序号: 24

学号: 16430121

常州大学

实 验 报 告

课 程 名 称：Java EE 程序设计与应用开发

实 验 题 目：Hibernate 框架的使用

实 验 次 数：第 五 次实验　实验时间：2018 年 5 月 7 日

学 生 姓 名：林锦雄

学　　院：信息数理　专 业 班 级：计算机 162

指 导 教 师：陆洁茹　评 阅 成 绩：

一、实验目的：

1. 掌握 Hibernate 框架的基本原理及使用方法。
2. 要求：利用 Hibernate 技术访问数据库。

二、实验仪器、设备

1、硬件环境

PC 微机；2G 以上内存；VGA 显示格式。

2、软件环境

Windows XP 以上操作系统，JDK，Tomcat 服务器等。

三、实验内容与要求

在数据库中建立表格 T_BOOK(BOOKID, BOOKNAME, BOOKPRICE)，插入一些记录。如图 1 所示。

```
MariaDB [(none)]> use books;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [books]> select * from T_BOOK;
+--------+--------------------+-----------+
| BOOKID | BOOKNAME           | BOOKPRICE |
+--------+--------------------+-----------+
| 10001  | 童话故事           |      32.5 |
| 10002  | 安徒生童话         |      23.8 |
| 10003  | 格林童话           |      15.8 |
| 10004  | 希腊神话全集       |    159.99 |
| 10005  | 数据结构           |      43.5 |
| 10006  | 编译原理           |      57.9 |
| 10007  | BigBen             |      32.3 |
+--------+--------------------+-----------+
7 rows in set (0.001 sec)

MariaDB [books]>
```

图 1. books 数据库中的 T_BOOK 表

1. 制作一个查询页面，输入两个数字，显示价格在这两个数字之间的图书信息。要求使用 Hibernate 的 HQL 查询方法。

queryBooks.jsp 是用一些简单的 html 标签做的简易版图书查询表单，有输入提示和两个输入框，分别为需要查询的图书的最低价和最高价。如图 1 所示。

```
1. <--! queryBooks.jsp -->
2. <form action="queryByPrice.action" method="post">
3.     请输入图书价格区间：<input type="text" name="priceMin"> 到
4.     <input type="text" name="priceMax"><br>
5.     <input type="submit" value="查询">
6. </form><hr>
```

图 2. 根据价格检索初始页面

searchBooksByPriceAction 类为使用 Struts2 框架书写的 Action 类，区别于 Struts1.x，这里把原本应该另写的 ActionForm 类取消，直接把对应于前端表单的属性封装在该类中，然后书写该 Action 类的 execute 方法。

在该 Action 的 execute 方法中，先实例化一个 DAO 层的 QueryBooks 类，调用其 queryBooksByPrice 方法，传入代表价格区间的两个浮点数，返回 ArrayList 对象代表查询结果，然后把结果通过保存在 session 中，返回逻辑名称，最后框架通过逻辑名称找到相对应的地址进行跳转。设计一个查询区间，如图 3 所示。

```
1. //searchBooksByPriceAction 类
2. private double priceMin;
3. private double priceMax;
4. public String execute() {
5.     QueryBooks queryBooks = new QueryBooks();
6.     ArrayList<TBookEntity> books = null;
7.     books = queryBooks.queryBooksByPrice(priceMin,priceMax);
8.     Map session = ActionContext.getContext().getSession();
9.     session.put("books",books);
10.      return "success";
11. }
```

图 3. 设计一个查询区间

QueryBooks 类为位于 DAO 层的一个类，他的作用是方法数据库。其中的一个方法 queryBooksByPrice 就是专为通过价格区间查询图书信息的表单进行数据库访问。首先是实例化配置对象、session 工厂、session 对象，然后通过 hibernate 的 HQL 查询方法查询数据库中指定价格区间的全部图书信息，并把查询结果存入一个 ArrayList 对象返回给 Action。查询结果如图 4 所示。

```
1. //QueryBooks 类
2. public ArrayList queryBooksByPrice(double priceMin, double priceMax) {
3.     Configuration configuration = new Configuration().configure();
4.     SessionFactory sessionFactory = configuration.buildSessionFactory();
5.     Session session = sessionFactory.openSession();
6.     String hql = "from TBookEntity where bookprice >= :priceMin and bookprice <=
:priceMax";
7.     Query query = session.createQuery(hql);
8.     query.setParameter("priceMin",priceMin);
9.     query.setParameter("priceMax",priceMax);
10.      ArrayList books = (ArrayList) query.list();
11.      session.close();
12.      return books;
13.  }
```

图 4. 指定价格区间的查询结果

TbookEntity.java 就是一个与数据库对应的 PO，它的编写方法和 JavaBean 相似，通过配置文件将它与数据库指定的表相关联，在程序中对数据库的操作都改为对该类的操作，这样可以大大降低了程序的耦合度，编写后端代码的程序员不需要关系数据库的使用类型，框架会自动去匹配。

```
1. @Entity
2. @Table(name = "T_BOOK", schema = "books")
3. public class TBookEntity {
4.     private String bookid;
5.     private String bookname;
6.     private Double bookprice;
7.     @Id
8.     @Column(name = "BOOKID")
9.     public String getBookid() {  return bookid; }
10.      public void setBookid(String bookid) {  this.bookid = bookid; }
11.      @Basic
12.      @Column(name = "BOOKNAME")
13.      public String getBookname() {  return bookname; }
```

```
14.     public void setBookname(String bookname) { this.bookname = bookname; }
15.     @Basic
16.     @Column(name = "BOOKPRICE")
17.     public Double getBookprice() { return bookprice; }
18.     public void setBookprice(Double bookprice) { this.bookprice = bookprice; }
19.     @Override
20.     public boolean equals(Object o) {
21.         if (this == o) return true;
22.         if (o == null || getClass() != o.getClass()) return false;
23.         TBookEntity that = (TBookEntity) o;
24.         return Objects.equals(bookid, that.bookid) &&
25.                 Objects.equals(bookname, that.bookname) &&
26.                 Objects.equals(bookprice, that.bookprice);
27.     }
28.     @Override
29.     public int hashCode() { return Objects.hash(bookid, bookname, bookprice); }
30. }
```

TbookEnitity.hbm.xml 为数据表和类相关联的配置文件，这里为表现为 Table.TBookEntity 类和 books.T_BOOK 表进行关联，所有对 TbookEntity 的操作就可以相应地操作到 T_BOOK 表。

```
1. <--! TbookEnitity.hbm.xml -->
2. <hibernate-mapping>
3.     <class name="Table.TBookEntity" table="T_BOOK" schema="books">
4.         <id name="bookid" column="BOOKID"/>
5.         <property name="bookname" column="BOOKNAME"/>
6.         <property name="bookprice" column="BOOKPRICE"/>
7.     </class>
8. </hibernate-mapping>
```

hibernate.cfg.xml 是 hibernate 的配置文件，这里保存了数据库的连接信息以及和 hibernate 映射文件的信息。我使用的是 MySQL 数据库，所以使用<property name="dialect">org.hibernate.dialect.MySQLDialect</property>。在这里还关联到 Table/TBookEntity.hbm.xml 映射文件。

```
1. <--! hibernate.cfg.xml -->
2. <hibernate-configuration>
3.     <session-factory>
4.         <property name="connection.username">root</property>
5.         <property name="connection.password">bingjie123</property>
6.         <property name="connection.url">jdbc:mysql://localhost:3306/books?charact
erEncoding=utf-8</property>
7.         <property name="connection.driver_class">com.mysql.jdbc.Driver</property>

8.         <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
9.         <mapping class="Table.TBookEntity"/>
10.          <mapping resource="Table/TBookEntity.hbm.xml"/>
11.     </session-factory>
12. </hibernate-configuration>
```

struts.xml 是 Struts2 的配置文件，在这里配置 Action 类的地址，还有 Action 中的逻辑名称对应的地址信息。

```xml
<!-- struts.xml -->
<package name="struts2" extends="struts-default">
    <action name="queryByPrice" class="Action.searchBooksByPriceAction">
        <result name="success">/WEB-INF/jsps/showBooks.jsp</result>
    </action>
</package>
```

showBooks.jsp 为查询结果页面，把 Action 类保存在 session 对象中的查询结果一一显示在网页上。

```jsp
<!-- showBooks.jsp -->
<h2>查询结果如下：</h2><hr>
<table border="2">
    <tr>
        <th>图书编号</th>
        <th>图书名称</th>
        <th>图书价格</th>
    </tr>
    <%
        ArrayList<TBookEntity> books = (ArrayList) session.getAttribute("books"
);
        if(books != null) {
            for(int i = 0;i < books.size();i++) {
     %>
     <tr>
        <td><%=books.get(i).getBookid()%></td>
        <td><%=books.get(i).getBookname()%></td>
        <td><%=books.get(i).getBookprice()%></td>
     </tr>
     <%
            }
        }
     %>
</table>
```

2. 编写一个网页，输入图书名称的模糊资料，并输入一个数字，然后查询价格在该数字以下的图书信息。要求使用 Hibernate 的准则查询方法。

queryBooks.jsp 和第一题类似，也是用一些简单的 html 标签做的简易版图书查询表单，在第一个表单的基础上添加一个新表单，这个表单有一个图书名称的输入框和一个最高价格的输入框，图书名称可以是书名的一部分，就是说系统会查询到包含用户输入图书关键字的全部图书，这些图书还会进一步筛选出价格不高于用户设定的最高价格。设定一组查询数据，如图 5 所示。

```jsp
<!-- queryBooks.jsp -->
<form action="queryByName.action" method="post">
    请输入图书名称：<input type="text" name="bookName"><br>
    请输入最高价格：<input type="text" name="maxPrice"><br><br>
    <input type="submit" value="查询">
</form><hr>
```

图 5. 设计一组查询数据

searchBooksByNameAction 类也是使用 Struts2 框架书写的 Action 类。在该 Action 的 execute 方法中，先实例化一个 DAO 层的 QueryBooks 类，调用其 queryBooksByNameAndPrice 方法，传入图书关键字和图书最大价格，返回 ArrayList 对象代表查询结果，然后把结果通过保存在 session 中，返回逻辑名称，最后框架通过逻辑名称找到相对应的地址进行跳转。

```
1. //searchBooksByNameAction 类
2. private String bookName;
3. private Double maxPrice;
4.
5. public String execute() {
6.     QueryBooks queryBooks = new QueryBooks();
7.     ArrayList<TBookEntity> books = null;
8.     books = queryBooks.queryBooksByNameAndPrice(bookName,maxPrice);
9.     Map session = ActionContext.getContext().getSession();
10.      session.put("books",books);
11.      return "success";
12. }
```

QueryBooks 类为位于 DAO 层的一个类，他的作用是方法数据库。其中的一个方法 queryBooksByNameAndPrice 就是专为通过图书名称关键字和图书最大价格的组合信息来进行数据库访问。首先是实例化配置对象、session 工厂、session 对象，然后通过 hibernate 的 HQL 查询方法查询数据库中指定价格区间的全部图书信息，并把查询结果存入一个 ArrayList 对象返回给 Action。在 hibernate 5.2 版本中 criteria.createQuery 方法已不再维护，即过时，使用 criteriaBuilder，criteriaQuery 来取代，使用 Root 来联系表和类，criteria.where 来传入 HQL 参数。

上面设计的查询组合的查询结果，如图 6 所示。

```
1. //QueryBooks 类
2. public ArrayList queryBooksByNameAndPrice(String bookName, double maxPrice) {
```

```
3.      Configuration configuration = new Configuration().configure();
4.      SessionFactory sessionFactory = configuration.buildSessionFactory();
5.      Session session = sessionFactory.openSession();
6.      CriteriaBuilder criteriaBuilder = session.getCriteriaBuilder();
7.      CriteriaQuery<TBookEntity> criteria = criteriaBuilder.createQuery(TBookEntity
.class);
8.      Root<TBookEntity> root = criteria.from(TBookEntity.class);
9.      criteria.where(criteriaBuilder.and(criteriaBuilder.like(root.get("bookname"),
"%" + bookName + "%"),criteriaBuilder.le(root.get("bookprice"),maxPrice)));
10.     ArrayList<TBookEntity> books = (ArrayList) session.createQuery(criteria).li
st();
11.     session.close();
12.     return books;
13. }
```

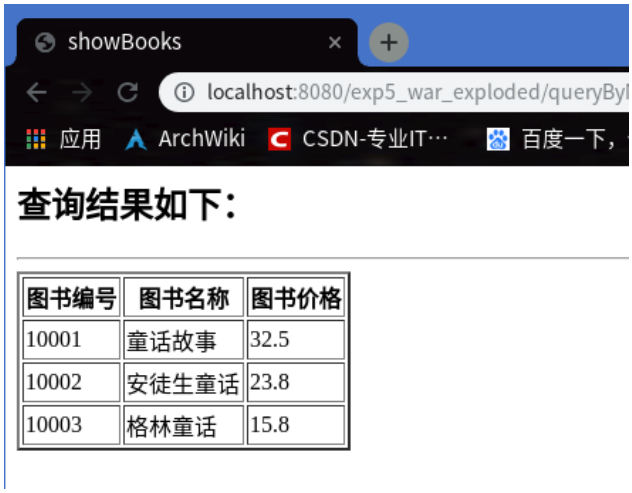

图 6. 指定图书关键字和最高价的查询结果

struts.xml 是 Struts2 的配置文件，在这里配置 Action 类的地址，还有 Action 中的逻辑名称对应的地址信息。

```
1. <--! struts.xml -->
2. <package name="struts2" extends="struts-default">
3.     <action name="queryByName" class="Action.searchBooksByNameAction">
4.         <result name="success">/WEB-INF/jsps/showBooks.jsp</result>
5.     </action>
6. </package>
```

使用 Struts2 框架和 Hibernate 框架构建的本项目的目录树，Struts2 框架相比 Struts1.x 框架，其配置文件的位置改变了，固定在了 src 目录下，这个配置文件不再把 JSP、ActionForm 和 Action 关联起来，因为 Struts2 中和前端表单对应的属性直接书写在了对应的 Action 中。Hibernate 框架有两种配置文件，一种是 Hibernate 的配置文件，这里配置了数据库连接时的信息，还有不同数据库的语言选择信息，该配置文件一般为 hibernate.cfg.xml；另一种时 Hibernate 的映射文件，里面配置 PO 的信息，把数据库表和 PO 类关联起来，这样后端数据就不需要直接操作数据库，只需要操作对应的 PO 类即可通过框架映射到相应的数据库表中。如图 7 所示。

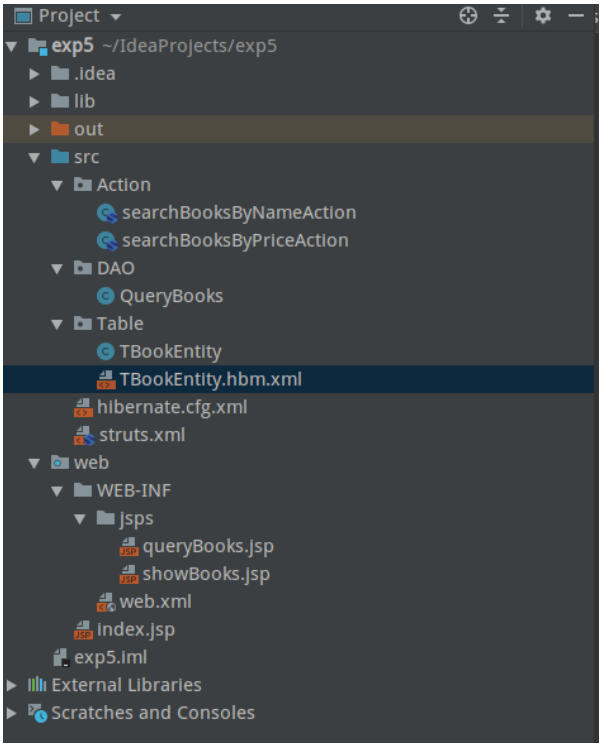

图 7. 使用 Struts2 和 Hibernate 框架的目录树

3 .将前面两题改为使用动态实体模型来实现。

删除 TbookEntity 实体类和 TbookEntity.hbm.xml 映射文件，重新建立 Tbook_entity.hbm.xml 映射文件。

运行期的持久化实体没有必要一定表示为像 POJO 类或 JavaBean 对象那样的形式。Hibernate 也支持动态模型（在运行期使用 Map 的 Map）那样的实体表示。使用这种方法，不用写持久化类，只写映射文件就行了。配置好实体映射文件之后，就可以使用 Map 来使用动态实体。

```
<--! TBook_entity.hbm.xml -->
<hibernate-mapping>
    <class entity-name="TBook_entity" table="T_BOOK">
    <id name="bookid" column="BOOKID" type="java.lang.String">
        <generator class="assigned" />
    </id>
    <property name="bookname" column="BOOKNAME" type="java.lang.String"/>
    <property name="bookprice" column="BOOKPRICE" type="java.lang.Double"/>
    </class>
</hibernate-mapping>
```

```
<--! hibernate.cfg.xml -->
<hibernate-configuration>
    <session-factory>
    <mapping resource="TBook_entity.hbm.xml" />
    </session-factory>
</hibernate-configuration>
```

更换为动态模型后重新测试该图书检索系统，测试数据和结果如图 8，图 9，图 10，图 11 所示。

图 8. 使用动态实体模型价格区间查询

图 9. 使用动态模型价格区间查询结果

图 10. 使用动态实体模型图书模糊查询

图 11. 使用动态实体模型图书模糊查询结果

四、实验心得

本次实验，我使用了 Struts2 和 hibernate 这两种框架的组合来实现网页查询设计。通过本次实验，我了解了 Struts2 框架和 Struts1.x 框架的不同点和基础开发方法，掌握了 hibernate 框架的基本原理及使用方法，学会使用 HQL 进行查询数据库。

Struts2 是基于 WebWork 框架发展起来的，它的 Action 类可以实现一个 Action 接口，也可以实现其它接口，也可以继承 ActionSupport 基类，甚至不需要实现任何接口，只编写 execute 函数即可。在 Struts2 中，Action 为每一个请求产生一个实例，不会产生线程安全问题。Struts2 不依赖于容器，允许脱离容器单独被测试。在 Struts2 中表单数据不再使用单独的 ActionForm 类，而是直接在 Action 类中编写表单数据对应的属性。Struts2 的原理：用户输入，前端的表单数据被

FilterDispatcher 截获；FilterDispatcher 将表单数据转交给 Action，并封装在 Action 内；Action 调用 DAO 层的 JavaBean 访问数据库；Action 根据数据库结果返回要跳转到的前端页面逻辑名称给框架；最后框架根据逻辑名称找到相应的网页地址，进行跳转。总体上来说，Struts2 框架相比 Struts1.x 框架做出了较大的优化和简化，使代码的书写和维护变得更加清晰简单。

Hibernate 框架是一种基于对象关系映射思想的框架，框架通过读取 hibernate 配置文件来连接数据库；当操作 PO 时，框架通过 hibernate 映射文件来决定操作的表名和列名；框架执行相对应的 SQL 语句访问数据库。使用 hibernate 框架来操作数据库，大大降低了程序的耦合度，同时简化了访问数据库的操作，使得编写数据库操作代码更加简单方便。最后我还学会了使用动态实体模型，这个技术进一步简化了代码的书写，并且减少了冗余代码以及提高了代码的可重用性。