浙江工业大学

JavaEE技术实验报告

实验名称：Hibernate 的体系结构——登录用户信息的增删改查

学 院： 计算机学院

班 级： 软件工程1801

姓 名： 陈南

学 号： 201806062101

组 号： 无

时 间： 2020年11月4日

**一、基础实验——Hibernate 常用 API**

（一）实验目的

1、进一步掌握 Hibernate 应用的开发方法，理解 Hibernate 配置文件中主要元素

的作用，会开发持久化类，并进行相应的 Hibernate 映射文件配置；

2、学习并掌握 Hibernate 框架的常用 API，掌握利用 Hibernate 基本 API 载入配

置文件、建立数据库连接的基本步骤；

3、理解 Hibernate 基本 API 中 Session 的主要作用，掌握利用 Session 进行数据

库操作的基本步骤。

（二）基本知识与原理

1、Hibernate 进行持久化操作，通常有如下操作步骤：

（1）开发持久化类和映射文件；

（2）获取 Configuration；

（3）获取 SessionFactory；

（4）获取 Session，打开事务；

（5）用面向对象的方式操作数据库；

（6）关闭事务，关闭 Session。

2、对 PO 的操作必须在 Session 管理下才能同步到数据库，Session 是应用程序

与持久储存层之间交互操作的一个单线程对象，它底层封装了 JDBC 连接，

主要用来对 PO 进行创建、读取、删除等操作；

3、Session 由 SessionFactory 工厂产生，SessionFactory 是数据库编译后的内存镜

像，通常一个应用对应一个 SessionFactory 对象；

4、SessionFactory 对象由 Configuration 对象生成，Configuration 对象负责加载

Hibernate 配置文件，每个 Hibernate 配置文件对应一个 Configuration 对象。

（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

个名称为 customer 的数据表，表结构如表 1-1 所示：

2、在 Eclipse 中新建 Web 工程 hibernate-prj2，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；

3、在 hibernate-prj2/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；

4、在 hibernate-prj2 中 新 建 cn.edu.zjut.po 包 ， 并 在 其 中 创 建 持 久 化 类

Customer.java以及Hibernate映射文件Customer.hbm.xml，具体代码可参照“实

验五 Hibernate 基础应用”中基础实验里的内容；

5、修改配置文件 hibernate.cfg.xml，增加 Customer.hbm.xml 映射文件的声明；

6、在 hibernate-prj2 中新建 cn.edu.zjut.dao 包，并在其中创建 DAO 操作辅助类

HibernateUtil.java，用于生成 Configuration 对象和 SessionFactory 对象，具体

代码如下：

package cn.edu.zjut.dao;

import org.hibernate.HibernateException;

import org.hibernate.Session;

import org.hibernate.cfg.Configuration;

public class HibernateUtil {

private static String CONFIG\_FILE\_LOCATION = "/hibernate.cfg.xml";

private static final ThreadLocal<Session>

threadLocal = new ThreadLocal<Session>();private static Configuration configuration = new Configuration();

private static org.hibernate.SessionFactory sessionFactory;

private static String configFile = CONFIG\_FILE\_LOCATION;

static {

try {

**configuration.configure(configFile);**

**sessionFactory = configuration.buildSessionFactory();**

} catch (Exception e) {

System.err

.println("%%%% Error Creating SessionFactory %%%%");

e.printStackTrace();

}

}

private HibernateUtil() { }

public static org.hibernate.SessionFactory getSessionFactory() {

return sessionFactory;

}

public static void rebuildSessionFactory() {

try {

configuration.configure(configFile);

sessionFactory = configuration.buildSessionFactory();

} catch (Exception e) {

System.err

.println("%%%% Error Creating SessionFactory %%%%");

e.printStackTrace();

}

}

public static void setConfigFile(String configFile) {

HibernateUtil.configFile = configFile;

sessionFactory = null;

}

public static Configuration getConfiguration() {

return configuration;

}

}

7、完善辅助类 HibernateUtil.java，在其中添加代码用于“获取 Session 对象”和

“关闭 Session 对象”，代码片段如下：

package cn.edu.zjut.dao;

public class HibernateUtil {

……

**public static Session getSession() throws HibernateException** {

79Session session = (Session) threadLocal.get();

if (session == null || !session.isOpen()) {

if (sessionFactory == null) {

rebuildSessionFactory();

}

session = (sessionFactory != null)

? sessionFactory.openSession(): null;

threadLocal.set(session);

}

return session;

}

**public static void closeSession() throws HibernateException** {

Session session = (Session) threadLocal.get();

threadLocal.set(null);

if (session != null) {

session.close();

}

}

}

8、在 hibernate-prj2 的 cn.edu.zjut.dao 包中创建 BaseHibernateDAO.java，作为数

据库操作基础类，具体代码如下：

package cn.edu.zjut.dao;

import org.hibernate.Session;

public class BaseHibernateDAO{

public Session getSession() {

return HibernateUtil.getSession();

}

}

9、在 hibernate-prj2 的 cn.edu.zjut.dao 包中创建 CustomerDAO.java，继承数据库

操作基础类 BaseHibernateDAO.java，实现 Customer 增删改查操作，代码片

段如下：

package cn.edu.zjut.dao;

import java.util.List;

import org.hibernate.Query;

import org.apache.commons.logging.Log;

import org.apache.commons.logging.LogFactory;

import cn.edu.zjut.po.Customer;

public class CustomerDAO extends BaseHibernateDAO{

private Log log = LogFactory.getLog(CustomerDAO.class);

public List findByHql(String hql) {

log.debug("finding Customer instance by hql");

try {

80String queryString = hql;

Query queryObject = getSession().createQuery(queryString);

return queryObject.list();

} catch (RuntimeException re) {

log.error("find by hql failed", re);

throw re;

}

}

public void save(Customer instance) {

log.debug("saving Customer instance");

try {

getSession().save(instance);

log.debug("save successful");

} catch (RuntimeException re) {

log.error("save failed", re);

throw re;

}

}

public void update(Customer instance) { //省略代码 }

public void delete(Customer instance) { //省略代码 }

}

10、在hibernate-prj2中新建cn.edu.zjut.service包，并在其中创建UserService.java，

用于实现登录、注册、个人信息修改和删除逻辑，代码片段如下：

package cn.edu.zjut.service;

import com.opensymphony.xwork2.ActionContext;

……

public class UserService {

private Map<String, Object> request, session;

public boolean login(Customer loginUser) {

ActionContext ctx= ActionContext.getContext();

session=(Map) ctx.getSession();

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

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);**

**dao.getSession().close();**

if (list.isEmpty())

return false;

else{

**session.put("user", account);**

81**request.put("tip", "登录成功！");**

loginUser=(Customer)list.get(0);

request.put("loginUser", loginUser);

return true;

}

}

public boolean register(Customer loginUser) { //省略代码 }

public boolean update(Customer loginUser) { //省略代码 }

public boolean delete(Customer loginUser) {

ActionContext ctx= ActionContext.getContext();

session=(Map) ctx.getSession();

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

**CustomerDAO dao = new CustomerDAO();**

**Transaction tran = null;**

try {

**tran = dao.getSession().beginTransaction();**

**dao.delete(loginUser);**

**tran.commit();**

session.remove("user");

request.put("tip", "删除个人信息成功，请重新登录！");

return true;

} catch (Exception e) {

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

return false;

} finally {

**dao.getSession().close();**

}

}

}

11、在 hibernate-prj2 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;

//省略 getters/setters 方法

public String login() {

UserService userServ = new UserService();

if (userServ.login(loginUser))

return "loginsuccess";

else

return "loginfail";

82}

public String register() { //省略代码 }

public String update() { //省略代码 }

public String delete() { //省略代码 }

}

12、在 hibernate-prj2 中新建 login.jsp 页面，作为用户登录视图，代码片段如下：

<body>

**<s:property value="#request.tip"/>**<p>

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

<s:textfield name="loginUser.account" label="请输入用户名"/>

<s:password name="loginUser.password" label="请输入密码"/>

<s:submit value="登录"/>

</s:form>

</body>

13、在 hibernate-prj2 中新建 loginSuccess.jsp 页面，作为登录成功视图，并能在

该页面中修改个人信息或删除个人信息，代码片段如下：

<body>

**<s:property value="#request.tip"/>**<p>

修改个人信息<p>

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

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

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

<s:textfield name="loginUser.account" label="用户名不能修改"

value="%{#request.loginUser.account}" readonly="true"/>

<s:textfield type="password" name="loginUser.password"

label="修改密码" value="%{#request.loginUser.password}"/>

<!-- 省略其它表单域 -->

<s:submit value="修改"/>

</s:form>

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

<s:hidden name="loginUser.customerId"

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

<s:submit value="删除"/>

</s:form>

</body>

14、在 hibernate-prj2 中新建 register.jsp 页面，作为用户注册视图（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）；

15、在 hibernate-prj2 中新建 registerSuccess.jsp 页面，作为注册成功的视图（代

码略）；

16、在 hibernate-prj2 中新建 CURDFail.jsp 页面，作为“修改个人信息和删除个

83人信息”操作失败的视图（代码略）；

17、在 hibernate-prj2 的 src 目录中创建 struts.xml 文件，用于配置 Action 并设置

页面导航，使得：登录成功时转向 loginSuccess.jsp 页面，失败时转向 login.jsp

重新登录；注册成功时转向 registerSuccess.jsp 页面，失败时转向 register.jsp

重新注册；修改个人信息成功时转向 loginSuccess.jsp 页面，失败时转向

CURDFail.jsp 页面；删除个人信息成功时转向 login.jsp 页面重新登录，失败

时转向 CURDFail.jsp 页面；

18、编辑 Web 应用的 web.xml 文件，增加 Struts2 核心 Filter 的配置；

19、将 hibernate-prj2 部署在 Tomcat 服务器上；

20、通过浏览器访问 register.jsp 页面，注册一个新用户（假设用户名各不相同），

然后用该新用户登录，并修改用户信息或删除用户，查看并记录运行结果。

（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，总结利用 Hibernate 基本 API 载入配置文

件、建立数据库连接并进行数据库操作的基本步骤；

（3）结合实验过程，查找相关资料，总结 Session 中的主要方法，及其与数据库

操作的对应关系；

（4）根据实验步骤 8、9，总结 Service 和 DAO 之间的调用关系，思考事务操作

相关代码所处的位置，并记录下来；

（5）根据实验步骤 12，思考表单域中 value 值的写法，总结显示 value 值的实现

方法并记录下来；

（6）碰到的问题及解决方案或思考；

（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。

**二、提高实验——HQL 语言**

（一）实验目的

1、学习 HQL 的使用方法，掌握 select 子句、from 子句、where 子句、order by

子句、聚合函数、group by 子句、子查询等基本语法并能正确运用；

2、理解 HQL 语言是一种面向对象的查询语言，能正确区分 HQL 语言与 SQL 语

8485

言的差别。

（二）基本知识与原理

1、HQL（Hibernate Query Language）语言是 Hibernate 框架定义的查询语言；

2、HQL 语言的语法结构与 SQL 语言非常类似，但 HQL 是面向对象的查询语言，

HQL 语句中使用的是 Java 类名和属性名，大小写敏感；

3、HQL 语言包括 select 子句、from 子句、where 子句、order by 子句、聚合函数、

group by 子句、子查询、连接查询等。

（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：

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-pr2 的 cn.edu.zjut.po 包中创建持久化类 Item.java 以及 Hibernate

映射文件 Item.hbm.xml，具体代码可参照“实验五 Hibernate 基础应用”中

提高实验里的内容；

4、修改 hibernate-prj2 的 Hibernate 配置文件 hibernate.cfg.xml，增加 Item.hbm.xml映射文件的声明；

5、在 hibernate-prj2 的 cn.edu.zjut.dao 包中创建数据库操作类 ItemDAO.java，继

承数据库操作基础类 BaseHibernateDAO.java，具体代码如下：

package cn.edu.zjut.dao;

import java.util.List;

import org.hibernate.Query;

import org.apache.commons.logging.Log;

import org.apache.commons.logging.LogFactory;

public class ItemDAO extends BaseHibernateDAO{

private static final Log log = LogFactory.getLog(ItemDAO.class);

public List findByHql(String hql) {

log.debug("finding Item instance by hql");

try {

String queryString = hql;

Query queryObject = getSession().createQuery(queryString);

return queryObject.list();

} catch (RuntimeException re) {

log.error("find by hql failed", re);

throw re;

}

}

}

6、在 hibernate-prj2 的 cn.edu.zjut.service 包中创建 ItemService.java，使用 from 子

句实现最简单的查询，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 findByHql() {

ItemDAO dao = new ItemDAO();

**String hql = "from cn.edu.zjut.po.Item";**

List list = dao.findByHql(hql);

dao.getSession().close();

return list;

}

}

7、在 hibernate-prj2 的 cn.edu.zjut.action 包中创建 ItemAction.java，并在其中定义

86findItems()方法用于调用“获取商品信息”逻辑，代码片段如下：

package cn.edu.zjut.action;

import java.util.List;

import cn.edu.zjut.service.ItemService;

public class ItemAction{

private List items;

//省略 getters/setters 方法

public String findItems() {

ItemService itemServ = new ItemService();

items=itemServ.findByHql();

System.out.println("Item Action executed!");

return "success";

}

}

8、在 hibernate-prj2 中新建 itemList.jsp 页面，作为显示商品信息的视图，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>

9、修改 hibernate-prj2 的 loginSuccess.jsp 页面，在视图中增加超链接，用于查看

商品信息，代码片段如下：

<a href="./findItems">查看商品信息</a>

10、修改 hibernate-prj2 的 struts.xml 文件，增加 ItemAction 的配置并设置页面导

航；

11、将 hibernate-prj2 重新部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页

87面，并记录运行结果；

12、修改 ItemService.java，将 hql 语句替换成“from Item”，省略包名，直接通

过类名查询，代码片段如下：

String hql = "from Item";

13、将 hibernate-prj2 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；

14、修改 ItemService.java 中的 hql 语句，使用 as 为类取名，以便在其它地方使

用，代码片段如下：

String hql = "from Item as item";

15、将 hibernate-prj2 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；

16、修改 ItemService.java 中的 hql 语句，使用 select 子句查询商品名称，代码片

段如下：

String hql = "select item.title from Item as item";

17、由于 select 子句只返回 title 属性，则返回结果将直接封装到该元素类型的集

合中，即封装到 List<String>集合中，因此需要修改 itemList.jsp 页面，显示

商品名称，代码片段如下：

<table border=1>

<tr> <th>书名</th> </tr>

<s:iterator value="items" **id="object"**>

<tr>

<td><s:property **value="object"**/></td>

</tr>

</s:iterator>

</table>

18、将 hibernate-prj2 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；

19、修改 ItemService.java 中的 hql 语句，使用 select 子句查询商品名称和商品价

格，代码片段如下：

String hql = "select item.title, item.cost from Item as item";

20、若 select 子句返回多个属性，则返回结果将封装到 List<Object[]>集合中，因

此需要修改 itemList.jsp 页面，显示商品名称和商品价格，代码片段如下：

<table border=1>

<tr> <th>书名</th> <th>单价</th> </tr>

<s:iterator value="items" **id="object"**>

<tr>

<td><s:property **value="#object[0]"**/></td>

<td><s:property **value="#object[1]"**/></td>

</tr>

88</s:iterator>

</table>

21、将 hibernate-prj2 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；

22、修改 ItemService.java 中的 hql 语句，尝试使用聚集函数、where 子句、order

by 子句和子查询进行数据查询，并通过 itemList.jsp 页面显示查询结果。

（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）查找相关资料，完成实验步骤 22，并记录运行结果；

（3）结合实验过程，查找相关资料，总结 HQL 的常用语句及语法规则；

（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。

**三、扩展实验——深入 Hibernate 配置文件**

（一）实验目的

1、深入学习 Hibernate 配置文件中的配置属性，了解数据库连接池配置等可选配

置属性的作用；

2、深入学习 Configuration 对象，能通过编程方式创建 Configuration 实例并为该

对象设置一系列属性。

（二）基本知识与原理

1、Hibernate 配置文件中包括 JDBC 连接属性、数据库方言、Hibernate 事务属性、

二级缓存等配置属性；

2、一个 Configuration 实例代表了一个应用程序中 Java 类型到 SQL 数据库映射

的完整集合；

3、每个 Hibernate 配置文件对应一个 Configuration 对象，但在没有任何配置文件

的情况下，也可以通过编程方式创建 Configuration 对象；

4、Configuration 对象提供了若干方法，如：

89（1）Configuration addResource(String resourceName)：用于为 Configuration 对象

添加一个映射文件；

（2）Configuration setProperty(String property, String value)：用于为 Configuration

对象设置属性。

（三）实验内容及步骤

1、修改 hibernate-prj2 的 Hibernate 配置文件 hibernate.cfg.xml，增加配置属性，

使得能在控制台输出 Hibernate 生成的格式良好的 SQL 语句，并在其中添加

有助于调试的注释，代码片段如下：

<hibernate-configuration>

<session-factory name="HibernateSessionFactory">

……

<property name="hibernate.show\_sql">true</property>

<property name="hibernate.format\_sql">true</property>

<property name="hibernate.use\_sql\_comments">true</property>

</session-factory>

</hibernate-configuration>

2、将 hibernate-prj2 重新部署在 Tomcat 服务器上,通过浏览器访问 login.jsp 页面，

查看控制台输出的 SQL 语句；

3、修改 hibernate.cfg.xml，增加 C3P0 连接池的配置属性，代码片段如下：

<hibernate-configuration>

<session-factory name="HibernateSessionFactory">

……

<property name="hibernate.c3p0.max\_size">20</property>

<property name="hibernate.c3p0.min\_size">1</property>

<property name="hibernate.c3p0.timeout">1800</property>

<property name="hibernate.c3p0.max\_statements">50</property>

</session-factory>

</hibernate-configuration>

4、将 hibernate-prj2 重新部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页

面，查看控制台的相关输出（INFO）并记录下来；

5、删除 hibernate-prj2 的 Hibernate 配置文件 hibernate.cfg.xml；

6、修改cn.edu.zjut.dao包中的HibernateUtil.java，通过编程方式创建Configuration

对象，代码片段如下：

public class HibernateUtil {

private static Configuration configuration = new Configuration();

……

static {

try {

90configuration

//通过 setProperty 方法设置 Hibernate 的连接属性

.setProperty("hibernate.connection.driver\_class",

"com.mysql.jdbc.Driver")

.setProperty("hibernate.connection.url",

"jdbc:mysql://localhost:3306/hibernatedb")

.setProperty("hibernate.connection.username", "root")

.setProperty("hibernate.connection.password", "")

.setProperty("hibernate.dialect",

"org.hibernate.dialect.MySQLDialect")

//通过 addResource 方法添加映射文件

.addResource("cn/edu/zjut/po/Customer.hbm.xml")

.addResource("cn/edu/zjut/po/Item.hbm.xml");

sessionFactory = configuration.buildSessionFactory();

} catch (Exception e) { …… }

}

}

7、将 hibernate-prj2 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；

8、在 MySQL 的 hibernatedb 数据库中删除 customer 数据表；

9、修改 HibernateUtil.java，通过编程方式添加 hibernate.hbm2ddl.auto 配置属性，

使得能根据映射文件自动建立数据库表，代码片段如下：

public class HibernateUtil {

private static Configuration configuration = new Configuration();

……

static {

try {

……

configuration

.setProperty("hibernate.hbm2ddl.auto", "create-drop");

} catch (Exception e) { …… }

}

}

10、将 hibernate-prj2 重新部署在 Tomcat 服务器上，通过浏览器访问 register.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，写出实验步骤 1、3 中所使用的配置属性的

91作用，并记录下来；

（3）根据实验步骤 9，查找相关资料，写出“hibernate.hbm2ddl.auto”配置属性

及其取值的作用；

（4）根据实验步骤 6，查找相关资料，总结通过编程方式创建 Configuration 实

例或设置配置属性的基本步骤，思考在实际应用中如何将 Hibernate 配置文

件设置配置属性与编程方式设置配置属性相结合，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。

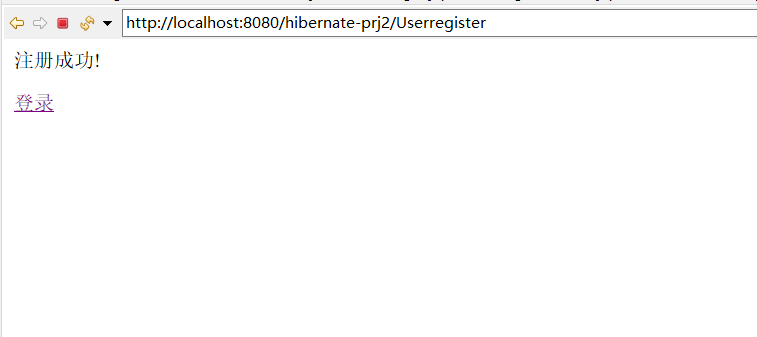
**一、基础实验——Hibernate 常用 API**

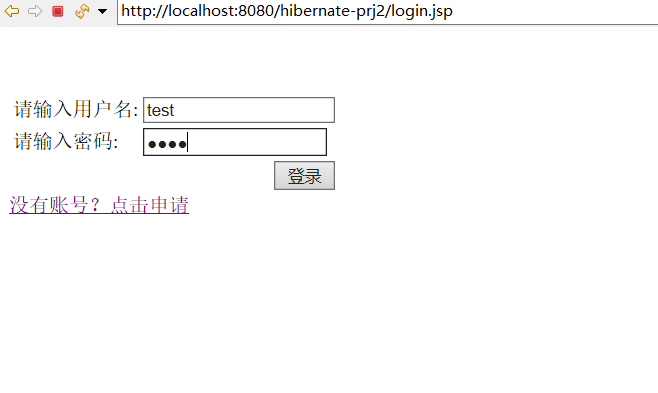
1. **运行结果截图；**

1.注册界面:



2.注册成功

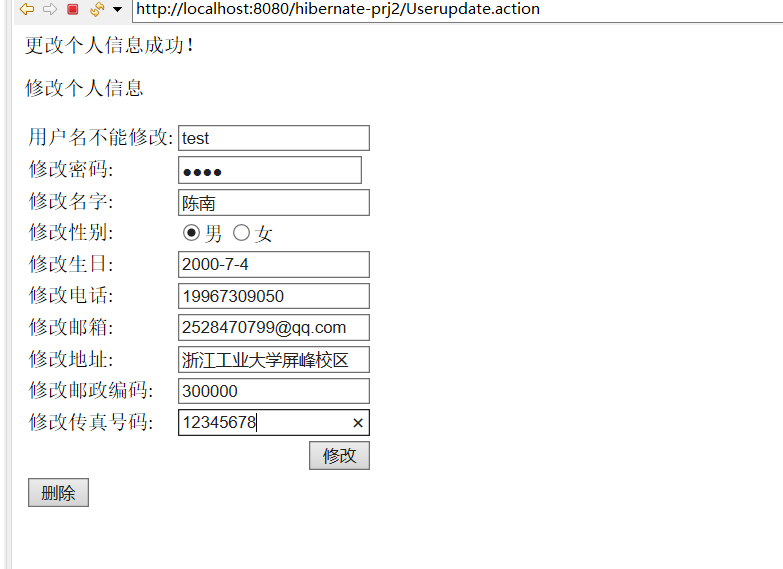
3.登录界面

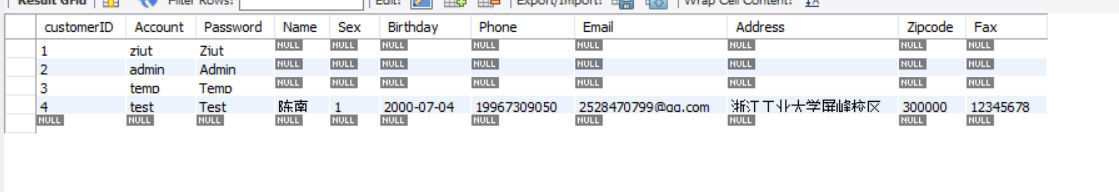


1. 登录成功界面（登录失败仍会转至login.jsp）



1. 修改成功界面（将邮政编码由310012改为300000，将传真号码由88484666改为12345678）





1. **结合实验过程，查找相关资料，总结利用 Hibernate 基本 API 载入配置文件、建立数据库连接并进行数据库操作的基本步骤；**
2. 读取Hibernate配置文件。使用Hibernate包中的Configuration类读取配置文件的方法configure()，读取默认的hibernate.cfg.xml文件  
   2.Hibernate中，数据库操作是用Hibernate包中的Session完成的，Session有Hibernate包中的SessionFactory管理，所以需要生成SessionFactory  
   3.利用SessionFactory打开Session  
   4.利用Session操作数据库

**(3)结合实验过程，查找相关资料，总结 Session 中的主要方法，及其与数据库操作的对应关系；**

1. save方法

这个方法表示将一个对象保存到数据库中，可以将一个不含OID的new出来的临时对象转换为一个处于Session缓存中具有OID的持久化对象。

需要注意的是：在save方法前设置OID是无效的但是也不会报错，在save方法之后设置OID程序会抛出异常，因为持久化之后的对象的OID是不可更改的，因为对象的OID此时和数据库中的一条记录对应。

2.persist方法

这个方法基本个save方法差不多，唯一的区别是，在这个方法之前也不可以设置对象的OID，否则不会执行插入操作，而是会抛出异常。

3.get/load方法

这个方法是从数据库中获取一个对象到Session缓存中，load方法也是这个功能。二者有着明显的区别是：

一是若数据库中有和传入OID相对性的数据时，get方法是立即加载并且发送sql语句来从数据库中获取对象，而load对象时采用一种延迟加载的方式来获取，即只有当需要使用这个对象的时候才会发送SQL语句。

二是若数据库中没有和传入OID相对性的数据时，get方法返回null值，load方法会抛出一个对象未找到的异常。

三是若在使用返回对象的属性前将session进行了关闭，那么load方法会抛出一个lazyInitialedException--懒加载异常。这是因为load方法返回一个代理对象，当我想向这个代理对象中填充属性时，因为session已经关闭，那么久无法加载从而抛出懒加载异常。

4.update方法

这个方法顾名思义就是更新一个对象在数据库中的对照情况，从而使一个游离对象转换为一个持久化对象。

若是更新一个持久化对象，不需要再显式子的进行update方法，因为在commit方法中已经进行过flush了。

5.saveOrUpdate方法

这个方法同时包含了前边save和update的功能。当对象时临时的，那么执行save方法，当对象时游离的，那么执行update方法。

6. delete方法

顾名思义，这个方法就是来删除游离的或者持久化的对象及其在数据库中对应的记录。

7. evict方法

这个方法就是将持久化对象从session缓存中删除，使其成为一个游离的对象。

**(4)根据实验步骤 8、9，总结 Service 和 DAO 之间的调用关系，思考事务操作相关代码所处的位置，并记录下来；**

功能上说Dao层主要是对封装对数据库的访问,查询数据,或者对对象持久化,不应改包含任何的业务逻辑，service层包含对数据访问和修改。所以业务逻辑应该写在service层内。Service层调用DAO层的

**(5)根据实验步骤 12，思考表单域中 value 值的写法，总结显示 value 值的实现方法并记录下来；**

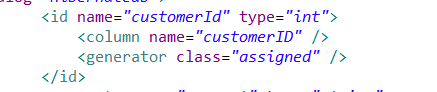
1.用value="#request.tip"显示存在request中的tip

2.用表达式，形如value="%{#session.User.userName}"

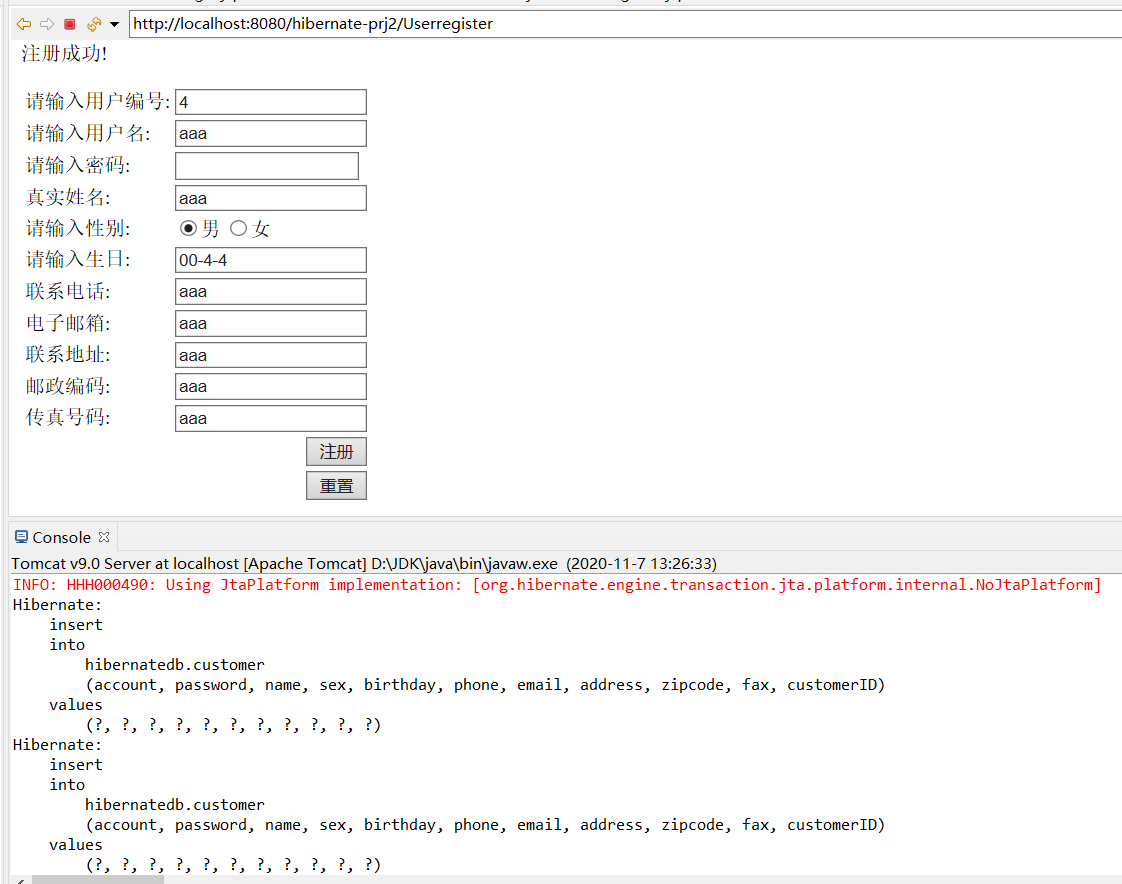
**(6)碰到的问题及解决方案或思考；**

**注册时遇到的问题**

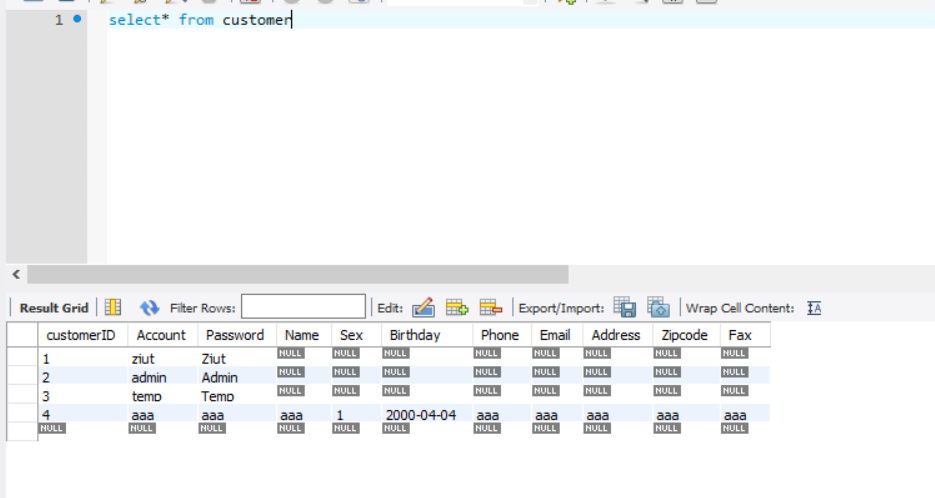
1.主键配置方式为assigned



注册页面及工作台显示的sql语句：

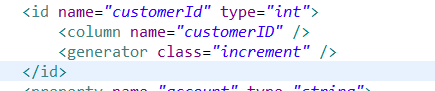


发现数据库中只有一条记录：

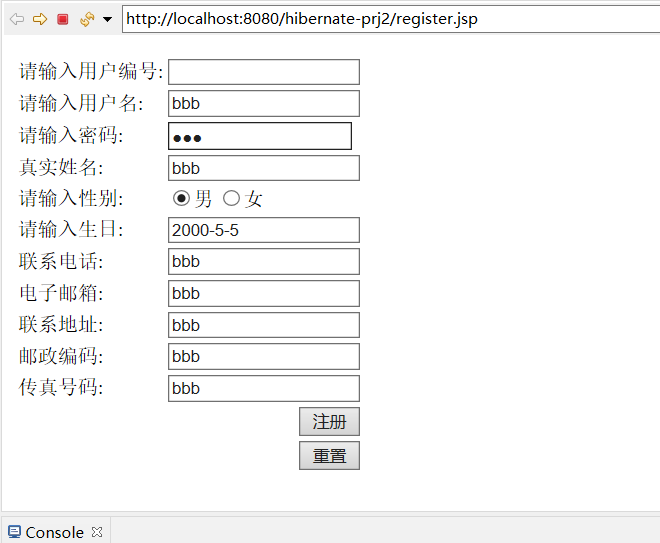


问题点：代码配置的为注册成功会转到注册成功页面，注册失败重新转到注册页面（register.jsp）,这里明明注册成功了（我觉得成功了……?）但他还是转到了注册页面，而且上面的提示还是注册成功，就很奇怪。

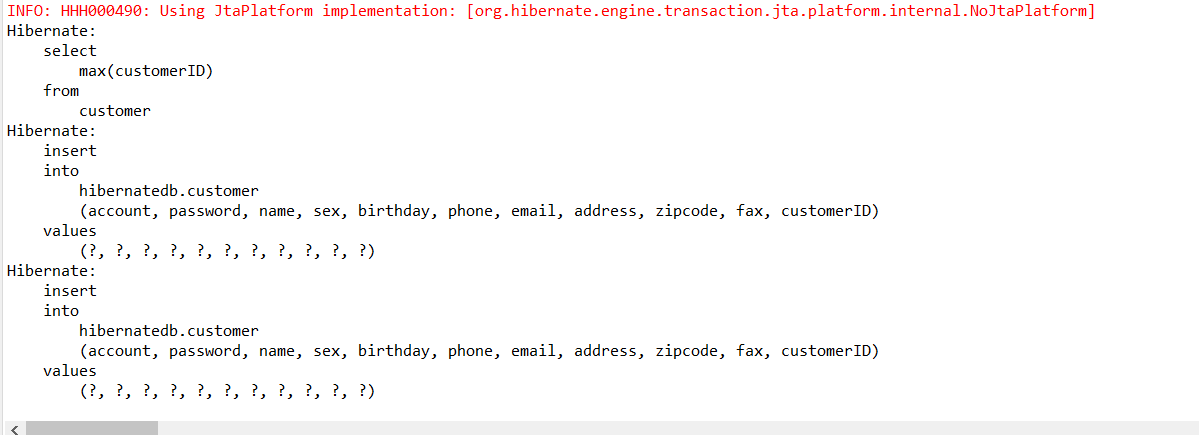
1. 主键配置方式为increment



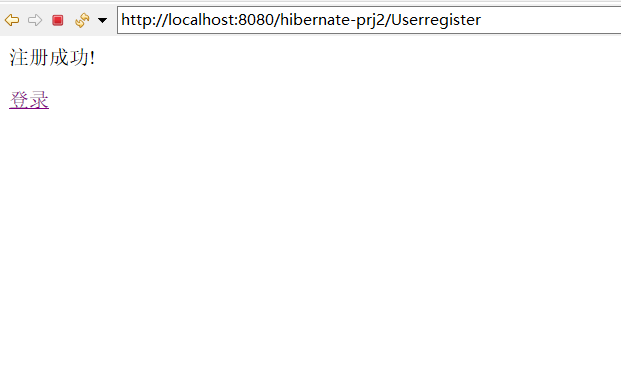
注册页面：（自增长就没有自己填入用户编号）



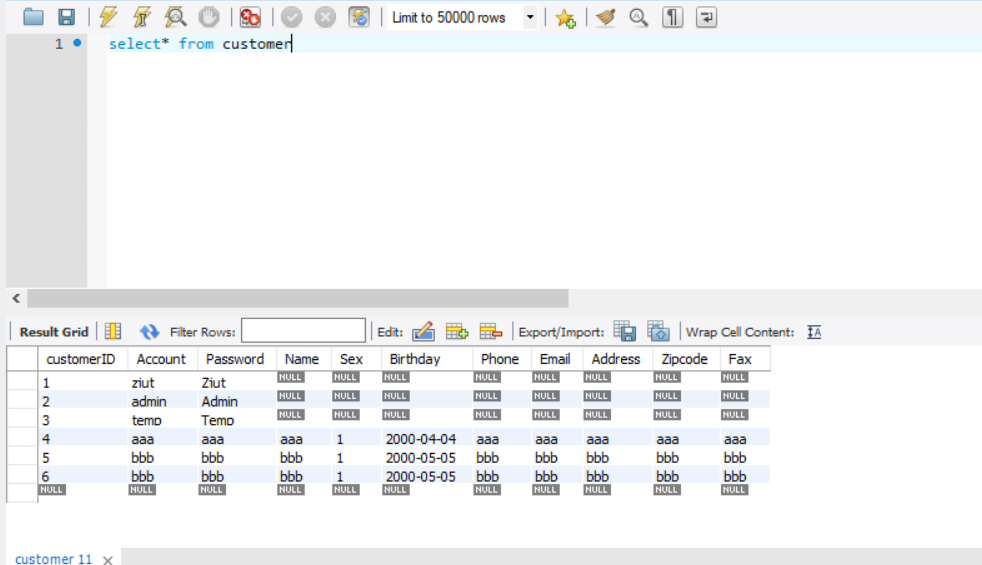
工作台显示出来的sql语句：



跳转后页面：



数据库页面：（产生两条相同的数据记录）



问题点：数据库中出现两条相同的数据，但是注册之后正确的转到了注册成功的页面（registerSuccess.jsp）。

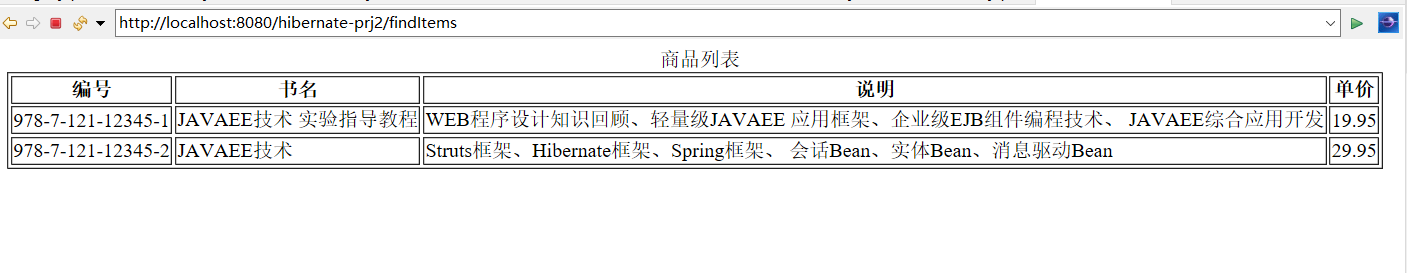
解决方法：仔细查看出现两条sql语句的问题，找到关键在于为什么会执行两次，在姗姗老师的指导下，检查了表单有没有提交两遍……，发现在Action中重写了两次service方法，后解决。

1. **实验收获及总结。**
2. 明白并实践了如何正确的配置hibernate并且合理的利用DAO层和Service层，清楚了事务代码应该加在哪一层中。
3. 更了解了hibernate连接数据库的操作。

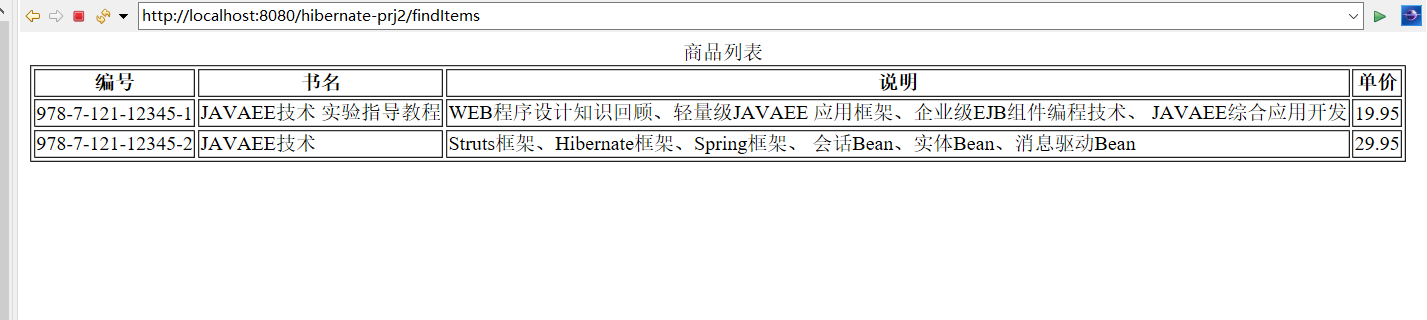
**二、提高实验——HQL 语言**

**（1）运行结果截图；**

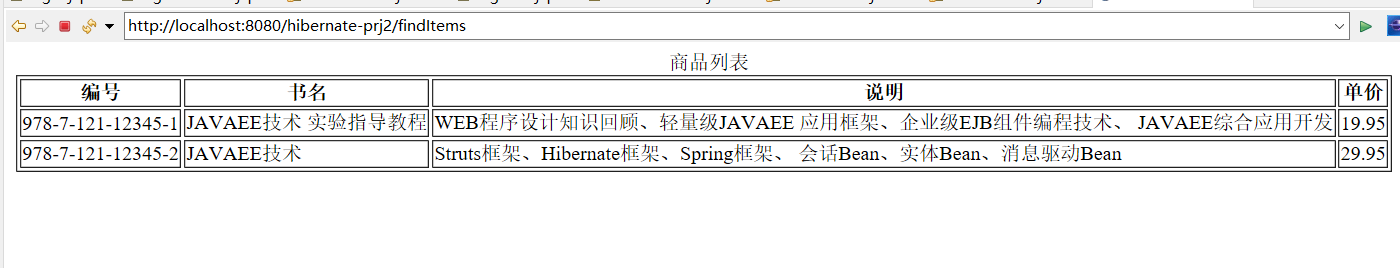
1.第十一小题，用from cn.edu.zjut.po.Item进行查询



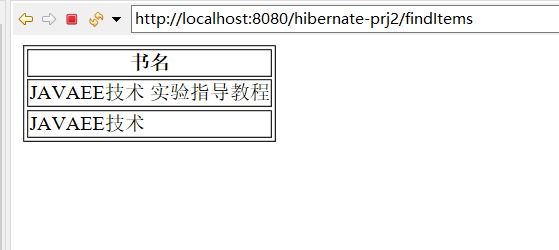
2.第十三小题，用from Item进行查询，也能成功查询



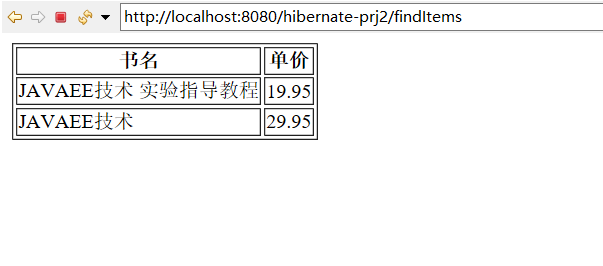
1. 第十五小题，用from Item as item进行查询，也能成功查询



4.第十八小题，查询书名



5.第二十小题，查询书名和单价



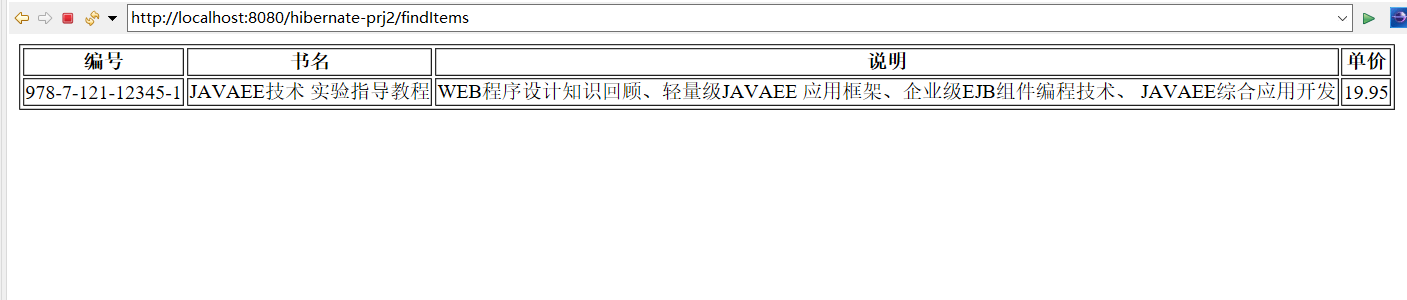
1. **查找相关资料，完成实验步骤 22，并记录运行结果；**
2. 用count语句查询总共有几本书

String hql = "select count(\*) from Item as count";



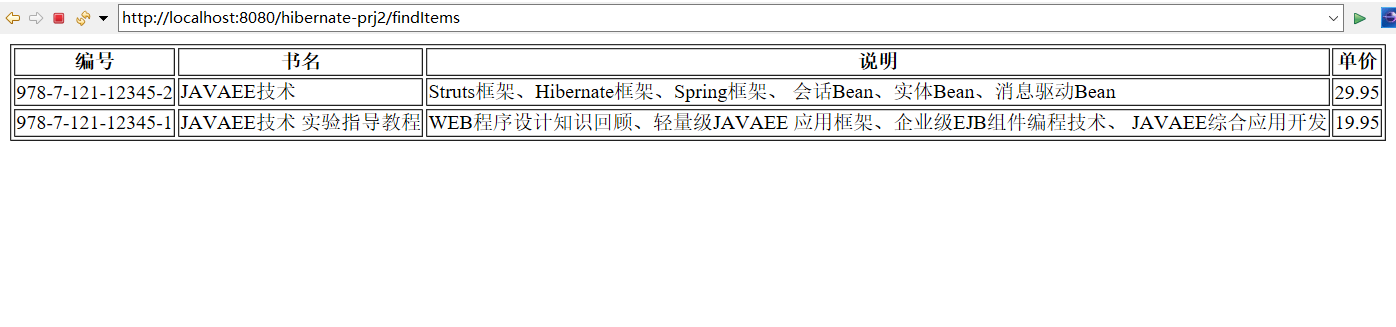
1. 用where查询单价小于20的书籍

String hql = "from Item where cost<20"



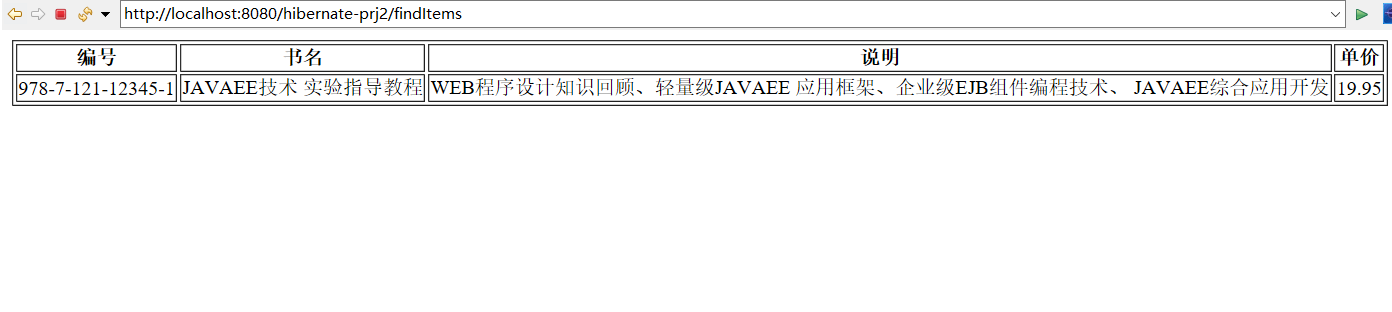
1. 用order根据单价降序排序书籍

String hql = "from Item order by cost DESC";



1. 用子查询查询单价小于20的书籍

String hql = "From Item i where i.cost<20";



1. **结合实验过程，查找相关资料，总结 HQL 的常用语句及语法规则；**

完整的HQL语句形势如下：

Select/update/delete…… from …… where …… group by …… having …… order by …… asc/desc

1.from子句

from Person

表明从Person持久化类中选出全部的实例。

推荐：from Person as p

2.select子句

select p.name from Person as p

select p.name.firstName from Person as p

select new list(p.name, p.address) fromPerson as p

select new ClassTest(p.name, p.address) fromPerson as p (有前提)

3.聚集函数

avg,count,max,min,sum

select count(\*) from Person

select max(p.age) from Person as p

select p.name || "" || p.addressfrom Person as p

4.多态查询

from Person as p

from java.lang.Object o

from Named as n

5.where子句

from Person where name like "tom%"

from Person as p where p.name like"tom%"

6.表达式

from DomesticCat cat where cat.name between'A' and 'B'

from DomesticCat cat where cat.name in('Foo', 'Bar', 'Baz')

from DomesticCat cat where cat.name notbetween 'A' and 'B'

from DomesticCat cat where cat.name not in('Foo', 'Bar', 'Baz')

7.order by子句

from Person as p

order by p.name, p.age

from Person as p

order by p.name asc, p.age desc

8.group by子句

select cat.color, sum(cat.weight), count(cat)

from Cat cat

group by cat.color

//select后出现的id处出现在group by之后，而name属性则出现在聚集函数中

select foo.id, avg(name), max(name)

from Foo foo join foo.names name

group by foo.id

9.子查询

from Cat as fatcat

where fatcat.weight > (selectavg(cat.weight) from DomesticCat cat)

from Cat as cat

where not (cat.name, cat.color) in (

select cat.name, cat.color from DomesticCatcat

)

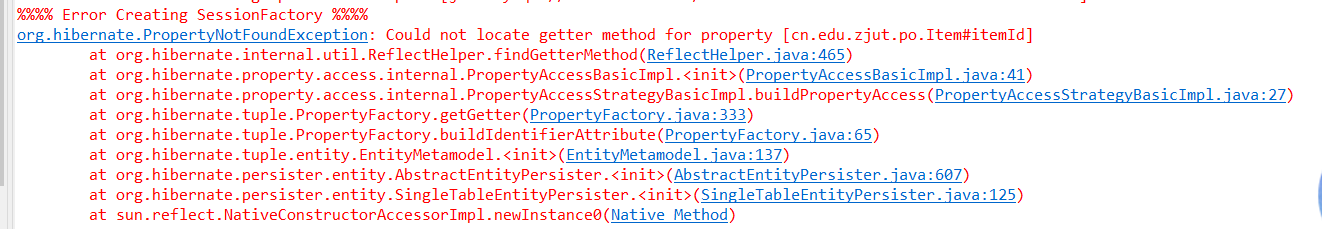
10.fetch关键字

from Person as p join p.scores

from Document fetch all properties order byname

from Document doc fetch all properties wherelower(doc.name) like '%cat%'

1. **碰到的问题及解决方案或思考；**
2. 报错如下



解决方法：检查Item中的getter方法，发现书写无问题，再检查item.hbm.xml文件，发现其中的itemId与item文件中的itemID不对应。修改后解决。

1. itemlist中展示不出ISBN这一列。

解决方法：还是itmList.jsp中itemID一列不对应而出错，修改后解决。

**（5）实验收获及总结。**

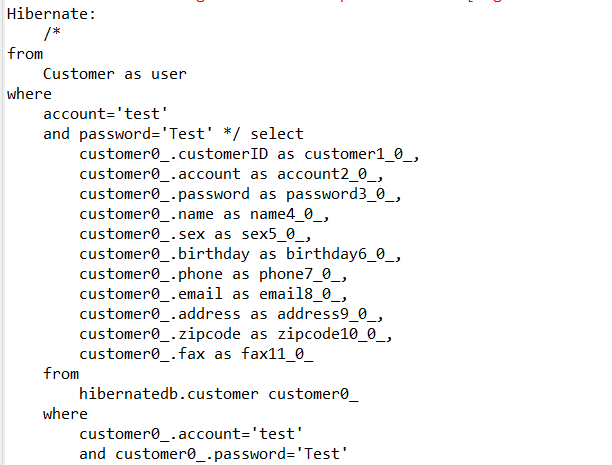
1.熟悉了HQL的常用语法及用HQL查询的查询步骤

2.掌握了HQL与SQL之间的区别与对应关系。

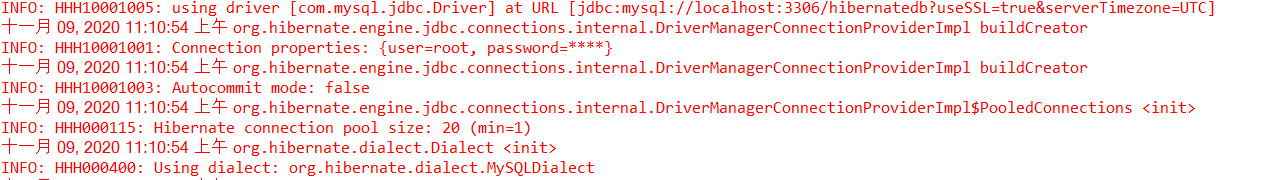
**三、扩展实验——深入 Hibernate 配置文件**

1. **运行结果截图；**

1.第二小题，控制台的sql语句



2.第四小题，相关info

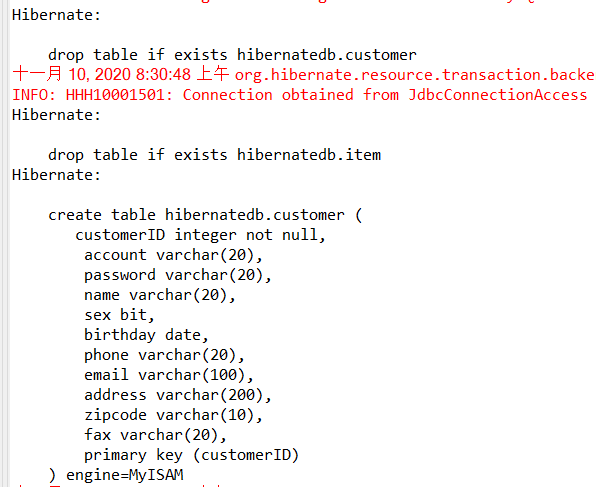


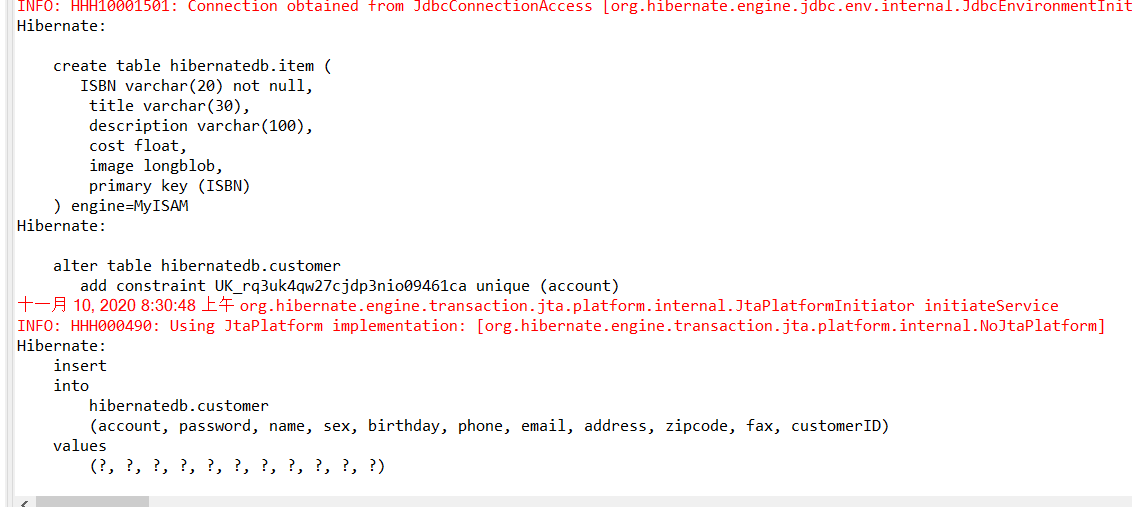
1. 第7小题，也能成功访问

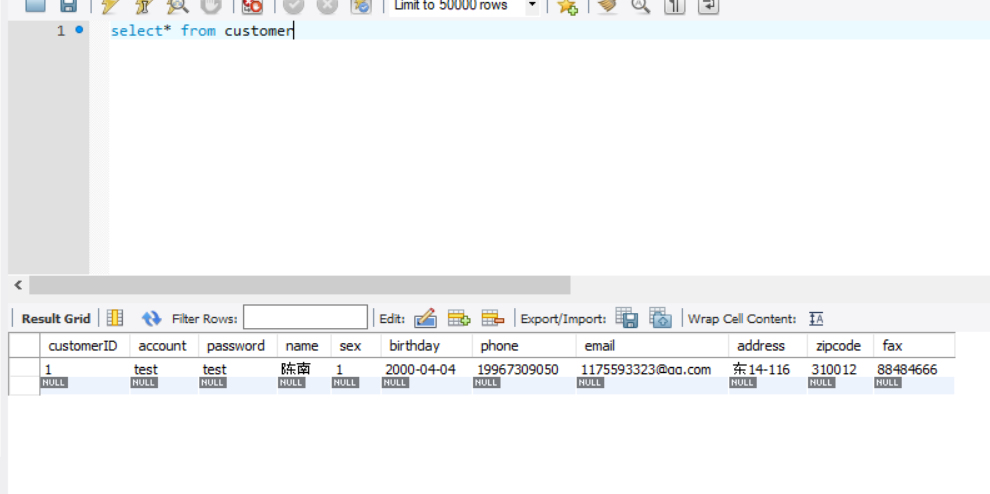


4.第十小题









**（2）结合实验过程，查找相关资料，写出实验步骤 1、3 中所使用的配置属性的作用，并记录下来；**

1.show\_sql(输出sql)，能在工作台上显示sql语句。

2.format\_sql(将输出的sql格式化)

将show\_sql配制为true，输出的sql无论多长都只会在一行显示。我们可以将format\_sql配制为true，这样Hibernate会先将sql格式化，然后再输出，方便了我们调试。

3.use\_sql\_comments(生成SQL注释)

将use\_sql\_comments配制为true，Hibernate会为我们生成SQL的注释，更加的方便了我们调试

**（3）根据实验步骤 9，查找相关资料，写出“hibernate.hbm2ddl.auto”配置属性及其取值的作用；**

hibernate.hbm2ddl.auto参数的作用主要用于：自动创建|更新|验证数据库表结构。如果不是此方面的需求建议set value="none"。

create：

每次加载hibernate时都会删除上一次的生成的表，然后根据你的model类再重新来生成新表，哪怕两次没有任何改变也要这样执行，这就是导致数据库表数据丢失的一个重要原因。

create-drop ：

每次加载hibernate时根据model类生成表，但是sessionFactory一关闭,表就自动删除。

update：

最常用的属性，第一次加载hibernate时根据model类会自动建立起表的结构（前提是先建立好数据库），以后加载hibernate时根据 model类自动更新表结构，即使表结构改变了但表中的行仍然存在不会删除以前的行。要注意的是当部署到服务器后，表结构是不会被马上建立起来的，是要等 应用第一次运行起来后才会。

validate ：

每次加载hibernate时，验证创建数据库表结构，只会和数据库中的表进行比较，不会创建新表，但是会插入新值。

**（4）根据实验步骤 6，查找相关资料，总结通过编程方式创建 Configuration 实**

**例或设置配置属性的基本步骤，思考在实际应用中如何将 Hibernate 配置文**

**件设置配置属性与编程方式设置配置属性相结合，并记录下来；**

Configuration addResource(String resourceName)：用于为Configuration对象添加一个映射文件。

Configuration setProperties(Properties properties)：用于为Configuration对象设置一系列属性，这系列属性通过Properties实例传入。Configuration setProperty(String propertyName, String value)：用于为Configuration对象设置一个单独的属性。

正是通过如上三个方法，可以无须任何配置文件支持，直接通过编程方式创建Configuration实例

在代码中设置Hibernate 配置属性也并非完全没有用处。这种策略可将部分关键配置属性放在代码中添加。

1. **碰到的问题及解决方案或思考；**

1.问题：不能创建数据库表，说SQL语句有问题，上网查阅资料之后发现有可能是SQL方言问题，将MySQLDialect改成MySQL5Dialect后解决。

2.思考：在最后一步，hibernate自动生成表时，hibernate会将item表和customer表全都先删除，再做，这样很容易造成数据的丢失，所以在选在用hibernate.hbm2ddl.auto属性时需要谨慎。

**（6）实验收获及总结。**

1.学会了如何在工作台上显示sql语句，对检查错误修改代码可以起到很好的作用，方便了调试。

2.学会了不配置hibernate.cfg.xml文件而连接数据库的方式。

3.了解了hibernate.hbm2ddl.auto这一属性。