浙江工业大学

JavaEE技术实验报告

实验名称：Hibernate 基础应用——基于 Hibernate 框架的用户登录模块

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10月28日

**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；

3、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/mysqlconnector/j/ 站 点 ， 下 载 并 安 装

MySQL 数据库；

5758

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 1-1 所示：

3、在表 customer 中添加 3 条记录，具体如表 1-2 所示：

4、登录 http://downloads.mysql.com/archives/c-j/站点，下载 MySQL JDBC 驱动；

5、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；

6、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；

7、在 http://commons.apache.org/proper/commons-logging/download\_logging.cgi 站

点，下载 commons-logging-1.2-bin.zip 并解压缩，将 commons-logging-1.2.jar

添加到工程 hibernate-prj1 中；

8、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59

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

}

（2）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"?>

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD

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>

60（3）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>

9、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）；

10、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();

61Session 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();

}

}

}

11、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;

}

}

12、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;

62public 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";

}

}

13、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"**

**method="login"**>

<result **name="success"**>/loginSuccess.jsp</result>

<result **name="fail"**>/login.jsp</result>

</action>

</package>

</struts>

14、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>

63<url-pattern>/\*</url-pattern>

</filter-mapping>

15、将 hibernate-prj1 部署在 Tomcat 服务器上；

16、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；

（3）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；

（4）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；

（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——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

6465

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；

3、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；

4、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；

5、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；

6、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 1-1 所示：

1. 在表 item 添加 2 条记录，具体如表 1-2 所示：

3、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

}

4、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；

5、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>

6、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：

66package 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();

}

}

}

7、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();

67return items;

}

}

8、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；

9、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>

10、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>

11、修改 hibernate-prj1 的 struts.xml 文件，增加 ItemAction 的配置并设置页面导

航；

12、将 hibernate-prj1 重新部署在 Tomcat 服务器上；

13、通过浏览器访问 login.jsp 页面，并记录运行结果；

14、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

68}

15、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;

……

}

16、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>

17、修改 itemList.jsp 页面，使其页面中能正确显示书本的编号与书名；

18、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；

（3）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；

（4）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；

（5）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；

（6）碰到的问题及解决方案或思考；

69（7）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();

70}

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

}

2、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

}

3、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"?>

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD

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" />

71</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>

<component name="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>

4、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>

725、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）；

6、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();

}

}

}

7、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);

}

}

8、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";

73}

}

9、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；

10、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；

11、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 Ffloat 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() {

}

public ItemDetail(String itemID, String title, String description,

74float cost, Blob image) {

super(itemID, title, description, cost);

this.image=image;

}

//省略 setters/getters 方法

}

12、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"?>

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD

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="fFloat">

<column name="cost" />

</property>

<property name="image" type="java.sql.Blob">

<column name="image" />

</property>

</class>

</hibernate-mapping>

13、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">

75……

<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>

14、相应地修改 ItemDAO 将 hibernate-prj1 重新部署在 Tomcat 服务器上；

15、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；

（4）碰到的问题及解决方案或思考；

（5）实验收获及总结。

2、上交程序源代码，代码中应有相关注释。

**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；

3、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/mysqlconnector/j/ 站 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 1-1 所示：

3、在表 customer 中添加 3 条记录，具体如表 1-2 所示：

4、登录 http://downloads.mysql.com/archives/c-j/站点，下载 MySQL JDBC 驱动；

5、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；

6、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；

7、在 http://commons.apache.org/proper/commons-logging/download\_logging.cgi 站

点，下载 commons-logging-1.2-bin.zip 并解压缩，将 commons-logging-1.2.jar

添加到工程 hibernate-prj1 中；

8、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59

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

}

（2）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"?>

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD

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>

60（3）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>

9、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）；

10、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();

61Session 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();

}

}

}

11、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;

}

}

12、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;

62public 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";

}

}

13、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"**

**method="login"**>

<result **name="success"**>/loginSuccess.jsp</result>

<result **name="fail"**>/login.jsp</result>

</action>

</package>

</struts>

14、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>

63<url-pattern>/\*</url-pattern>

</filter-mapping>

15、将 hibernate-prj1 部署在 Tomcat 服务器上；

16、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；

（3）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；

（4）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；

（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——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

6465

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；

3、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；

4、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；

5、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；

6、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

2、在表 item 添加 2 条记录，具体如表 1-2 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

3、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

}

4、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；

5、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>

6、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：

66package 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();

}

}

}

7、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();

67return items;

}

}

8、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；

9、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>

10、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>

11、修改 hibernate-prj1 的 struts.xml 文件，增加 ItemAction 的配置并设置页面导

航；

12、将 hibernate-prj1 重新部署在 Tomcat 服务器上；

13、通过浏览器访问 login.jsp 页面，并记录运行结果；

14、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

68}

15、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;

……

}

16、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>

17、修改 itemList.jsp 页面，使其页面中能正确显示书本的编号与书名；

18、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；

（3）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；

（4）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；

（5）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；

（6）碰到的问题及解决方案或思考；

69（7）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();

70}

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

}

2、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

}

3、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"?>

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD

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" />

71</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>

<component name="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>

4、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>

725、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）；

6、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();

}

}

}

7、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);

}

}

8、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";

73}

}

9、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；

10、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；

11、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 Ffloat 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() {

}

public ItemDetail(String itemID, String title, String description,

74float cost, Blob image) {

super(itemID, title, description, cost);

this.image=image;

}

//省略 setters/getters 方法

}

12、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"?>

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD

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="fFloat">

<column name="cost" />

</property>

<property name="image" type="java.sql.Blob">

<column name="image" />

</property>

</class>

</hibernate-mapping>

13、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">

75……

<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>

14、相应地修改 ItemDAO 将 hibernate-prj1 重新部署在 Tomcat 服务器上；

15、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；

（4）碰到的问题及解决方案或思考；

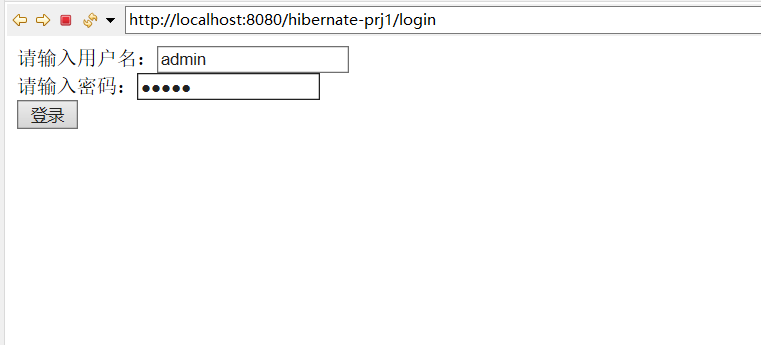
（5）实验收获及总结。

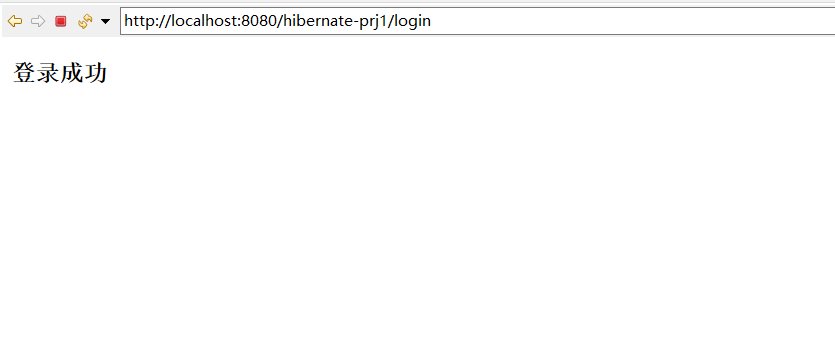
2、上交程序源代码，代码中应有相关注释。

**一、基础实验——Hibernate 框架搭建**

**（1）运行结果截图；**

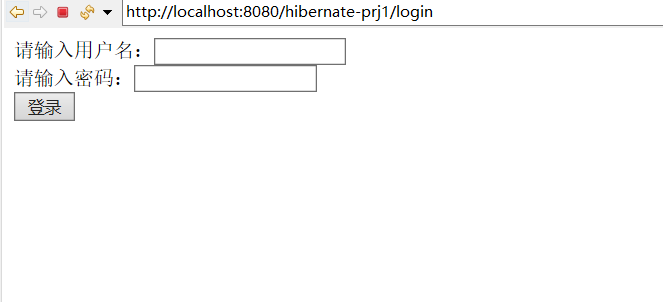
**1.登录成功**





1. **登录失败返回login.jsp**





1. **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；**
2. connection.url ：数据库 URL
3. connection.username ：数据库用户名
4. connection.password ：数据库用户密码
5. connection.driver\_class ：数据库 JDBC 驱动
6. dialect ：配置数据库的方言，根据底层的数据库不同产生不同的 sql 语句， Hibernate 会针对数据库的特性在访问时进行优化
7. **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；**
8. Hibernate映射文件由<hibernate-mapping package="JavaBean所在包的全路径">节点定义映射内容并指定所对应的JavaBean的位置(也可以不在该节点中用package属性指定对应的JavaBean位置,而在该节点下的class节点中的name属性中指定)
9. Hibernate映射文件中用<hibernate-mapping>节点下的

      <class name="JavaBean名称" table="对应数据库中的表名">节点指定数据库表和JavaBean的关联。(该节点的父节点中用package属性指定了JavaBean的包名时用)/全路径(该节点的父节点中没用package属性指定JavaBean的包名时用)

3.在<class>节点下用

     <id name="数据库中主键在JavaBean中的属性名称" column="数据库中主键名" type="string">节点映射对应的主键,该节点必须有且只有一个(因为主键只有一个),同时必须放在<property ...>节点前

4.在<class ...>节点下用

        <property name="数据库中字段在JavaBean中的属性名称" column="数据库中的字段名" type="string"/>节点映射普通字段,该节点可有多个(一个字段用一个该节点来映射)

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；**

1.Action，管理业务调度和页面跳转，可以跳转到service层中去调用service层中的方法，类似于业务逻辑层。

2.Service中管理具体的某种方法，相比于Action的只负责管理，Service层起到了实施的作用。

3.DAO层起到了连接数据库的作用。这里连接数据库的操作用的是hibernate。

DAO的区别：Hibernate 是一个面向Java环境的对象/关系数据库映射工具。用来bai把对象模型表示的对象映射到基于SQL 的关系模型结构中去，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，并且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. 第一次尝试hibernate连接数据库，在配置完所有文件之后发现连接不上数据库，根据报错信息去网上查找相关信息，第一步先仔细检查了hibernate.cfg.xml文件，发现没有配置错误，第二步下载了版本较低的mysql驱动器，成功运行。
3. 发现eclipse项目前有个红叉，调出problem页面发现是版本较高的mysql驱动器产生了冲突，对lib中的文件重新进行build path后解决。

**（6）实验收获及总结。**

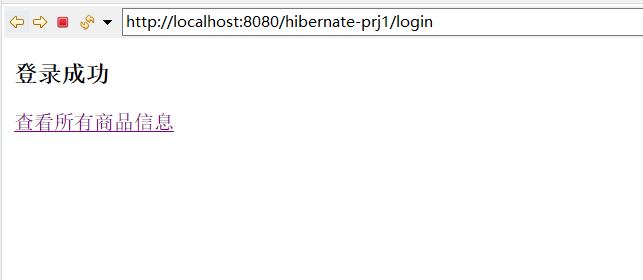
1.成功配置了hibernate并且用hibernate的方式进行了数据库的连接。

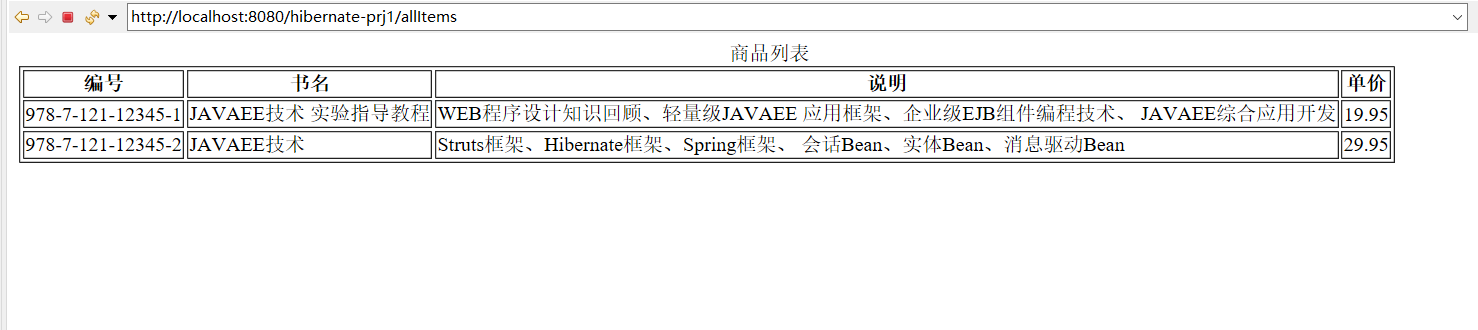
2，了解了hibernate的使用方法及与普通连接数据库的方式进行了对比。

**二、提高实验——持久化对象与 Hibernate 映射文件**

**（1）运行结果截图；**

**1.登录成功页面**

  
2 **.显示数据库中存在的书籍数据**



1. **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；**

七个规范：

1. 公有类
2. 共有无参构造
3. 私有属性
4. 公有的getter和setter方法
5. 实现java.io.Serializable接口
6. 不能用final修饰
7. 如果是基础类型，要使用他们的包装类

**（3）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；**

**class标签的一些属性：**

1.class 元素用于指定类和表的映射

2.name:指定该持久化类映射的持久化类的类名

3.table:指定该持久化类映射的表名, Hibernate 默认以持久化类的类名作为表名

4.dynamic-insert: 若设置为 true, 表示当保存一个对象时, 会动态生成 insert 语句, insert 语句中仅包含所有取值不为 null 的字段. 默认值为 false

5.dynamic-update: 若设置为 true, 表示当更新一个对象时, 会动态生成 update 语句, update 语句中仅包含所有取值需要更新的字段. 默认值为 false

6.select-before-update:设置 Hibernate 在更新某个持久化对象之前是否需要先执行一次查询. 默认值为 false

7.batch-size:指定根据 OID 来抓取实例时每批抓取的实例数.

lazy: 指定是否使用延迟加载.

8.mutable: 若设置为 true, 等价于所有的<property> 元素的 update 属性为 false, 表示整个实例不能被更新. 默认为 true.

9.discriminator-value: 指定区分不同子类的值. 当使用<subclass/>元素来定义持久化类的继承关系时需要使用该属性

**id标签：**

id：设定持久化类的 OID 和表的主键的映射

1.name: 标识持久化类 OID 的属性名

2.column: 设置标识属性所映射的数据表的列名(主键字段的名字).

3.unsaved-value:若设定了该属性, Hibernate 会通过比较持久化类的 OID 值和该属性值来区分当前持久化类的对象是否为临时对象

4.type:指定 Hibernate 映射类型. Hibernate 映射类型是 Java 类型与 SQL 类型的桥梁. 如果没有为某个属性显式设定映射类型, Hibernate 会运用反射机制先识别出持久化类的特定属性的 Java 类型, 然后自动使用与之对应的默认的 Hibernate 映射类型

5.Java 的基本数据类型和包装类型对应相同的 Hibernate 映射类型. 基本数据类型无法表达 null, 所以对于持久化类的 OID 推荐使用包装类型

**generator标签**

设定持久化类设定标识符生成器

class: 指定使用的标识符生成器全限定类名或其缩写名

**property标签**

property 元素用于指定类的属性和表的字段的映射

1.name:指定该持久化类的属性的名字

2.column:指定与类的属性映射的表的字段名. 如果没有设置该属性, Hibernate 将直接使用类的属性名作为字段名.

3.type:指定 Hibernate 映射类型. Hibernate 映射类型是 Java 类型与 SQL 类型的桥梁. 如果没有为某个属性显式设定映射类型, Hibernate 会运用反射机制先识别出持久化类的特定属性的 Java 类型, 然后自动使用与之对应的默认的 Hibernate 映射类型.

4.not-null:若该属性值为 true, 表明不允许为 null, 默认为 false

5.access:指定 Hibernate 的默认的属性访问策略。默认值为 property, 即使用 getter, setter 方法来访问属性. 若指定 field, 则 Hibernate 会忽略 getter/setter 方法, 而通过反射访问成员变量

6.unique: 设置是否为该属性所映射的数据列添加唯一约束.

7.update:设置是否允许更改该字段

8.index: 指定一个字符串的索引名称. 当系统需要 Hibernate 自动建表时, 用于为该属性所映射的数据列创建索引, 从而加快该数据列的查询.

9.length: 指定该属性所映射数据列的字段的长度

10.scale: 指定该属性所映射数据列的小数位数, 对 double, float, decimal 等类型的数据列有效.

11.formula：设置一个 SQL 表达式, Hibernate 将根据它来计算出派生属性的值.

12.派生属性: 并不是持久化类的所有属性都直接和表的字段匹配, 持久化类的有些属性的值必须在运行时通过计算才能得出来, 这种属性称为派生属性

**（4）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；**

1.改写 PO类

若表中有多个列组成主键，则为这几个列封装成一个类，作为主键，并增加setter和getter方法

2.编写po类，将主键对象作为属性之一

3.在映射文件中进行配置

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

**（5）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；**

1.increment：适用于代理主键。由Hibernate自动以递增的方式生成。

2.identity：适用于代理主键。由底层数据库生成标识符。

3.sequence：适用于代理主键。Hibernate根据底层数据库的序列生成标识符，这要求底层数据库支持

4.hilo：适用于代理主键。Hibernate分局high/low算法生成标识符。

5.seqhilo：适用于代理主键。使用一个高/低位算法来高效的生成long，short或int类型的标识符。

6.native：适用于代理主键。根据底层数据库对自动生成标识符的方式，自动选择identity，sequence或hilo

7.uuid.hex适用于代理主键。Hibernate采用128位的UUID算法生成标识符。

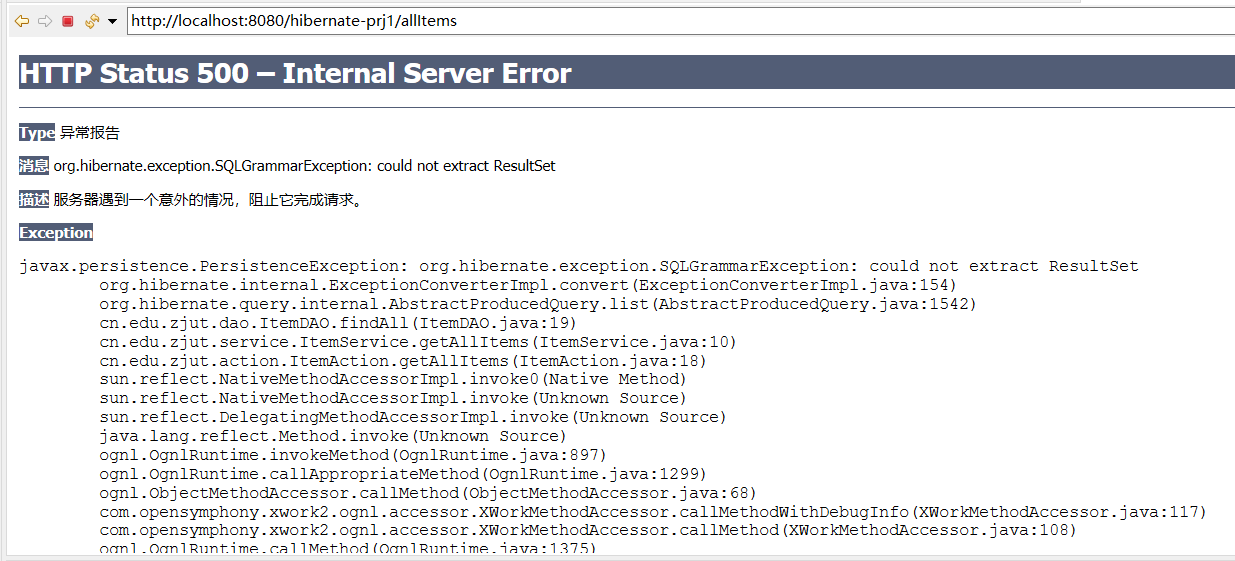
9.uuid.string：适用于代理主键。UUID被编码成一个16位字符长的字符串。

10.assigned：适用于自然主键，由java应用程序负责生成标识符。

11.foreign：适用于代理主键。使用另外一个相关联的对象的标识符。

1. **碰到的问题及解决方案或思考；**
2. item.hbm.xml文件配置出现问题，导致了数据库连接不上.

报错信息（部分）:



解决方法：查询相关信息，判断哪里出现了问题，仔细核对item.hbm.xml文件，找出其中的书写错误，改正之后解决。

2.思考：对hbm.xml文件的书写有了更好的理解，对hibernate连接数据库的方式也理解的更深了。

**（7）实验收获及总结。**

1.对持久化类与数据库的对应关系应该如何在hbm.xml文件中书写有了更好的认识和理解

2.对复合主键有了一定的认识和理解。

**三、扩展实验——粒度设计**

**（1）运行结果截图；**

**1.第10小题**

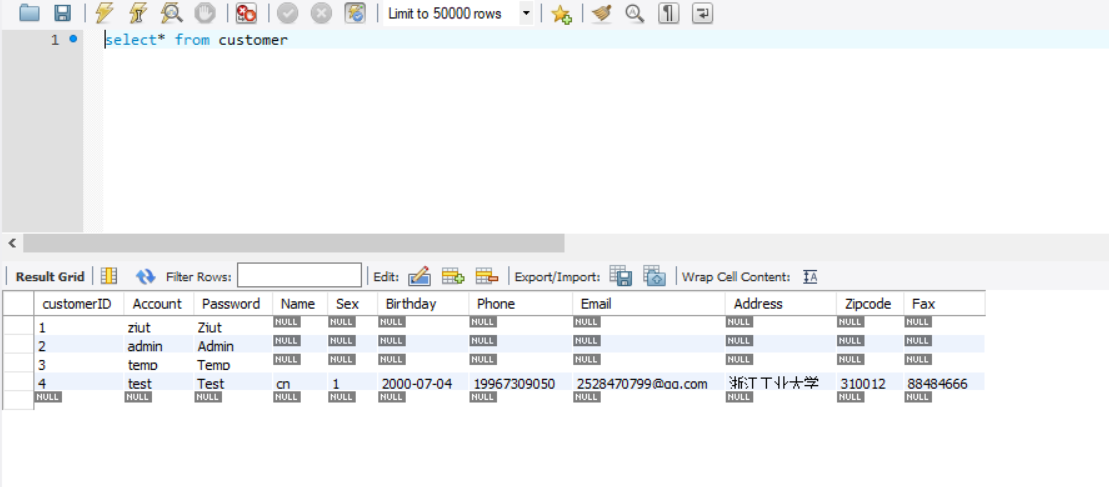
**注册界面：**



**注册成功界面：**

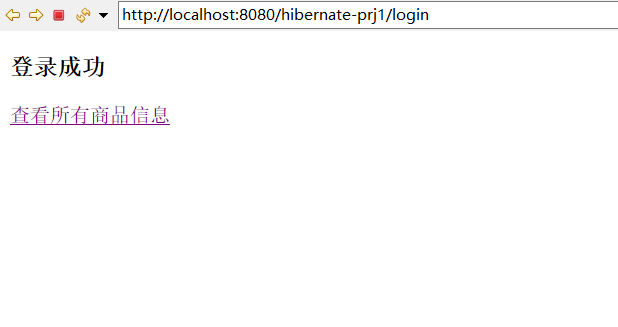


**数据库查询界面：**

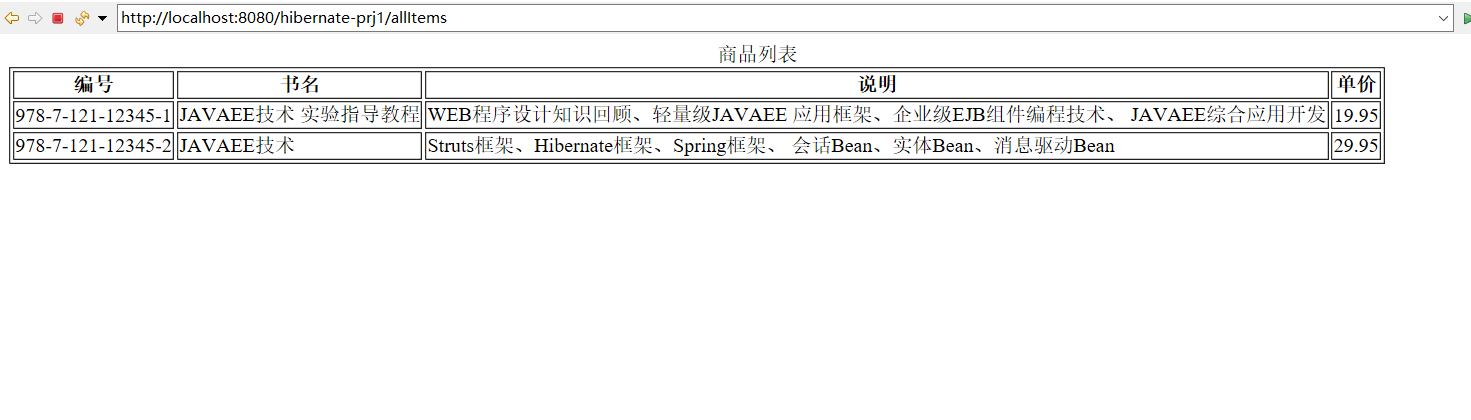


1. **第15小题**

**登录成功界面：**



**商品信息界面：**



1. **结合实验过程，总结两种粒度设计的方法及特点；**
2. 将表中几个属性单独封装到其中某一个类中，在hbm.xml文件中用<component>将其体现。

特点：通过对象细化，实现更清晰的系统逻辑划分。

2.将不常用的、读取代价较高的属性，例如blob类型，可以将原先的类拆分成两个不同的类。

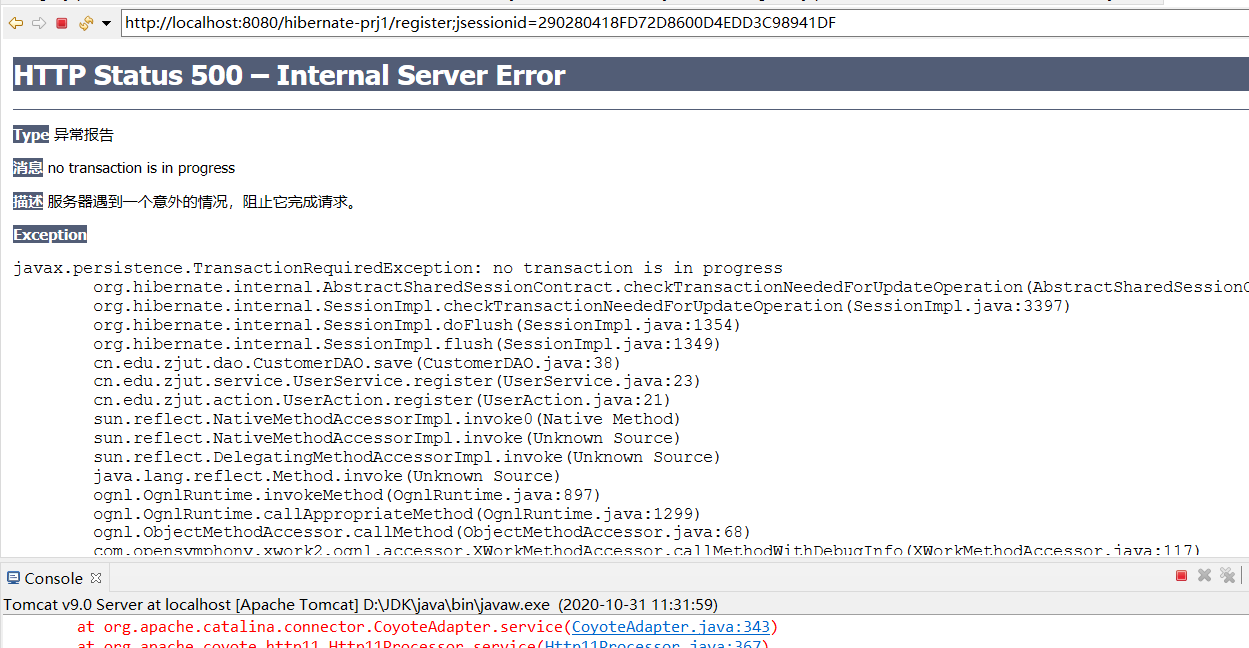
特点：针对业务逻辑，通过合理的细粒度对象，提高系统的能耗比。

1. **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，及缓存中的数据可以自动与数据库进行连接操作。（无需事务的提交）

**（5）碰到的问题及解决方案或思考；**

1.session.flush（）报错，说没有事务



检查了很多次，网上也找了很多资料，但是最后发现是register.jsp中写错了，没有将.contactInfo加上，加上后就能正确保存到数据库中。

1. 思考：写代码时还是需要理解他的代码结构和运作方式，了解每个函数的作用才能够继续往下做，不然盲目的根据报错信息找错误是很难真正写好实验的。
2. 思考：在遇到困难时，可以将真正的sql语句显示在工作台上，有利于更好的检验哪里出错。

**（6）实验收获及总结。**

1.了解并实践了粒度设计。

2.对hibernate的实现形式有了更好的理解。