实验五 Hibernate 基础应用——登录用户的信息增删改查模块

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/connector/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 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	TINYINT(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

3、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

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

5、在 Eclipse 中新建 Java 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/orm 站点，下载 Hibernate 发布版（如：hibernate-release-*.*.*.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

式来操作数据库：

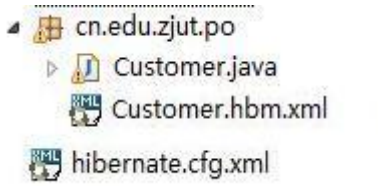

图 5-1 Hibernate 相关文件

（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，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>
```

（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，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>
      <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.MySQL8Dialect</property>
       <mapping resource="cn/edu/zjut/po/Customer.hbm.xml" />
   </session-factory>
</hibernate-configuration>
```

9、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，代码如下：

```
package cn.edu.zjut.dao;
import java.util.List;
import org.hibernate.Query;
import org.hibernate.Session;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

public class CustomerDAO {
    private Log log = LogFactory.getLog(CustomerDAO.class);
    private Session session;

    public void setSession(Session session) {
        this.session=session;
    }

    public List findByHql(String hql) {
        log.debug("finding LoginUser instance by hql");
        try {
            String queryString = hql;
            Query queryObject = session.createQuery(queryString);
```

```
            System.out.println("customer is found");
            return queryObject.list();
        } catch (RuntimeException re) {
            log.error("find by hql failed", re);
            System.out.println("customer is not found "+re);
            throw re;
        } finally{
        }
    }
}
```

10、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，代码如下：

```
package cn.edu.zjut.service;
import java.util.List;
import org.hibernate.SessionFactory;
import org.hibernate.Session;
import org.hibernate.cfg.Configuration;
import cn.edu.zjut.po.Customer;
import cn.edu.zjut.dao.CustomerDAO;

public class UserService {
    public Session getSession() {
        SessionFactory sf= new
Configuration().configure().buildSessionFactory();
        return sf.openSession();
    }

    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 +"'";
        Session session=this.getSession();
        CustomerDAO dao = new CustomerDAO();
        dao.setSession(session);
        List list = dao.findByHql(hql);
        session.close();
        if(list.isEmpty())
            return false;
        else
            return true;
    }
}
```

11、在 hibernate-prj1 中新建 cn.edu.zjut.app 包，并在其中创建 HibernateTest.java，用于测试登录逻辑，代码如下：

```
package cn.edu.zjut.app;
import cn.edu.zjut.dao.*;
import cn.edu.zjut.po.*;
import cn.edu.zjut.service.*;

public class HibernateTest {
    public static void main(String[] args) {
        Customer loginUser = new Customer();
        loginUser.setCustomerId(1);
        loginUser.setAccount("zjut");
        loginUser.setPassword("Zjut");

        UserService uService =new UserService();
        if(uService.login(loginUser))
            System.out.println(loginUser.getAccount()+" login
Success!");
        else
            System.out.println(loginUser.getAccount()+" login
Fail!");
    }
}
```

12、运行测试类 HibernateTest，观察控制台的输出，并记录运行结果；

13、修改 Hibernate 配置文件 hibernate.cfg.xml，增加打印 sql 语句的相关配置，代码片段如下：

```
<hibernate-configuration>
    <session-factory>
    ……
        <!--打印 sql 语句-->
        <property name="hibernate.show_sql">true</property>
        <!--格式化 sql-->
        <property name="hibernate.format_sql">true</property>
    </session-factory>
</hibernate-configuration>
```

14、运行测试类 HibernateTest，观察控制台的输出，并记录运行结果；

15、在 hibernate-prj1 中新建 cn.edu.zjut.util 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，代码如下：

```
package cn.edu.zjut.util;
import org.hibernate.HibernateException;
import org.hibernate.Session;
```

```
import org.hibernate.SessionFactory;
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;
    }
```

```
}
```

16、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：

```
public class HibernateUtil {
    ……
    public static Session getSession() throws HibernateException {
        Session 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();
        }
    }
}
```

17、在 cn.edu.zjut.service 包中修改 UserService.java 的 getSession()方法，使用辅助类 HibernateUtil 获取 Session 对象，代码片段如下；

```
package cn.edu.zjut.util.HibernateUtil;
public class UserService {
    public Session getSession() {
        return HibernateUtil.getSession();
    }
    ……
}
```

18、在 cn.edu.zjut.service 包中修改 UserService.java 的 login()方法，使用辅助类 HibernateUtil 关闭 Session 对象，代码片段如下；

```
package cn.edu.zjut.util.HibernateUtil;
public class UserService {
    ……
    public boolean login(Customer loginUser) {
        ……
        dao.setSession(session);
```

```
        List list = dao.findByHql(hql);
        HibernateUtil.closeSession();
        ……
    }
}
```

19、运行测试类 HibernateTest。

（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，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；

5、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；

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；

7、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；

8、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；

9、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 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

2、在表 item 添加 2 条记录，具体如表 5-4 所示：

表 5-4 item 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	Spring框架、Struts/Spring MVC框架、Hibernate/MyBatis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;

    public Item() {
    }
    public Item(String itemID) {
        this.itemID = itemID;
    }
    public Item(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
}
```

4、在 Item.java 的同一目录下中创建 Hibernate 映射文件 Item.hbm.xml，并参照 Customer.hbm.xml 填写 Item.hbm.xml 中的内容，代码略；

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：

```
<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，代码片段如下：

```
……
public class ItemDAO {
    private static final Log log = LogFactory.getLog(ItemDAO.class);
    ……
    public List findAll() {
        log.debug("finding all Item instances");
        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{
        }
    }
}
```

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，代码片段如下：

```
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() {
        Session session=this.getSession();
        ItemDAO dao = new ItemDAO();
        dao.setSession(session);
        List items = dao.findAll();
        HibernateUtil.closeSession();
        return items;
    }
}
```

8、修改 cn.edu.zjut.app 中的 HibernateTest.java，测试“显示所有商品信息”功能，代码略；

9、运行测试类 HibernateTest，观察控制台的输出，并记录运行结果；

10、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
}
```

11、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;
    ……
}
```

12、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>
```

13、修改测试类 HibernateTest.java，使其能正确显示书本的编号与书名，代码略；

14、运行测试类 HibernateTest，观察控制台的输出，并记录运行结果；

15、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
}
```

16、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
}
```

17、修改 Hibernate 映射文件 Customer.hbm.xml，将 customer 表与两个类（Customer 和 ContactInfo）映射，并修改主键的生成方式，代码片段如下：

```
<hibernate-mapping>
    <class name="cn.edu.zjut.po.Customer" table="customer"
catalog="hibernatedb">
        <id name="customerId" type="int">
            <column name="customerID" />
            <generator class="increment" />
        </id>
```

```
        ……
    <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>
```

18、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");
        try {
            session.save(customer);
            log.debug("save successful");
        } catch (RuntimeException re) {
            log.error("save failed", re);
            throw re;
        } finally{
        }
    }
}
```

19、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 boolean register(Customer loginUser) {
        Session session=this.getSession();
        CustomerDAO dao = new CustomerDAO();
```

```
        dao.setSession(session);
        Transaction tran = null;
        try {
            tran = session.beginTransaction();
            dao.save(loginUser);
            tran.commit();
            return true;
        } catch (Exception e) {
            System.out.println("save customer failed "+ e);
            return false;
        } finally {
            HibernateUtil.closeSession();
        }
    }
}
```

20、修改测试类 HibernateTest.java，使其能测试用户注册逻辑，代码略；

21、运行测试类 HibernateTest，观察数据库的变化，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）结合实验过程，总结粒度设计的方法及特点；

（7）碰到的问题及解决方案或思考；

（8）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——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 cn.edu.zjut.service 包中修改 ItemService.java，使用 from 子句实现简单查询，获取所有商品信息，代码片段如下：

```
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() {
        Session session=this.getSession();
        ItemDAO dao = new ItemDAO();
        String hql = "from cn.edu.zjut.po.Item";
        List list = dao.findByHql(hql);
        HibernateUtil.closeSession();
        return list;
    }
}
```

2、修改测试类 HibernateTest.java，使其能测试“获取所有商品信息”功能，代码略；
3、运行测试类 HibernateTest，观察控制台的输出，并记录运行结果；
4、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";
```

5、运行测试类 HibernateTest，观察控制台的输出，并记录运行结果；

6、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";
```

7、运行测试类 HibernateTest，观察控制台的输出，并记录运行结果；

8、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.ipk.title from Item as item";
```

9、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，根据该变化修改并运行测试类 HibernateTest，观察控制台的输出，并记录运行结果；

10、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.ipk.title, item.cost from Item as item";
```

11、若 select 子句返回多个属性，则返回结果将封装到 List<Object[]>集合中，根据该变化修改并运行测试类 HibernateTest，观察控制台的输出，并记录运行结果；

12、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）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；

（3）碰到的问题及解决方案或思考；

（4）实验收获及总结。

2、上交程序源代码，代码中应有相关注释。