据库驱动）。

# （三）实验内容及步骤

1. 登录 [https://dev.mysql.com/downloads/connector/j/站](http://dev.mysql.com/downloads/mysql/)点，下载并安装 MySQL 数据库；
2. 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，表结构如表 5-1 所示：

表 5-1 customer 数据表

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 中文含义 |
| customerID | INTEGER(11), Primary key, Not Null | 用户编号 |
| Account | VARCHAR(20) | 登录用户名 |
| Password | VARCHAR(20) | 登录密码 |
| Name | VARCHAR(20) | 真实姓名 |
| Sex | TYPEINT(1) | 性别 |
| Birthday | DATE | 出生日期 |
| Phone | VARCHAR(20) | 联系电话 |
| Email | VARCHAR(100) | 电子邮箱 |
| Address | VARCHAR(200) | 联系地址 |
| Zipcode | VARCHAR(10) | 邮政编码 |
| Fax | VARCHAR(20) | 传真号码 |

1. 在表 customer 中添加 3 条记录，具体如表 5-2 所示：

表 5-2 customer 中的记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 用户编号 |  | 登录用户名 | 登录密码 |
| 1 |  | zjut |  | Zjut |
| 2 |  | admin |  | Admin |
| 3 |  | temp |  | Temp |

1. 登录 <http://downloads.mysql.com/archives/c-j/>站点，下载 MySQL JDBC 驱动；
2. 在 Eclipse 中新建 Web 工程 hibernate-prj1，并添加 MySQL 驱动程序库文件和 Struts2 核心包到工程中；
3. 登录 http://www.hibernate.org/downloads 站点，下载 Hibernate 发布版（如： hibernate-release-4.3.5.Final）并解压缩，将 Hibernate 发布版中 lib\required 里的 jar 包添加到工程 hibernate-prj1 中；
4. 在 <http://commons.apache.org/proper/commons-logging/download_logging.cgi> 站点，下载 [commons-logging-1.2-bin.zip](http://apache.fayea.com/apache-mirror//commons/logging/binaries/commons-logging-1.2-bin.zip) [并](http://apache.fayea.com/apache-mirror//commons/logging/binaries/commons-logging-1.2-bin.zip)解压缩，将 commons-logging-1.2.jar 添加到工程 hibernate-prj1 中；
5. 如图 5-1 所示，在工程 hibernate-prj1 中新建 Hibernate 相关文件，用于将对象模型表示的对象映射到关系模型数据结构中去，从而采用完全面向对象的方式来操作数据库：

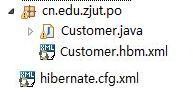


图 5-1 Hibernate 相关文件

1. 其中，cn.edu.zjut.po 包中的 Customer.java 是与 customer 数据库表相对应的持久化类，持久化类常用 POJO 编程模式实现，代码片段如下：

|  |
| --- |
| package cn.edu.zjut.po;  import java.util.Date; public class Customer implements java.io.Serializable {  private int customerId; private String account; private String password; private String name; private Boolean sex; private Date birthday; private String phone; private String email; private String address; private String zipcode; private String fax;  public Customer() {  }  public Customer(int customerId) { this.customerId = customerId;  }  public Customer(int customerId, String account, String password,  String name, Boolean sex, Date birthday, String phone, String email, String address, String zipcode, String fax) { this.customerId = customerId; this.account = account; this.password = password; this.name = name; this.sex = sex;  this.birthday = birthday; this.phone = phone; this.email = email; this.address = address; this.zipcode = zipcode; this.fax = fax; }  //省略getters/setters方法 } |

1. Customer.hbm.xml 是 Hibernate 映射文件，其中名为<class>的元素表示持久化类 Customer.java 与数据库表 customer 的映射关系，其子元素<id>表示持久化类中的主键，子元素<column>表示持久化类中的其它属性与数据库表中某个列的映射关系，具体代码如下：

|  |
| --- |
| <?xml version="1.0"?>  <!DOCTYPEhibernate-mappingPUBLIC"-//Hibernate/HibernateMappingDTD 3.0//EN"  "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  <hibernate-mapping>  **<class name="cn.edu.zjut.po.Customer" table="customer" catalog="hibernatedb">**  **<id name="customerId" type="int">**  **<column name="customerID" />**  **<generator class="assigned" />**  **</id>**  <property name="account" type="string">  <column name="account" length="20" unique="true" />  </property>  <property name="password" type="string">  <column name="password" length="20" />  </property>  <property name="name" type="string">  <column name="name" length="20" />  </property>  <property name="sex" type="java.lang.Boolean">  <column name="sex" />  </property>  <property name="birthday" type="date">  <column name="birthday" length="10" />  </property>  <property name="phone" type="string">  <column name="phone" length="20" />  </property>  <property name="email" type="string">  <column name="email" length="100" />  </property>  <property name="address" type="string">  <column name="address" length="200" />  </property>  <property name="zipcode" type="string">  <column name="zipcode" length="10" />  </property>  <property name="fax" type="string">  <column name="fax" length="20" />  </property>  </class>  </hibernate-mapping> |

1. hibernate.cfg.xml 是 Hibernate 配置文件，用于设置 JDBC 连接相关的属性，如连接数据库的地址、用户名、密码等，并在其中增加 Customer.hbm.xml 映射文件声明，具体代码如下：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <!DOCTYPE hibernate-configuration PUBLIC  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">  <hibernate-configuration>  <session-factory name="HibernateSessionFactory"> <property name="hibernate.connection.driver\_class"> com.mysql.jdbc.Driver</property>  <property name="hibernate.connection.url"> jdbc:mysql://localhost:3306/hibernatedb</property>  <property name="hibernate.connection.username"> root</property>  <property name="hibernate.connection.password"/>  <property name="hibernate.dialect"> org.hibernate.dialect.MySQLDialect</property>  **<mapping resource="cn/edu/zjut/po/Customer.hbm.xml" />** </session-factory>  </hibernate-configuration> |

1. 在 hibernate-prj1 中新建 login.jsp 页面，作为用户登录的视图；新建 loginSuccess.jsp 页面，作为登录成功的视图（可重用“实验二 Struts 基础应

用”中基础实验里的页面代码）；

1. 在 hibernate-prj1 中新建 cn.edu.zjut.dao 包，并在其中创建数据库操作类

CustomerDAO.java，具体代码如下：

|  |
| --- |
| package cn.edu.zjut.dao; import java.util.List; import org.hibernate.Query; import org.hibernate.SessionFactory; import org.hibernate.Session; import org.hibernate.cfg.Configuration; import org.apache.commons.logging.Log; import org.apache.commons.logging.LogFactory;  public class CustomerDAO { private Log log = LogFactory.getLog(CustomerDAO.class);  public List findByHql(String hql) { log.debug("finding LoginUser instance by hql");  SessionFactory sf= new Configuration()  .configure().buildSessionFactory(); |
| Session session=sf.openSession(); try {  String queryString = hql;  Query queryObject = session.createQuery(queryString); return queryObject.list();  } catch (RuntimeException re) { log.error("find by hql failed", re); throw re;  } finally{ session.close();  }  }  } |

1. 在hibernate-prj1中新建cn.edu.zjut.service包，并在其中创建UserService.java，用于实现登录逻辑，具体代码如下：

|  |
| --- |
| package cn.edu.zjut.service; import java.util.List; import cn.edu.zjut.po.Customer; import cn.edu.zjut.dao.CustomerDAO;  public class UserService { public boolean login(Customer loginUser) {  String account = loginUser.getAccount();  String password = loginUser.getPassword();  String hql = "from Customer as user where account='"  +account+ "' and password='" + password +"'";  CustomerDAO dao = new CustomerDAO(); List list = dao.findByHql(hql); if(list.isEmpty()) return false;  else return true;  }  } |

1. 在 hibernate-prj1 中新建 cn.edu.zjut.action 包，并在其中创建 UserAction.java，定义 login()方法用于调用登录逻辑，具体代码如下：

|  |
| --- |
| package cn.edu.zjut.action; import cn.edu.zjut.po.Customer; import cn.edu.zjut.service.UserService;  public class UserAction { private Customer loginUser; |
| public Customer getLoginUser() { return loginUser;  } public void setLoginUser(Customer loginUser) { this.loginUser = loginUser;  }  public String login() {  UserService userServ = new UserService(); if (userServ.login(loginUser)) { return "success";  } return "fail";  }  } |

1. 在工程 hibernate-prj1 的 src 目录中创建 struts.xml 文件，用于配置 Action 并设置页面导航，代码片段如下：

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <!DOCTYPE struts PUBLIC  "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"  "http://struts.apache.org/dtds/struts-2.3.dtd">  <struts>  <package name="strutsBean" extends="struts-default" namespace="/">  <action **name="login" class="cn.edu.zjut.action.UserAction"**>  <result **name="success"**>/loginSuccess.jsp</result>  <result **name="fail"**>/login.jsp</result> </action>  </package>  </struts> |

1. 编辑 Web 应用的 web.xml 文件，增加 Struts2 核心 Filter 的配置，代码片段如下：

|  |
| --- |
| <!-- 定义Struts2的核心Filter -->  <filter>  <filter-name>struts2</filter-name>  <filter-class> org.apache.struts2.dispatcher.ng.filter  .StrutsPrepareAndExecuteFilter </filter-class>  </filter>  <!-- 让Struts2的核心Filter拦截所有请求 -->  <filter-mapping>  <filter-name>struts2</filter-name>  <url-pattern>/\*</url-pattern> |

</filter-mapping>

1. 将 hibernate-prj1 部署在 Tomcat 服务器上；
2. 通过浏览器访问 login.jsp 页面，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. 观察工程 hibernate-prj1 中的 Hibernate 配置文件 hibernate.cfg.xml，查找相关资料，总结配置文件中各元素及其属性的作用，并记录下来；

1.首先<hibernate-configuration>，为文件的根元素，所有的相关配置都要在此元素下配置。

2.<session-factory>为session工厂配置，相当于JDBC中的DriverManager.

3.<property name="connection.username">root</property>,指定连接数据库的用户名。

4.<property name="connection.password"></property>，指定连接数据库的密码。

5.<property name="connection.driver\_class"> com.mysql.jdbc.Driver </property>，指定连接数据库的驱动程序。

6.<property name="connection.url"> jdbc:mysql://localhost:3306/mybbs </property>，指定连接数据库的URL.

7.<property name="dialect">org.hibernate.dialect.MySQLDialect</property>，指定数据库使用的SQL方言。

8.<property name="show\_sql">true</property>，显示SQL语句。

1. 观察工程 hibernate-prj1 中的 JAVA 持久化类 Customer.java、Hibernate 映射文件 Customer.hbm.xml，总结持久化类与数据库表的映射关系，以及映射文件中主要元素及其属性的作用，并记录下来；

1.<class>：用于配置一个实体类的配置信息。

name属性对应实体类的类名，table属性对应数据库的表名。

2.<id>：用于定义实体的OID（对应数据库的主键），在<class>节点内必须有一个<id>结点。

name属性对应实体类的类名，column属性用于指定数据库的字段

<generator>为<id>的子节点，用于指定主键的生成策略。

3.<property>：用于定义实体的普通属性（非主键）

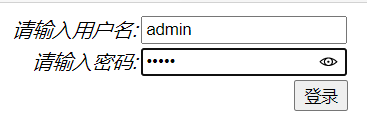
name属性对应实体类的属性，column属性对应数据库的字段。

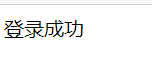
1. 根据实验过程，总结 Action、Service 和 DAO 之间的调用关系，思考实验一的扩展实验中的 DAO 类与本实验中 DAO 类的区别，并记录下来；

Action中的代码：这个类似于管理业务调度和管理跳转的，从这可以跳转到Service层，调用Service层中的方法，类似于三层结构中的B层，业务逻辑层。每个service代码中要有个ser方法，Service层是管理具体的功能，Action只负责管理，而Service负责实施。dao层相当于三层中的D层，这里连接数据库的操作用的是hibernate，完成增删改查。Hibernate的最大好处就是根据数据库的表，反向生成实体类，并且还有关系在里面，还有就是它对数据的操作也很方便；

1. 碰到的问题及解决方案或思考；
2. 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——持久化对象与 **Hibernate** 映射文件

# （一）实验目的

1. 进一步熟悉 Hibernate 应用的基本开发方法；
2. 掌握持久化类与持久化对象的概念，能按照规范进行持久化类的设计开发；
3. 掌握 Hibernate 映射文件的作用，熟悉映射文件中主要元素及其属性的含义和作用，并能进行正确应用；
4. 掌握 Hibernate 中主键的各种生成策略。

# （二）基本知识与原理

1. 在应用程序中，用来实现业务实体的类被称为持久化类（Persistent Class）如客户信息管理系统中的 Customer 类；
2. Hibernate 框架中的持久化类与数据库表对应，常用 POJO 编程模式实现，符合 JavaBean 规范，提供 public 的无参构造方法，提供符合命名规范的 getters

和 setters 方法；

1. 持久化类与数据库表对应，类的属性与表的字段对应；持久化类的对象被称为持久化对象 PO（Persistent Objects），PO 对应表中的一条记录；
2. 持久化对象映射数据库中的记录，其映射关系依赖 Hibernate 框架的映射文件配置，映射文件是 XML 文件，往往以\*.hbm.xml 形式命名，其中\*是持久化对象的类名；
3. Hibernate 映射文件中，元素<id>表示持久化类中的主键，<id>的子元素

<generator>表示主键的生成策略，其取值可以是“assigned”（用户赋值）、

“increment”（自动递增）等等；

1. 若数据库表中有多个列组成主键，则需要将其对应的持久化类中相应的多个属性封装成一个类，作为复合主键。

# （三）实验内容及步骤

1. 在 MySQL 的 hibernatedb 数据库中创建一个名称为 item 的数据表，表结构如表 5-3 所示：

表 5-3 item 数据表

|  |  |  |
| --- | --- | --- |
| 字段名称 | 类型 | 中文含义 |
| ISBN | VARCHAR(20), Primary key, Not Null | ISBN号 |
| title | VARCHAR(30) | 书名 |
| description | VARCHAR(100) | 说明 |
| cost | FLOAT | 单价 |
| image | BLOB | 图片 |

1. 在表 item 添加 2 条记录，具体如表 5-4 所示：表 5-4 itemr 中的记录

|  |  |  |  |
| --- | --- | --- | --- |
| **ISBN**号 | 书名 | 说明 | 单价 |
| 978-7-121-12345-1 | JAVAEE技术实验指导教程 | WEB程序设计知识回顾、轻量级JAVAEE  应用框架、企业级EJB组件编程技术、  JAVAEE综合应用开发 | 19.95 |
| 978-7-121-12345-2 | JAVAEE技术 | Struts框架、Hibernate框架、Spring框架、会话Bean、实体Bean、消息驱动Bean | 29.95 |

1. 在 hibernate-prj1 的 cn.edu.zjut.po 包中手动创建 JAVA 持久化类 Item.java，使其与 item 数据表相映射，代码片段如下：

|  |
| --- |
| package cn.edu.zjut.po; import java.sql.Blob; public class Item { private String itemID; private String title; private String description; private float cost;  private Blob image;  public Item() {  } public Item(String itemID) { this.itemID = itemID;  } public Item(String itemID, String title, String description, float cost, Blob image) {  this.itemID=itemID; this.title=title; this.description=description; this.cost=cost; this.image=image;  }  //省略setters/getters方法 } |

1. 在 Item.java 的同一目录下中手动创建 Hibernate 映射文件 Item.hbm.xml，并参照 Customer.hbm.xml（基础实验步骤 8）填写 Item.hbm.xml 中的内容；
2. 修改 hibernate-prj1 的 Hibernate 配置文件 hibernate.cfg.xml，增加 Item.hbm.xml 映射文件声明，代码片段如下：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <!DOCTYPE hibernate-configuration PUBLIC  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">  <hibernate-configuration>  <session-factory name="HibernateSessionFactory">  ……  <mapping resource="cn/edu/zjut/po/Customer.hbm.xml" />  **<mapping resource="cn/edu/zjut/po/Item.hbm.xml" />** </session-factory>  </hibernate-configuration> |

1. 在 hibernate-prj1 的 cn.edu.zjut.dao 包中创建数据库操作类 ItemDAO.java，具体代码如下：

package cn.edu.zjut.dao;

|  |
| --- |
| import java.util.List; import org.hibernate.Query; import org.hibernate.SessionFactory; import org.hibernate.Session; import org.hibernate.cfg.Configuration; import org.apache.commons.logging.Log; import org.apache.commons.logging.LogFactory;  public class ItemDAO { private static final Log log = LogFactory.getLog(ItemDAO.class);  public List findAll() { log.debug("finding all Item instances");  SessionFactory sf= new Configuration()  .configure().buildSessionFactory();  Session session=sf.openSession(); try {  String queryString = "from Item";  Query queryObject = session.createQuery(queryString); return queryObject.list();  } catch (RuntimeException re) { log.error("find all failed", re); throw re;  } finally{ session.close();  }  }  } |

1. 在 hibernate-prj1 的 cn.edu.zjut.service 包中创建 ItemService.java，用于获取所有商品信息，具体代码如下：

|  |
| --- |
| package cn.edu.zjut.service;  import java.util.List; import java.util.ArrayList; import cn.edu.zjut.dao.ItemDAO;  public class ItemService { private List items = new ArrayList();  public List getAllItems() {  ItemDAO dao = new ItemDAO(); List items = dao.findAll(); return items;  } |

}

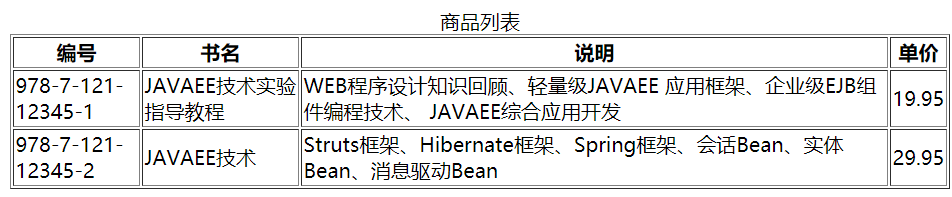
1. 在 hibernate-prj1 的 cn.edu.zjut.action 包中创建 ItemAction.java，并在其中定义 getAllItems()方法用于调用“获取所有商品信息”逻辑，参照 UserAction 类（基础实验步骤 16）填写 ItemAction 类的内容；
2. 在 hibernate-prj1 中新建 itemList.jsp 页面，作为显示所有商品信息的视图，代码片段如下：

|  |
| --- |
| <body>  <center>商品列表</center>  <table border=1>  <tr>  <th>编号</th>  <th>书名</th>  <th>说明</th>  <th>单价</th>  </tr>  <s:iterator value="items" >  <tr>  <td><s:property value="itemID"/></td>  <td><s:property value="title"/></td>  <td><s:property value="description"/></td>  <td><s:property value="cost"/></td>  </tr>  </s:iterator>  </table>  </body> |

1. 修改 hibernate-prj1 的 loginSuccess.jsp 页面，在视图中增加超链接，用于查看所有商品信息，代码片段如下；

<a href="./allItems">查看所有商品信息</a>

1. 修改 hibernate-prj1 的 struts.xml 文件，增加 ItemAction 的配置并设置页面导航；
2. 将 hibernate-prj1 重新部署在 Tomcat 服务器上；
3. 通过浏览器访问 login.jsp 页面，并记录运行结果；



1. 假设使用持久化类 Item.java 中的 itemID 属性和 title 属性作为复合主键，则需要将这两个属性封装成一个类，代码片段如下：

|  |
| --- |
| package cn.edu.zjut.po; public class ItemPK implement Serializable{ **private String itemID; private String title;**  //省略setters/getters方法 } |

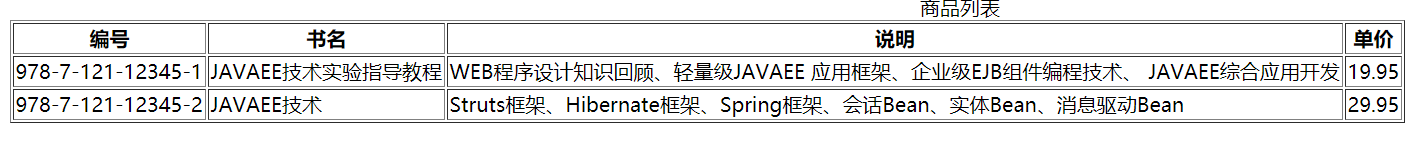
1. 修改 Item.java，将 ItemPK 主键作为其属性之一，代码片段如下：

|  |
| --- |
| package cn.edu.zjut.po;  ……  public class Item { **private ItemPK ipk;** private String description; private float cost;  private Blob image;  ……  } |

1. 修改 Hibernate 映射文件 Item.hbm.xml，将主键类中每个属性和表中的列对应，并指定复合主键的类型，代码片段如下：

|  |
| --- |
| <hibernate-mapping>  <class name="cn.edu.zjut.po.Item" table="item" >  **<composite-id name="ipk" class="cn.edu.zjut.po.ItemPK">**  **<key-property name="itemID" column="ISBN"/>**  **<key-property name="title" column="title"/> </composite-id>**  ……  </class>  </hibernate-mapping> |

1. 修改 itemList.jsp 页面，使其页面中能正确显示书本的编号与书名；
2. 将 hibernate-prj1 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。



# （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. 结合实验过程，查找相关资料，总结 POJO 模式下持久化类的规范，并记录下来；

## **1、提供一个无参的构造器**

## **2、提供一个标识属性**

**3、为持久化的类每个属性提供setter和getter方法**

## **4、使用非final类**

## **5、如果涉及到集合操作 还要重写equals（）和hashCode（）方法**

## **6、 持久化类中如果使用了集合类型数据，只能使用集合所对应的接口类型来声明(List/Map/Set)，而不能用它们的实现类**

1. 结合实验过程，查找相关资料，总结映射文件中主要元素（如 class、id、 generator、property）及其属性的含义与作用，并记录下来；

1.<class>：用于配置一个实体类的配置信息。

name属性对应实体类的类名，table属性对应数据库的表名。

2.<id>：用于定义实体的OID（对应数据库的主键），在<class>节点内必须有一个<id>结点。

name属性对应实体类的类名，column属性用于指定数据库的字段

<generator>为<id>的子节点，用于指定主键的生成策略。

3.<property>：用于定义实体的普通属性（非主键）

name属性对应实体类的属性，column属性对应数据库的字段。

1. 结合实验过程，查找相关资料，总结设置复合主键的方法和步骤，并记录下来；
2. 查找相关资料，总结 Hibernate 映射文件中主键各种生成策略的作用，并记录下来；

****increment****对主键值采取自动顺序增长的方式生成新的主键，值默认从1开始

****UUID****能够保证数据库中的主键唯一性，但是在生成的主键占用比较多的存贮空间

****Hilo****需要额外的数据库表和字段提供高位值来源。默认情况下使用的表是 　　hibernate\_unique\_key，默认字段叫作next\_hi。next\_hi必须有一条记录否则会出现错误。需要额外的数据库表的支持，能保证同一个数据库中主键的唯一性，但不能保证多个数据库之间主键的唯一性。Hilo主键生成方式由Hibernate 维护，所以Hilo方式与底层数据库无关。

****sequence****实际是就是一张单行单列的表。

****assigned****在程序中session.save();之前,由程序员自己指定主键值为多少。

****identity****根据底层数据库，来支持自动增长，不同的数据库用不同的主键增长方式

****foreign****只适用基于共享主键的一对一关联映射的时候使用。即一个对象的主键是参照的另一张表的主键生成的。

1. 碰到的问题及解决方案或思考；
2. 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——粒度设计

# （一）实验目的

1. 进一步熟悉持久化类与 Hibernate 映射文件的开发方法；
2. 学习在实际应用中进行粒度细分，将一张表映射到多个类。

# （二）基本知识与原理

1. 在实际应用中，并不都是一张表与一个实体类映射，往往可能会有一张表跟多个实体类映射的情况，称为粒度设计；
2. 如果表中的某些字段联合起来能表示持久化类中的某一个属性，那么可以进行基于设计的粒度设计：将表跟多个类映射；类和类之间使用关联关系；只需要一个映射文件，其中使用 component 元素进行映射；
3. 如果表中的某些字段不经常使用，而且占有空间较大，则可以使用基于性能的粒度设计：一个表可以映射为多个类；每个类对应一个\*.hbm.xml 文件；根据实际情况，使用不同的类。

# （三）实验内容及步骤

1. 在 hibernatedb 数据库里的 customer 数据表中，包括 phone、email、address、 zipcode、fax 在内的字段都是用户的联系方式，基于设计的粒度设计将用户的联系方式单独封装到类 ContactInfo，代码片段如下：

|  |
| --- |
| package cn.edu.zjut.po; public class ContactInfo {  private String phone; private String email; private String address; private String zipcode; private String fax;  public ContactInfo() { super();  }  public ContactInfo(String phone, String email, String address,  String zipcode, String fax) { |
| super(); this.phone = phone; this.email = email; this.address = address; this.zipcode = zipcode; this.fax = fax;  }  //省略setters/getters方法 } |

1. 修改 Customer.java，将 ContactInfo 实例作为 Customer 的属性，用于表示用户的联系方式，代码片段如下：

|  |
| --- |
| package cn.edu.zjut.po; public class Customer {  private int customerId; private String account; private String password; private String name; private Boolean sex; private Date birthday; private ContactInfo contactInfo;  //省略构造函数和setters/getters方法  } |

1. 修改 Hibernate 映射文件 Customer.hbm.xml，将 customer 表与两个类（Customer 和 ContactInfo）映射，具体代码如下：

|  |
| --- |
| <?xml version="1.0"?>  <!DOCTYPEhibernate-mappingPUBLIC"-//Hibernate/HibernateMappingDTD 3.0//EN"  "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  <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>  <property name="account" type="string">  <column name="account" length="20" unique="true" />  </property>  <property name="password" type="string">  <column name="password" length="20" />  </property> |
| <property name="name" type="string">  <column name="name" length="20" />  </property>  <property name="sex" type="java.lang.Boolean">  <column name="sex" />  </property>  <property name="birthday" type="date">  <column name="birthday" length="10" /> </property>  <componentname="contactInfo"class="cn.edu.zjut.po.ContactInfo">  <property name="phone" type="string">  <column name="phone" length="20" />  </property>  <property name="email" type="string">  <column name="email" length="100" />  </property>  <property name="address" type="string">  <column name="address" length="200" />  </property>  <property name="zipcode" type="string">  <column name="zipcode" length="10" />  </property>  <property name="fax" type="string">  <column name="fax" length="20" />  </property>  </component>  </class>  </hibernate-mapping> |

1. 修改 hibernate-prj1 的 Hibernate 配置文件 hibernate.cfg.xml，开启增删改操作的事务自动提交，代码片段如下：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <!DOCTYPE hibernate-configuration PUBLIC  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">  <hibernate-configuration>  <session-factory name="HibernateSessionFactory">  ……  **<property name="connection.autocommit">true</property>** </session-factory>  </hibernate-configuration> |

1. 在 hibernate-prj1 中新建 register.jsp 页面，作为用户注册的视图（可参考“实验二 Struts 基础应用”中提高实验里的页面代码，并根据数据库表 customer 设计页面内容）；新建 regSuccess.jsp 页面，作为注册成功的视图（可重用“实验二 Struts 基础应用”中提高实验里的页面代码）；
2. 修改 cn.edu.zjut.dao 包中的 CustomerDAO.java，增加“添加新用户”的操作，代码片段如下：

|  |
| --- |
| public class CustomerDAO {  ……  public void save(Customer customer) { log.debug("saving customer instance");  SessionFactory sf= new Configuration(). configure().buildSessionFactory();  Session session=sf.openSession(); try {  **session.save(customer);** session.flush(); log.debug("save successful");  } catch (RuntimeException re) { log.error("save failed", re); throw re;  } finally{ session.close();  }  }  } |

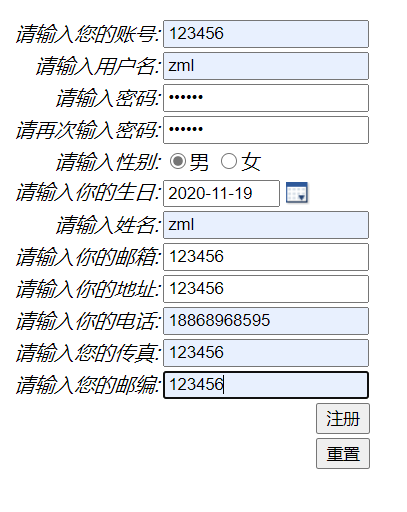
1. 修改 cn.edu.zjut.service 包中的 UserService.java，增加用户注册逻辑，代码片段如下：

|  |
| --- |
| public class UserService {  ……  public void register(Customer loginUser) { CustomerDAO dao = new CustomerDAO(); dao.save(loginUser);  }  } |

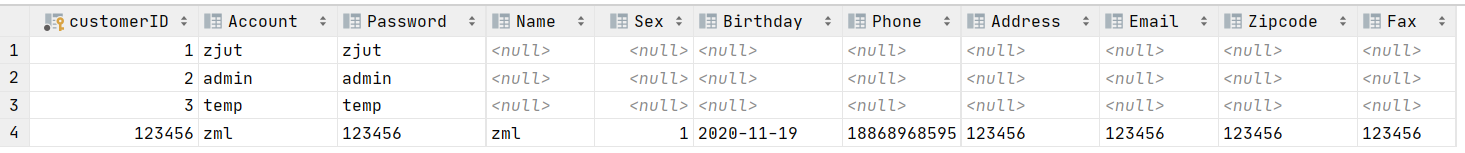
1. 修改 cn.edu.zjut.action 包中的 UserAction.java，定义 register()方法用于调用用户注册逻辑，代码片段如下：

|  |
| --- |
| public class UserAction {  ……  public String register() {  UserService userServ = new UserService(); userServ.register(loginUser); return "registersuccess";  }  } |

1. 修改工程 hibernate-prj1 中的 struts.xml 文件，为“用户注册”增加 Action 配置并设置页面导航；
2. 将 hibernate-prj1 重新部署在 Tomcat 服务器上；通过浏览器访问 register.jsp 页面，记录运行结果，并在数据库 customer 表中检查是否写入新的记录；







1. 在 hibernatedb 数据库里的 item 数据表中，image 字段表示商品的照片，使用 Blob 类型，所占空间较大，如果该字段不常被使用，则基于性能的粒度设计将表 item 映射为两个类，其中一个类为 ItemDetail，映射表中所有的字段，而另一个类 ItemBasic，映射表中除了 image 之外的字段，具体代码如下：

|  |
| --- |
| //ItemBasic  package cn.edu.zjut.po; public class ItemBasic { private String itemID; private String title; private String description; private Float cost;  public ItemBasic() {  } public ItemBasic(String itemID) { this.itemID = itemID;  } public ItemBasic(String itemID, String title, String description, float cost) {  this.itemID=itemID; this.title=title; this.description=description; this.cost=cost;  }  //省略setters/getters方法 } |
| //ItemDetail  package cn.edu.zjut.po; Import java.sql.Blob public class ItemDetail extends ItemBasic{ private Blob image;  public ItemDetail() {  } publicItemDetail(StringitemID,Stringtitle,Stringdescription, float cost, Blob image) {  super(itemID, title, description, cost); this.image=image; |
| }  //省略setters/getters方法 } |

1. 在 ItemBasic.java 和 ItemDetail.java 的同一目录下中手动创建 Hibernate 映射文件 ItemBasic.hbm.xml 和 ItemDetail.hbm.xml，ItemDetail.hbm.xml 的具体代码如下（ItemBasic.hbm.xml 略）：

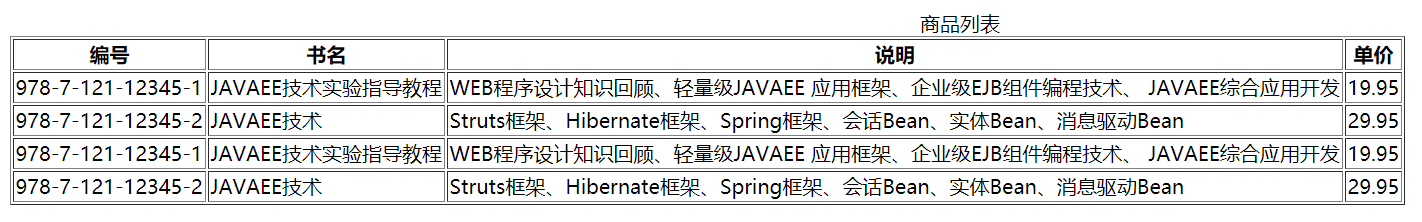
|  |
| --- |
| <?xml version="1.0"?>  <!DOCTYPEhibernate-mappingPUBLIC"-//Hibernate/HibernateMappingDTD 3.0//EN"  "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  <hibernate-mapping>  <class name="cn.edu.zjut.po.ItemDetail" table="item" catalog="hibernatedb">  <id name="itemID" type="string">  <column name="ISBN" length="20" />  <generator class="assigned" />  </id>  <property name="title" type="string">  <column name="title" length="30" />  </property>  <property name="description" type="string">  <column name="description" length="100" />  </property>  <property name="cost" type="Float">  <column name="cost" />  </property>  <property name="image" type="java.sql.Blob">  <column name="image" />  </property>  </class>  </hibernate-mapping> |

1. 修改hibernate-prj1的Hibernate配置文件hibernate.cfg.xml，删除Item.hbm.xml 映射文件声明，增加 ItemDetail.hbm.xml 和 ItemBasic.hbm.xml 映射文件声明，

代码片段如下：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <!DOCTYPE hibernate-configuration PUBLIC  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">  <hibernate-configuration>  <session-factory name="HibernateSessionFactory">  ……  <mapping resource="cn/edu/zjut/po/Customer.hbm.xml" />  **<mapping resource="cn/edu/zjut/po/ItemDetail.hbm.xml" />** |
| **<mapping resource="cn/edu/zjut/po/ItemBasic.hbm.xml" />** </session-factory>  </hibernate-configuration> |

1. 相应地修改 ItemDAO 将 hibernate-prj1 重新部署在 Tomcat 服务器上；
2. 通过浏览器访问 login.jsp 页面，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. 结合实验过程，总结两种粒度设计的方法及特点；
3. 根据实验步骤 4，查找相关资料，写出 Hibernate 配置文件 hibernate.cfg.xml 中的“connection.autocommit”属性的作用；

hibernate.connection.autocommit 允许被缓存的 JDBC 连接开启自动提交

1. 碰到的问题及解决方案或思考；
2. 实验收获及总结。



2、上交程序源代码，代码中应有相关注释。